广西科学 Guangxi Sciences 2007，14(2)：98～102

基于进化策略方法求多项式的根
Work Out the Roots of Polynomial Based on the Evolution Strategy

曹敦虔，张　明
CAO Dun-qian，ZHANG Ming

（广西民族大学数学与计算机科学学院，广西南宁　530006）
（Department of Mathematics and Computer Sciences，Guangxi University for Nationalities，Nanning，Guangxi，530006，China）

摘要：针对传统算法如牛顿迭代法在求多项式的根的过程中，只能对某一有限的区间求出数值解，对于一个根、重根或者是选择迭代初始点等问题的解决也不是很理想的弊端，提出一种在整个实数域（或复数域）上进行求根的进化策略算法．该算法充分发挥进化策略的群体搜索和全局收敛的特性，有效的解决了传统算法在求解过程中存在迭代初值选取难的问题，而且对系数为复（实）系数的高阶多项式求根的问题同样适用．模拟实验表明，该算法收敛速度快，精度高，比一般的求多项式根的智能算法还要好，是一种求多项式根的有效方法．

关键词：多项式　根　高阶　进化策略
中图法分类号：O174；TP18　　**文献标识码**：A　　**文章编号**：1005-9164(2007)02-0098-05

Abstract：It can be only worked out numerical solution in some limited interval during the process of solving the roots of polynomial by some traditional algorithm，such as Newton iteration method. And there are some shortcomings in solving the problems of single root，multiple root or choosing iterated initial point. According to above disadvantages，the author of this passage puts forward to an evolutionary strategy algorithm based on working out roots on real number field. This methods makes full use of the property of evolution strategy's population search and overall constringency and efficiently deals with the hard problems in iteration in traditional algorithm. It is also suitable for the complex higher order polynomial roots. Simulating experiments shows that this method has high speed of constringency and exactness and takes good advantages over the usual method of intelligent algorithms. It's an efficient method in working out the roots of polynomial.

Key words：polynomial，roots，higher order，evolution strategy

多项式求根问题是信号处理中经常遇到的问题．如滤波器和最小相位系统的设计、频谱分析、语音信号处理、信道编码与解码等等．四次以下的一元多项式在17世纪之前已有了公式解，但是直到今天，五次以上的一元多项式的求根问题还远未解决．尽管已经出现了一些数值计算意义下的求近似解的方法，如：二分法、弦截法、迭代法、牛顿法等[1]．但是，即使如此，任何一种方法却都有模糊的先决条件和其他一些局限性，比如牛顿法是一种化曲为直的求解方法，只要选取合适的初值，其收敛速度是很快的，但在迭代计算之前必须解决较难解决的选取合适的初值的问题，同时一个迭代公式仅可求出一个实根．

进化策略（Evolutionary Strategies，ES）是由德国柏林技术大学的 I. Rechenbery 和 H. P. Schweful 为研究风洞中的流体力学问题提出的[2]．进化策略的基本算法构成类似于遗传算法的构成形式，区别主要在于进化算子的不同选择，在遗传算法中主要采用交叉算子来产生新个体，而突变算子只是作为生成新个体的辅助手段，但在进化策略中则是主要采用突变算子来生成新个体，而交叉算子则较少使用．

本文基于进化策略的群体搜索和全局收敛的特性，提出一种能解决系数为复（实）系数的多项式在复数（实数）域上求根的问题的算法．模拟实验表明该算法收敛速度快、精度高，能有效的解决包括高阶多项式在内的多项式求根问题，而且无任何求根条件和局限性．

收稿日期：2006-11-02
修回日期：2007-01-21
作者简介：曹敦虔（1978-），男，助教，主要从事计算数学研究。

1 多项式全部根的算法

假设给定一个 n 次的实系数多项式

$$f(x)=a_0x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n.$$

设 $x_k(k=1,\cdots,n)$ 是 $f(x)$ 的根，则存在实数 $R>0$，使得 x_k 满足

$$|x_k|<R,(k=1,\cdots,n).$$

R 的选取方法很多，我们取

$$R=1+\max_{1\leqslant k\leqslant n}\{|\frac{a_k}{a_0}|\}.$$

令 $y=\frac{x}{R}$，则

$$f(x)=g(y)=b_0y^n+b_1y^{n-1}+\cdots+b_{n-1}y+b_n. \tag{1.1}$$

$g(y)$ 的根 y_k 满足 $|y_k|<1$，且 $f(x)$ 对应的根为 $x_k=Ry_k(k=1,\cdots,n)$. 这样，要求 $f(x)$ 的根，可先为求 $g(y)$ 的根，而 $g(y)$ 的根在 $(-1,1)$ 内，这样方便确定解求范围.

定理 1(代数基本定理)[3] 每个次数不小于 1 的复系数多项式在复数域中至少有一个根.

定理 2[3] 复系数一元 n 次多项式 $f(x)=0$ 在复数范围里有且仅有 n 个根.

定理 3[3] 如果实系数方程 $f(x)=0$ 有虚根 $a+bi$，这里 a 和 b 都是实数，$b\neq 0$，那么它还有另一个虚根 $a-bi$.

由以上定理知任何一元 n 次多项式在复数域上都有 n 个根，可以应用进化策略和求多项式全部根的算法把任何多项式的实根和复根求出.

为了能让进化策略算法求出 $g(y)$ 所有根，我们的方法是，先求出 $g(y)$ 的一个根 y_1，然后用多项式除法将 $g(y)$ 降次，即令

$$g'(y)=\frac{g(y)}{y-y_1},$$

则 $g'(y)$ 为 $n-1$ 次多项式，且其 $n-1$ 个根都是 $g(y)$ 的不同于 y_1 的根，再对 $g'(y)$ 用进化策略求出它的根. 重复这个过程，依次可求出 $y_2,\cdots,y_{n-1}$，从而可依次得到 $x_2,\cdots,x_{n-1}$，当 $n=1$ 时，多项式为一次的，可直接得出 y_n 和相应的 x_n.

2 进化策略算法

进化策略的实现过程可以简述为：

(1) 确定问题的表达方式. 每个个体由目标变量 X 和标准差 σ 两部分组成，每部分又可以有 n 个分量，即：

$$[X,\sigma]=[(x_1,x_2,\cdots,x_i,\cdots,x_n),(\sigma_1,\sigma_2,\cdots,\sigma_i,\cdots,\sigma_n)].$$

X 和 σ 之间的关系是：

$$\begin{cases}\sigma'_i=\sigma_i\cdot\exp(r'\cdot N(0,1)+r\cdot N_i(0,1)),\\ x'_i=x_i+\sigma_i\cdot N_i(0,1),\end{cases} \tag{1.2}$$

式中 (x_i,σ_i) 是父代个体的第 i 个分量，(x'_i,σ'_i) 是子代新个体的第 i 个分量，$N(0,1)$ 是服从标准正态分布的随机数，$N_i(0,1)$ 是针对第 i 个分量重新产生一次符合标准正态分布的随机数，r' 是全局系数，常取 1；r 是局部系数，常取 1. (1.2) 式表明，新个体是在旧个体基础上随机变化而来.

(2) 随机生成初始群体，并计算其适应度. 进化策略中初始群体由 μ 个个体组成，每个个体 (X,σ) 内又可以包含 n 个 x_i 和 σ_i 分量. 产生初始个体的方法是随机生成. 为了便于和传统的方法比较，可以从某个初始点 $(X(0),\sigma(0))$ 出发，通过多次突变产生 μ 个初始个体，该初始点可以从可行域中用随机方法选取. 初始个体的标准差 $\sigma(0)=3.0$.

(3) 根据进化策略，用下述操作产生新群体.

1) 重组：将两个父代个体交换变量和随机因子，产生新个体.

2) 突变：对重组后的个体添加随机量，产生新个体.

3) 计算新个体适应度.

4) 选择：根据选择策略，挑选优良个体组成下一代群体.

(4) 反复执行(3)，直到达到终止条件，选择最佳个体作为进化策略的结果.

2.1 重组算子

重组是以两个父代个体为基础进行信息交换，从 μ 个父代个体中随机选取两个个体：

$$(X^1,\sigma^1)=((x_1^1,x_2^1,\cdots,x_n^1),(\sigma_1^1,\sigma_2^1,\cdots,\sigma_n^1)),$$

$$(X^2,\sigma^2)=((x_1^2,x_2^2,\cdots,x_n^2),(\sigma_1^2,\sigma_2^2,\cdots,\sigma_n^2)),$$

然后对于目标变量 X 采用离散重组，对于策略因子 σ 采用中值重组. 即新个体的目标变量是从两个父代个体中随机选取，将两个父代个体的策略因子 σ 的平均值作为新个体的策略因子，构成新的个体：

$$(X,\sigma)=((x_1^{q_1},x_2^{q_2},\cdots,x_n^{q_n}),((\sigma_1^1+\sigma_1^2)/2,(\sigma_2^1+\sigma_2^2)/2,\cdots,(\sigma_n^1+\sigma_n^2)/2), \tag{1.3}$$

其中 $q_i=1$ 或 $2,i=1,2,\cdots,n$.

2.2 突变算子

进化策略的突变是在旧个体基础上添加一个随机量，从而形成新个体，突变的过程如(1.2) 式为：

$$\begin{cases}\sigma'_i=\sigma_i\cdot\exp(r'\cdot N(0,1)+r\cdot N_i(0,1)),\\ x'_i=x_i+\sigma_i\cdot N_i(0,1).\end{cases}$$

其中 $r=(\sqrt{2\sqrt{n}})^{-1}$，$r'=(\sqrt{2n})^{-1}$，$n$ 是个体中所含分量数目，r 和 r' 分别为全局步长系数及局部步长系数，用于对随机变量 $N(0,1)$ 进行缩放，它们相当于人工神经网络中的“学习率”，r 及 r' 通常取 1.

2.3 选择算子

进化策略中的选择体现了达尔文的“物竞天择，适者生存”的原则，采用确定型操作，它严格根据适应度的大小，将优质个体 100% 地保留，劣质个体 100% 地被淘汰. 一般采用 (μ,λ) 选择法，即从 λ 个子代新个体中确定性地择优挑选 μ 个个体（要求 $\lambda>\mu$）组成下一代群体，每个个体只存活一代，随即被新个体顶替.

2.4 适应度计算

进化策略中对于约束条件的处理，主要采用重复试凑法. 每当新个体生成，将其代入约束条件中检验是否满足约束条件. 若满足，则接纳新个体；否则，舍弃该新个体，借助重组、突变再产生一个新个体. 由于进化策略采用实数编码，这种检验比较直观和简单易行.

2.5 终止条件

终止方法有两种，一种是规定最大迭代数 N，当迭代资数达到 N，则停止操作，输出结果. 另一种是规定最小偏差 ε，对于适应度目标已知的进化策略算法，如用方差作为适应度计算的曲线拟合问题，可用最小的偏差 ε 制定终止条件，即

$$|f_{\max}-f^*|<\varepsilon,$$

式中 f^* 为已知的适应度目标；$f_{\max}$ 为每代最大的适应度. 也可以是两种方法的结合，既规定最大迭代次数，又设定精度要求，只要满足其中一种条件即终止循环.

2.6 进化策略求解多项式的近似因式分解

(1) 确定个体的表达方式. 表达式中个体由目标变量 X 和标准差 σ 两部分组成，每部分有 2 个分量，分别代表根的实部和虚部，即：

$$(X,\sigma)=((x_1,x_2),(\sigma_1,\sigma_2)).$$

(1.1) 多项式的根限定在模小于 1 内，所以 x_1 和 x_2 可取 $(-1,1)$ 之间的随机数.

(2) 随机生成初始群体. 进化策略中初始群体由 μ 个个体组成，每个个体 (X,σ) 内包含 2 个 x_i 和 σ_i 分量. 产生初始个体的方法是随机生成. 为了便于和传统的方法比较，可以从某个初始点 (X_0,σ_0) 出发，通过多次突变产生 μ 个初始个体，该初始点可以从可行域中随机选取. 初始个体的标准差 $\sigma_0=3.0$.

(3) 根据进化策略，用下述操作产生新群体.

1) 重组：新个体的目标变量是从两个父代个体中随机选取，将两个父代个体的策略因子 σ 的平均值作为新个体的策略因子.

2) 突变：对重组后的个体添加随机量，产生新个体.

3) 计算适应度：取适应度函数为 $e(x)=1/(1+|F(x)|)$，则适应度越接近 1，解的近似程度越好. 对每个个体 x_i，计算对应的适应度 e_i，$i=1,2,\cdots,\mu$.

4) 选择：采用 (μ,λ) 选择策略，挑选优良个体组成一代群体.

(4) 反复执行(3)，直到 $|1-\max\{e_i|i=1,2,\cdots,\mu\}|<\varepsilon$ 或迭代次数达到规定次数时停止，其中 ε 是一个较小的正数，表示精度要求.

求出多项式的一个根，然后将多项式降次，利用上面的算法再求出多项式的另一个根，重复此过程，求出多项式的全部根.

3 实例应用

例 1 求文献[4]中的实系数多项式 $f(x)=x^4-3x^3-4x^2+9x+1$ 的根.

用牛顿法求解的结果如表1所示，用进化策略算法求解（取 $\mu=15$，$\lambda=100$，$\varepsilon=0.999999$）的结果如表2所示.

表1 用牛顿法求 $f(x)=x^4-3x^3-4x^2+9x+1$ 根的结果

Table 1 The roots of $f(x)=x^4-3x^3-4x^2+9x+1$ by newton method

区间 Interval	初始点 Initial point	实根 Real roots
[−4,−1.157]	−2.579	−1.8057
[−1.157,0.725]	−0.359	−0.1065
[0.725,2.683]	1.859	1.5441
[2.683,5]	3.841	3.3681

表2 用进化策略算法求 $f(x)=x^4-3x^3-4x^2+9x+1$ 根的结果

Table 2 The roots of $f(x)=x^4-3x^3-4x^2+9x+1$ by evolution strategy

进化策略算法 Evolution strategy	MATLAB[5]
−0.10648804435749	−0.10648804475491
1.54410067806699	1.54410067830230
−1.80568534936514	−1.80568534855749
3.36807271565564	3.36807271501009

从表1和表2可以看到，牛顿法要选好区间和初始解，才能较快的找到解，而进化策略算法的初始根是随机地从[−1,1]均匀分布区间任意选择，避免了牛顿法初始点选择的繁琐的步骤，而且，用进化策略求解速度快，只用几十次就可以了，解的精度也是

很高的.从表2中可以看到,求得的解同用MATLAB算出的解基本一致,误差为10^{-9}.

例 2 求文献[6]中的复系数多项式

$$f(x)=x^5+(-4+10i)x^4+(7-40i)x^3+(4+70i)x^2+(-8+40i)x-80i$$

的全部根.

用进化策略算法求解,取 $\mu=15,\lambda=80,\varepsilon=0.999999$;结果如表 3 所示.

表3给出的结果在精度上比文献[6]中用遗传算法算出的结果高得多,演化代数150次远比1500次少的多.

表3 用进化策略算法求 $f(x)=x^5+(-4+10i)x^4+(7-40i)x^3+(4+70i)x^2+(-8+40i)x-80i$ 的根的结果

Table 3 The roots of $f(x)=x^5+(-4+10i)x^4+(7-40i)x^3+(4+70i)x^2+(-8+40i)x-80i$ by evolution strategy

进化策略 Evolution strategy	遗传算法[6] Genetic algorithm	真正的根 Exact roots
0.99999999999817 − 0.00000000000128i	1.000002	1
2.00000000000046 − 2.00000000000286i	1.99999 − 2.000012i	$2-2i$
−1.00000000007977 − 0.00000000003670i	−0.9999937	−1
2.00000000005948 + 2.00000000005131i	2.000012 + 2.000004i	$2+2i$
0.00000000002166 − 10.00000000001047i	−0.000001533218 − 9.999991i	$-10i$

表4 3种方法求 $f(x)=x^8-10x^6+33x^4-40x^2+16$ 根的结果

Table 4 The roots of $f(x)=x^8-10x^6+33x^4-40x^2+16$ by three methods

方法 Method	求出的根 Obtained roots	平均迭代次数 Iteration times	真正的根 Exact roots
递推分块神经求根器 Partition for recursive neural root solver	−2.00069220676009	第1次分快 Be first divide:8621	−1.0
	1.00000693410678		−1.0
	−1.99933983350341	第2次分快 Be second divide:2805	1.0
	0.999993446470696		1.0
	−0.999938058267339	第3次分快 Be third divide:930	−2.0
	2.00057729143192		−2.0
	−1.00006988438118	二阶直接法 Second order direct method:564	2.0
	1.99940899375285		2.0
直接神经求根器 Direct neural root solver	−2.00008812923599	29735	
	0.999965089833384		
	−1.99989207805059		
	1.00006032497794		
	−0.99998993865839		
	1.99999042628593		
	−1.00003612027881		
	1.99999042628593		
进化策略 Evolution strategy	0.99999999887693	240	
	−0.99999989416441		
	−2.00000006632024		
	2.00000009978792		
	−1.99999993366999		
	1.99999990020117		
	−1.00000010585571		
	1.00000000114433		

例 3 求文献[7]中的实系数多项式 $f(x)=x^8-10x^6+33x^4-40x^2+16$ 的根.

表 4 结果显示,进化策略算法($\mu=15,\lambda=100,\varepsilon=0.9999999$)求出的解比两种神经网络算出的解精度高,而且计算的平均次数要少得多.

例 4 用进化策略算法求高阶多项式 $f(x)=\sum_{i=0}^{52}x^i$ 的全部根,

取 $\mu=20,\lambda=140,\varepsilon=0.999999$;结果见表5.

表5 用进化策略算法求的 $f(x)=\sum_{i=0}^{52}x^i$ 根的结果

Table 5 The roots of $f(x)=\sum_{i=0}^{52}x^i$ by evolution strategy

求出的根 Obtained roots	求出的根 Obtained roots
−0.91514561724302− 0.40312342928797i	0.93741966211536+ 0.34820163348406i
−0.63008784358171+ 0.77652386271804i	0.67498300098591− 0.73783328004470i
0.02963332782256− 0.99956083650880i	0.67498300605428+ 0.73783328152424i
0.58297947911447+ 0.81248687800566i	−0.32026985630714+ 0.94732635548060i
0.75751124216162− 0.65282211819051i	0.99298109742061− 0.11827317041792i
−0.99824373176433+ 0.05924062789372i	0.14764656346445− 0.98904019007496i
0.14764656400251+ 0.98904018732217i	−0.86104361174831− 0.50853111938590i
−0.43006520227651− 0.90279782996574i	−0.91514561936438+ 0.40312342934893i
0.88965709099472+ 0.45662923739370i	0.26358716389011+ 0.96463558448230i
−0.86104361176734+ 0.50853111864922i	0.82940568612545− 0.55864676679841i
0.37582758211426− 0.92668960743181i	−0.95640098483802− 0.29205677158302i
−0.71750725704435− 0.69655102906301i	0.82940568918467+ 0.55864676451478i
−0.20597861874103+ 0.97855649229944i	0.26358716823790− 0.96463558150635i
0.88965709099471− 0.45662923739360i	−0.79485444282132+ 0.60680014629236i
0.75751124216175+ 0.65282211819044i	−0.63008784376822− 0.77652386317961i
−0.98423057794764− 0.17689027512239i	0.48279220373250+ 0.87573494724417i
−0.20597861874126− 0.97855649229924i	0.97202291431987− 0.23488604761583i
−0.43006520227690+ 0.90279782996618i	−0.08879589575350+ 0.99604984521246i
0.93741966113289− 0.34820163543449i	−0.32026985453739− 0.94732635392026i
−0.98423057794027+ 0.17689027513106i	−0.53382337998976+ 0.84559600309841i
0.37582758198332+ 0.92668960744917i	−0.99824373263959− 0.05924062831048i
−0.53382337786540− 0.84559600349227i	0.48279220276958− 0.87573494221300i
0.97202291408379+ 0.23488604600707i	0.58297947928046− 0.81248687754380i
−0.71750725708646+ 0.69655102906922i	0.99298109627784+ 0.11827317030028i
−0.79485444139161− 0.60680014593323i	−0.95640098428912+ 0.29205676895461i
0.02963332774775+ 0.99956083623123i	−0.08879589644877− 0.99604984354622i

由表5可以看出，用进化策略算法不但能计算低阶多项式，还能计算高阶多项式，速度快，精度高.

4 结论

利用进化策略的群体搜索和全局收敛的特性，提出在整个实数域(或复数域)上进行求根的进化策略算法，能有效的解决了传统算法在求解过程中存在迭代初值选取难的问题，而且能解决系数为复(实)系数的高阶多项式在复数(实数)域上求根的问题，比一般的求多项式根的智能算法还要好. 该算法收敛速度快，精度高.

参考文献：

[1] 《现代应用数学手册》编委会. 现代应用数学手册计算与数值分析卷[M]. 北京：清华大学出版社，2005.

[2] SCHWEFEL H P，BACK T. Evolution Strategies Ⅱ：Theoretical Aspects [M]. In：Genetic Algorithms in Engineering and Computer Science. Winter G (ed). Wiley，1995，127-140.

[3] 北京大学数学系几何与代数教研室代数小组. 高等代数[M]. 北京：高等教育出版社，2002.

[4] 周智恒. 一元实系数多项式方程实根的求解问题[J]. 华南理工大学学报：自然科学版，2002，30(5)：8-11.

[5] 王沫然. MATLAB 与科学计算[M]. 第2版. 北京：电子工业出版社，2004.

[6] 程锦松. 求多项式全部根的遗传算法[J]. 微机发展，2001(1)：1-2.

[7] 黄德双，池哲儒. 基于神经网络的递推分快方法求任意高阶多项式的根[J]. 中国科学：E 辑，2003，33(12)：1115-1124.

(责任编辑：邓大玉)

(上接第97页 Continue from page 97)

Subcase 2. 1 $f(x) \geqslant x$ for each $x \in [b,c]$. We can prove it by using the same method as Lemma 2. 6.

Subcase 2. 2 $f(x) \geqslant x$ for each $x \in [b,c]$. There exist $e_1 = b < e_2 = c < e_3 \in Fix(f)$ satisfying the three conditions of Lemma 2. 4. It follows that f is turbulent.

The Main Theorem is obtained by Theorems 2. 1 to 2. 4 in the following.

Main theorem Let $X = [0,1)$ and $f: X \to X$ be a continuous map. If f is pointwise chain recurrent, then

(1) f is identity if $Fix(f)$ is connected;

(2) f is turbulent if $Fix(f)$ is disconnected.

References:

[1] BLOCK L，FRANKE J. The chain recurrent set，attractors，and explosions[J]. Ergodic Theory Dynamical System，1985，5(4)：321-327.

[2] BLOCK L，COVEN E M. Maps of the interval with every point chain recurrent [J]. Proc Amer Math Soc，1986，98(3)：513-515.

[3] BLOCK L，COPPEL W A. Dynamics in one dimension [M]. Berlin：Spinger，1991.

[4] LI T，YE X D. Chain recurrent point of a tree map[J]. Bull Austral Math Soci，1999，59(2)：181-186.

[5] GUO W J，ZENG F P，HU Q Y. Pointwise chain recurrent maps of the space Y[J]. Bull Austra Math Soc，2003，67(1)：79-85.

[6] ZHANG G R，ZENG F P. Pointwise chain recurrent maps of the tree[J]. Bull Austra Math Soc，2004，69(1)：63-68.

[7] YE X D. The centre and the depth of the centre of a tree map[M]. Bull Austra Math Soc，1993，48(2)：347-350.

(责任编辑：蒋汉明　韦廷宗)