文章编号：1005－8893（2006）02－0056－03

关于 Ehrlich 迭代法的一种推广①

黄清龙，阮宏顺
（江苏工业学院 信息科学系，江苏 常州 213164）

摘要：讨论 Ehrlich 迭代法的一种推广形式，给出收敛性定理及其简洁证明，并比较它和 Newton 迭代法的计算效率，得出当多项式的根全为单根时若多项式次数不低于 4，则 Ehrlich 迭代法的效率高于 Newton 迭代法；当多项式的根不全为单根时，则 Ehrlich 迭代法的效率总高于 Newton 迭代法。

关键词：Ehrlich 迭代法；多项式；重根；收敛性；
中图分类号：O 241.7　　文献标识码：A

On a Generalization of Ehrlich′s Method

HUANG Qing－long，RUAN Hong－shun
(Department of Information Science，Jiangsu Polytechnic University，Changzhou 213164，China)

Abstract：A generalized Ehrlich′s method is discussed；a version of its convergence theorem is proposed and a more concise proof of the theorem is given. Finally，the numerical efficiency of the generalized Ehrlich′s method and that of Newton method are compared. It is concluded that the Ehrlich′s method is more efficient than Newton method for polynomials of degree $n\geqslant 4$ with simple roots and for all polynomials with multiple roots.

Key words：Ehrlich′s method；polynomial；multiple root；convergence

对文献 [1] 提出的 Ehrlich 迭代法有文献 [2～6] 讨论其各种变形和推广形式。设 n 次多项式 $f(x)$ 在复数域上的典型分解式为

$$f(x)=(x-r_1)^{\mu_1}(x-r_2)^{\mu_2}(x-r_m)^{\mu_m} \tag{1}$$

该多项式的根满足 $r_i\neq r_j$（$i\neq j$），根 r_i 的重数 μ_i（$i=1，2，\cdots，m$）是正整数，且 $\sum_{i=1}^{m}\mu_i=n$。

对多项式（1），文献 [2] 给出了迭代公式

$$x_i^{(k+1)}=x_i^{(k)}-\frac{\mu_i\dfrac{f(x_i^{(k)})}{f'(x_i^{(k)})}}{1-\dfrac{f(x_i^{(k)})}{f'(x_i^{(k)})}\sum\limits_{\substack{j=1\\ j\neq i}}^{m}\dfrac{\mu_j}{x_i^{(k)}-x_j^{(k)}}} \tag{2}$$

其中 $i=1，2，\cdots，m$，$k=0，1，2，\cdots$。

$x_1^{(0)}$，$x_2^{(0)}$，…，$x_j^{(m)}$ 是多项式 $f(x)$ 的 m 个根的初始近似。迭代式（2）是求多项式重根时 Ehrlich 法的一种推广。特别当根的重数 μ_i（$i=1，2，\cdots，m$）都是 1 时，（2）式即为文献 [1] 中的 Ehrlich 迭代法。

文献 [2，3] 都讨论了（2）式的收敛性和收敛阶，但证明过程都较繁。本文给出（2）式收敛性定理的另外一种表述及更简洁的证明，并分析（2）式的计算效率，与 Newton 进行比较。

① 收稿日期：2005－12－22
作者简介：黄清龙（1963－），男，重庆忠县人，硕士，副教授，研究工作主要在应用数学和计算数学等方面。

1 收敛性、收敛阶的讨论

在（2）式两边同时减去 r_i，可将（2）式改写成

$$h_i^{(k+1)}=\frac{A_i^{(k)}}{1+A_i^{(k)}}h_i^{(k)} \tag{3}$$

其中 $h_i^{(k)}=x_i^{(k)}-r_i$

$$A_i^{(k)}=\frac{1}{\mu_i}\sum_{\substack{j=1\\ j\neq i}}^{m}\frac{\mu_j(r_j-x_j^{(k)})(x_i^{(k)}-r_i)}{(x_i^{(k)}-r_j)(x_i^{(k)}-x_j^{(k)})} \tag{4}$$

记

$$d=\min_{1\leqslant i<j\leqslant m}\{|r_i-r_j|\} \tag{5}$$

$$\lambda=n/\mu \tag{6}$$

其中 $\mu=\min_{1\leqslant i\leqslant m}\{\mu_i\}$，$n$ 为多项式次数。

定理 1：当迭代初值 $x_j^{(0)}$ 满足 $|x_j^{(0)}-r_j|<\frac{2d}{3+\sqrt{8\lambda-7}}$（$j=1,2,\cdots,m$）时，由（2）式产生的序列 $\{x_i^{(k)}\}_{k=0}^{\infty}$收敛到 r_i，且收敛阶至少为 3。

证明：当迭代初值 $x_j^{(0)}$ 满足 $|x_j^{(0)}-r_j|<\frac{2d}{3+\sqrt{8\lambda-7}}$ 时显然存在与 j 无关的常数 $s>\frac{3+\sqrt{8\lambda-7}}{2}$满足 $|x_j^{(0)}-r_j|\leqslant\frac{d}{s}$（$j=1,2,\cdots,m$），从而 $|x_i^{(0)}-r_j|\geqslant|r_i-r_j|-|x_i^{(0)}-r_i|\geqslant\frac{s-1}{s}d$，$|x_i^{(0)}-x_j^{(0)}|\geqslant|r_i-r_j|-|x_i^{(0)}-r_i|-|x_j^{(0)}-r_j|\geqslant\frac{s-2}{s}d$，所以

$$|A_i^{(0)}|\leqslant\frac{1}{\mu_i}\sum_{\substack{j=1\\ j\neq i}}^{m}\frac{\mu_j s^2|h_i^{(0)}|\cdot|h_j^{(0)}|}{(s-1)(s-2)d^2}\leqslant\frac{1}{\mu_i}\sum_{\substack{j=1\\ j\neq i}}^{m}\frac{\mu_j s^2}{(s-1)(s-2)d^2}\left(\frac{d}{s}\right)^2=\frac{1}{(s-2)(s-1)}\frac{1}{\mu_i}\sum_{\substack{j=1\\ j\neq i}}^{m}\mu_j \tag{7}$$

由常数 $s>\frac{3+\sqrt{8\lambda-7}}{2}$可推得

$$|A_i^{(0)}|\leqslant\frac{\lambda-1}{(s-1)(s-2)}<\frac{1}{2} \tag{8}$$

记

$$p=\frac{\lambda-1}{(s-2)(s-1)} \tag{9}$$

$$q=\frac{p}{1-p}\ (<1)$$

则得

$$|h_i^{(1)}|\leqslant\frac{|A_i^{(0)}|}{1-|A_i^{(0)}|}|h_i^{(0)}|\leqslant q|h_i^{(0)}| \tag{10}$$

从而当 $|h_j^{(0)}|\leqslant\frac{d}{s}$（$j=1,2,\cdots,m$）时，$|h_j^{(1)}|\leqslant q|h_j^{(0)}|\leqslant\frac{d}{s}$（$j=1,2,\cdots,m$）

一般地，当 $|h_j^{(k)}|\leqslant\frac{d}{s}$（$j=1,2,\cdots,m$）时，与（7），（8），（10）3 式类似地可得：

$$|A_i^{(k)}|\leqslant\frac{1}{\mu_i}\sum_{\substack{j=1\\ j\neq i}}^{m}\frac{\mu_j s^2|h_i^{(k)}|\cdot|h_j^{(k)}|}{(s-1)(s-2)d^2}\leqslant\frac{\lambda-1}{(s-1)(s-2)}<\frac{1}{2} \tag{11}$$

$$|h_i^{(k+1)}|\leqslant q|h_i^{(k)}|\leqslant\frac{d}{s} \tag{12}$$

其中 q 仍由（9）式确定。

因此由数学归纳法知在定理 1 的条件下，（12）式总成立。由（12）式即得 $|h_i^{(k)}|\leqslant q^k|h_i^{(0)}|\leqslant\frac{d}{s}q^k$，因 $0<q<1$，所以当 $k\to\infty$时，$h_i^{(k)}\to 0$，即 $x_i^{(k)}\to r_i$。

根据（11）式有 $|A_i^{(k)}|\leqslant\frac{1}{\mu_i}\sum_{\substack{j=1\\ j\neq i}}^{m}\frac{\mu_j s^2|h_i^{(k)}|\cdot|h_j^{(k)}|}{(s-1)(s-2)d^2}$，若记 $h^{(k)}=\max_{1\leqslant j\leqslant m}\{|h_j^{(k)}|\}$（$k=1,2,3,\cdots$），则

$$|A_i^{(k)}|\leqslant\frac{s^2}{(s-1)(s-2)d^2}\frac{1}{\mu_i}\sum_{\substack{j=1\\ j\neq i}}^{m}\mu_j\cdot h^{(k)^2}\leqslant\frac{s^2}{2d^2}h^{(k)^2}$$

因为 $|A_i^{(k)}|<1/2$，所以由 $|h_i^{(k+1)}|\leqslant\frac{|A_i^{(k)}|}{1-|A_i^{(k)}|}|h_i^{(k)}|$ 得 $|h_i^{(k+1)}|\leqslant\frac{s^2}{d^2}|h^{(k)}|^3$，从而（2）式的收敛阶至少为 3。

2 效率分析与数值实例

把（2）式右端第二项的分母变成 1 即得求重根的 Newton 迭代法，即

$$x_i^{(k+1)}=x_i^{(k)}-\frac{\mu_i f(x_i^{(k)})}{f'(x_i^{(k)})} \tag{13}$$

当迭代初值 $x_i^{(0)}$ 足够接近 r_i 时，Newton 迭代法是 2 阶收敛的[7]。

下面简单地分析一下本文讨论的迭代式（2）和上述 Newton 迭代法的计算效率。考虑每迭代一步的计算量时，忽略加减运算而只统计乘除运算的次数。结合各迭代法的收敛阶，则 Newton 迭代法的效率 $e_1=\frac{\ln 2}{2n-1}$；本文讨论的迭代式（2）的效率为 $e_2=\frac{\ln 3}{2n+m-1}$。比较知：

结论 1：当多项式（1）的根全为单根时，若 $n \leqslant 3$，则 $e_1 > e_2$；若 $n \geqslant 4$，则 $e_2 > e_1$。

结论 2：当多项式（1）的根不全为单根时，则总是 $e_2 > e_1$。

下面报告实际利用本文（2）式和 Newton 迭代法即（13）式求解 7 次多项式 $x^7 + x^5 - 10x^4 - x^3 - x + 10$ 的零点的数值结果。这个多项式零点的精确值是 $r_1 = 2$，$r_{2,3} = \pm 1$，$r_{4,5} = \pm i$，$r_{6,7} = -1 \pm 2i$。取迭代初值分别为 $x_1^{(0)} = 2.2$，$x_2^{(0)} = 1.2 + 0.1i$，$x_3^{(0)} = -0.8 - 0.1i$，$x_4^{(0)} = 0.1 + 1.2i$，$x_5^{(0)} = -0.1 - 0.8i$，$x_6^{(0)} = -1.1 + 2.2i$，$x_7^{(0)} = -1.1 - 1.8i$。编程用 MATLAB 语言，精度要求为 10^{-14}。

用（2）式只需 3 步迭代，所求零点全部都达到精度要求，数值结果见表 1；用 Newton 迭代法（13）式时需 6 步迭代，所求零点全部达到精度要求，在表 2 中只列出了 Newton 迭代法前 3 步的结果。对比表 1 和表 2 中的值可知（2）式收敛更快。

表 1 利用（2）式的数值计算结果

Table 1 The numerical results of the iterative formula (2)

第 1 步	$x_1^{(1)} = 1.993\ 451\ 778\ 690\ 92 - 0.007\ 287\ 382\ 904\ 64i$
	$x_2^{(1)} = 1.002\ 330\ 679\ 450\ 63 - 0.002\ 935\ 198\ 808\ 68i$
	$x_3^{(1)} = -0.988\ 507\ 307\ 381\ 19 + 0.005\ 820\ 018\ 431\ 58i$
	$x_4^{(1)} = -0.004\ 671\ 950\ 858\ 00 + 0.999\ 835\ 590\ 268\ 58i$
	$x_5^{(1)} = 0.001\ 991\ 257\ 097\ 17 - 0.995\ 882\ 713\ 259\ 27i$
	$x_6^{(1)} = -0.999\ 276\ 321\ 637\ 85 + 1.999\ 689\ 973\ 841\ 63i$
	$x_7^{(1)} = -0.999\ 954\ 751\ 699\ 68 - 2.000\ 134\ 860\ 190\ 83i$
第 2 步	$x_1^{(2)} = 1.999\ 999\ 770\ 164\ 47 - 0.000\ 000\ 420\ 073\ 15i$
	$x_2^{(2)} = 1.000\ 000\ 039\ 922\ 60 + 0.000\ 000\ 063\ 485\ 60i$
	$x_3^{(2)} = -0.999\ 999\ 305\ 719\ 22 - 0.000\ 000\ 027\ 105\ 85i$
	$x_4^{(2)} = 0.000\ 000\ 006\ 894\ 22 + 1.000\ 000\ 014\ 871\ 22i$
	$x_5^{(2)} = -0.000\ 000\ 000\ 620\ 27 - 0.999\ 999\ 997\ 652\ 96i$
	$x_6^{(2)} = -1.000\ 000\ 000\ 002\ 49 + 1.999\ 999\ 999\ 995\ 06i$
	$x_7^{(2)} = -1.000\ 000\ 000\ 000\ 00 - 2.000\ 000\ 000\ 000\ 00i$
第 3 步	$x_1^{(3)} = 2.000\ 000\ 000\ 000\ 00 - 0.000\ 000\ 000\ 000\ 00i$
	$x_2^{(3)} = 1.000\ 000\ 000\ 000\ 00 + 0.000\ 000\ 000\ 000\ 00i$
	$x_3^{(3)} = -1.000\ 000\ 000\ 000\ 00 + 0.000\ 000\ 000\ 000\ 00i$
	$x_4^{(3)} = 0.000\ 000\ 000\ 000\ 00 + 1.000\ 000\ 000\ 000\ 00i$
	$x_5^{(3)} = 0.000\ 000\ 000\ 000\ 00 - 1.000\ 000\ 000\ 000\ 00i$
	$x_6^{(3)} = -1.000\ 000\ 000\ 000\ 00 + 2.000\ 000\ 000\ 000\ 00i$
	$x_7^{(3)} = -1.000\ 000\ 000\ 000\ 00 - 2.000\ 000\ 000\ 000\ 00i$

表 2 利用 Newton 法的数值计算结果（前 3 步）

Table 2 The numerical results of Newton method (the first 3 steps)

第 1 步	$x_1^{(1)} = 2.063\ 921\ 439\ 433\ 44$
	$x_2^{(1)} = 1.025\ 409\ 742\ 995\ 52 + 0.018\ 708\ 409\ 060\ 03i$
	$x_3^{(1)} = -1.052\ 342\ 548\ 784\ 33 + 0.125\ 303\ 854\ 188\ 67i$
	$x_4^{(1)} = 0.034\ 795\ 470\ 284\ 14 + 1.046\ 724\ 666\ 308\ 63i$
	$x_5^{(1)} = 0.091\ 855\ 944\ 403\ 41 - 1.051\ 595\ 603\ 635\ 75i$
	$x_6^{(1)} = -1.030\ 195\ 810\ 908\ 23 + 2.063\ 767\ 122\ 683\ 98i$
	$x_7^{(1)} = -0.881\ 422\ 463\ 206\ 63 - 1.938\ 114\ 462\ 413\ 25i$
第 2 步	$x_1^{(2)} = 2.008\ 834\ 190\ 942\ 69$
	$x_2^{(2)} = 1.000\ 317\ 206\ 459\ 34 + 0.000\ 885\ 112\ 340\ 87i$
	$x_3^{(2)} = -0.983\ 403\ 714\ 981\ 90 + 0.025\ 456\ 633\ 858\ 41i$
	$x_4^{(2)} = 0.004\ 257\ 301\ 080\ 62 + 1.002\ 238\ 436\ 026\ 91i$
	$x_5^{(2)} = 0.015\ 101\ 025\ 661\ 26 - 0.995\ 818\ 832\ 484\ 05i$
	$x_6^{(2)} = -1.003\ 582\ 831\ 840\ 66 + 2.008\ 682\ 068\ 930\ 77i$
	$x_7^{(2)} = -1.051\ 161\ 428\ 050\ 89 - 1.980\ 563\ 400\ 666\ 23i$
第 3 步	$x_1^{(3)} = 2.000\ 197\ 049\ 943\ 97$
	$x_2^{(3)} = 0.999\ 999\ 318\ 802\ 00 + 0.000\ 000\ 562\ 484\ 16i$
	$x_3^{(3)} = -0.999\ 224\ 664\ 872\ 40 - 0.001\ 560\ 176\ 904\ 36i$
	$x_4^{(3)} = 0.000\ 031\ 241\ 815\ 41 + 0.999\ 984\ 407\ 546\ 05i$
	$x_5^{(3)} = -0.000\ 140\ 683\ 721\ 79 - 0.999\ 652\ 270\ 923\ 95i$
	$x_6^{(3)} = -1.000\ 053\ 254\ 584\ 45 + 2.000\ 184\ 780\ 132\ 78i$
	$x_7^{(3)} = -0.999\ 455\ 820\ 437\ 42 - 1.993\ 542\ 924\ 909\ 76i$

顺便指出，收敛性定理 1 中要求的条件即迭代初值都满足只是一个保证收敛的充分条件，更精细的估计将在今后讨论。

参考文献：

[1] Ehrlich L W. A Modified Newton Method for Polynomials [J]. Comm ACM, 1967, 10: 107－108.

[2] 魏木生，高利新. 一种同时求解多项式重根的迭代法及其收敛性 [J]. 华东师范大学学报，1998，(2)：16－21.

[3] Iliev A I. A Generalization of Obreshkoff－Ehrlich Method for Multiple Roots of Algebraic, Trigonometric and Exponential Equations [J]. Math Balkanica (N. S.), 2000, 14: 17－28.

[4] Wang Deren, Wu Yujiang. Some Modifications of the Parallel Halley Iteration Method and Their Convergence [J]. Computing, 1987, 38: 75－87.

[5] 黄清龙. 关于一个代数方程迭代解法的收敛性 [J]. 江苏工业学院学报，2003，15 (3)：58－60.

[6] 黄清龙. 一个修正的 Newton 法之改进 [J]. 高校计算数学学报，2002，24 (4)：313－319.

[7] 曹志浩，张玉德，李瑞遐. 矩阵计算和方程求根 [M]. 北京：人民教育出版社，1979.