复平面上超越函数零点的数值计算*

龙云亮[1)] 文希理　谢处方

(成都电子科技大学)

AN IMPLEMENTATION OF A ROOT FINDING ALGORITHM FOR TRANSCENDENTAL FUNCTIONS IN A COMPLEX PLANE

Long Yun-liang Wen Xi-li Xie Chu-fang

(University of Electronic Science and Technology of China)

Abstract

In this paper, the transcendental equations in a complex plane are solved by using Kuhn's complementary pivoting procedure with contour integral. Some numerical examples show the effectiveness of the algorithm.

本文把围线积分与 Kuhn 法结合起来求解复平面上的超越方程，数值结果显示该算法的效果很好。

一、引　　言

超越方程的数值求解，即超越函数的零点计算，是数值分析方面的一个困难的问题。最常用的牛顿迭代法、Muller 插值法和梯度法等都对初值选取具有较高的要求，而对超越函数来说，要事先估计其零点的大概位置是很困难的。王则柯[1]用 Kuhn 算法[2]计算一个复变量的超越函数的零点。取得了较好的效果。但由于该方法对零点搜索的随机性和不能保证收敛性，因此不能保证找到在某个指定区域内的所有零点。鉴于 Kuhn 算法在求解多项式零点中的优越性[2-4]，本文尝试把围线积分和 Kuhn 法结合起来搜索复平面上超越函数的零点，取得了很好的结果。

二、算法概述

考虑方程 $f(z)=0$ 的求根问题，其中 $f(z)$ 为一超越函数，z 为复变量。

* 1991 年 10 月 30 日收到.

1) 现为中山大学电子系博士后.

设L为复平面C上的简单闭曲线，它不含f的零点，D为L的内部，$f(z)$在D内解析．令

$$s_k = \frac{1}{2\pi i}\oint_L z^k \frac{f'(z)}{f(z)} dz, \tag{1}$$

则

$$s_k = \sum_{j=1}^{n} \xi_j^k, \tag{2}$$

其中 ξ_j，$j = 1, 2, \cdots, n$ 为 $f(z)$ 在D中的所有的零点（按重数计算）．在式（2）中令 $k = 0$，结合式(1)有

$$n = \frac{1}{2\pi i}\oint_L \frac{f'(z)}{f(z)} dz. \tag{3}$$

我们将式(2)重写，有

$$\begin{cases} s_1 = \xi_1 + \xi_2 + \cdots + \xi_n, \\ s_2 = \xi_1^2 + \xi_2^2 + \cdots + \xi_n^2, \\ \cdots \\ s_n = \xi_1^n + \xi_2^n + \cdots + \xi_n^n. \end{cases} \tag{4}$$

为了从式(4)中解出 $\xi_1, \xi_2, \cdots, \xi_n$，设

$$\begin{cases} \sigma_1 = -(\xi_1 + \xi_2 + \cdots + \xi_n), \\ \sigma_2 = \xi_1\xi_2 + \cdots + \xi_{n-1}\xi_n, \\ \cdots \\ \sigma_n = (-1)^n \xi_1 \cdots \xi_n, \end{cases} \tag{5}$$

则有 Newton 恒等式

$$\begin{cases} s_1 + \sigma_1 = 0, \\ s_2 + s_1\sigma_1 + 2\sigma_2 = 0, \\ s_3 + s_2\sigma_1 + s_1\sigma_2 + 3\sigma_3 = 0, \\ \cdots \\ s_n + s_{n-1}\sigma_1 + s_{n-2}\sigma_2 + \cdots + s_1\sigma_{n-1} + n\sigma_n = 0. \end{cases} \tag{6}$$

显然 $\{s_j\}$ 等价于 $\{\sigma_j\}$，可以证明[5]，解方程组(4)等价于解代数方程

$$z^n + \sigma_1 z^{n-1} + \cdots + \sigma_n = 0. \tag{7}$$

而解此方程对 Kuhn 法来说是轻而易举的事[4]．

三、算法步骤

综上所述，在复平面上搜索超越函数 $f(z)$ 的零点的步骤如下：

① 先确定复平面上一区域 D，由式（3）计算出 $f(z)$ 在该区域内的所有零点数目 n，再由式(1)计算出 $s_j, j = 1, 2, \cdots, n$.

② 由 Newton 恒等式(6)得出 $\sigma_j, j = 1, 2, \cdots, n$.

③ 将 $\{\sigma_j\}$ 作为 n 阶首一多项式的系数，应用 Kuhn 算法求出该多项式的 n 个零点，即为超越函数 $f(z)$ 在区域D内的全部 n 个零点．

④ 重复步骤①,②,③,搜索另一区域 D' 内 $f(z)$ 的零点，直到完成预定区域的搜索任务.

四、数值计算

在我们所编的通用程序中,设区域D为一复平面上的矩形(可以很方便地改为其它形状),其X轴的起始坐标为 x_1 和 x_2, Y 轴的起始坐标为 y_1 和 y_2. 只要输入 x_1,x_2,y_1,y_2,围线积分的精度要求 ES 和 Kuhn 算法的精度要求 EK,即可将该区域内所求超越函数的零点全部求出(如果有的话).

本算法不存在初值选取的问题,可作为标准库过程调用.

我们对许多超越函数进行了数值计算,均取得了很好的效果,部分结果见算例. 所有例子都以单精度在 sun-286 微机上运算. 一般来说，ES 和 EK 应相当. 大部分计算时间都用在围线积分上,因此,区域D的大小应恰当,对大区域可分成几次搜索完成.

例 1.

$$f(z) = \sin z - \cos z$$

$x_1 = -20$, $x_2 = 20$, $y_1 = -1$, $y_2 = 1$, 此区域内共有 13 个根，按自然求根序列分别是:

$$\begin{aligned}
z_1 &= 0.7853733 - 2.712674 \times 10^{-5}i,\\
z_2 &= 7.068576 - 2.712674 \times 10^{-5}i,\\
z_3 &= 13.35178 - 2.712674 \times 10^{-5}i,\\
z_4 &= 19.63496 - 5.425347 \times 10^{-5}i,\\
z_5 &= -18.06418 - 2.712674 \times 10^{-5}i,\\
z_6 &= -11.78098 - 2.712674 \times 10^{-5}i,\\
z_7 &= -5.497776 - 2.712674 \times 10^{-5}i,\\
z_8 &= -2.356174 + 2.712674 \times 10^{-5}i,\\
z_9 &= -8.639378 + 2.712674 \times 10^{-5}i,\\
z_{10} &= -14.92258 + 2.712674 \times 10^{-5}i,\\
z_{11} &= 16.49338 + 2.712674 \times 10^{-5}i,\\
z_{12} &= 10.21018 + 2.712674 \times 10^{-5}i,\\
z_{13} &= 3.926975 + 2.712674 \times 10^{-5}i.
\end{aligned}$$

与精确解完全一致,没有漏根.

例 2.

$$f(z) = \sin z - \sin z_0 = (z - z_0)\prod_{k=1}^{\infty}[1 - (z + z_0)^2/\pi^2(2k-1)^2] \cdot [1 - (z - z_0)^2/\pi^2(2k)^2],$$

其中 $z_0 = 0.25 + 0.25i$, $x_1 = -10$, $x_2 = 10$, $y_1 = -1$, $y_2 = 1$,此区域内有 7 个根,即

$$\begin{aligned}
z_1 &= 0.2500041 + 0.2499980i,\\
z_2 &= 6.533189 + 0.2500041i,
\end{aligned}$$

$$z_3 = -6.033187 + 0.2500020i,$$
$$z_4 = 2.891593 - 0.2500020i,$$
$$z_5 = -9.674776 - 0.2499980i,$$
$$z_6 = -3.391593 - 0.2500020i,$$
$$z_7 = 9.174776 - 0.2500020i.$$

所得结果与精确解完全一致。

例 3.

$$f(z) = z - \pi + i + i\cos z$$

如取 $x_1 = -6$，$x_2 = 12$，$y_1 = -4$，$y_2 = 4$，则此区域内有 6 个根为

$$z_1 = 3.141595 + 1.616136i,$$
$$z_2 = 10.52170 + 2.812300i,$$
$$z_3 = -4.238513 + 2.812300i,$$
$$z_4 = -1.306342 - 2.232541i,$$
$$z_5 = 7.589525 - 2.232541i,$$
$$z_6 = 3.141588 + 3.390842 \times 10^{-6}i.$$

如取 $x_1 = 12$，$x_2 = 21$，$y_1 = -4$，$y_2 = 4$，则此区域内有 3 个根为

$$z_1 = 13.94644 - 3.093404i,$$
$$z_2 = 16.97361 + 3.368657i,$$
$$z_3 = 20.27308 - 3.545878i.$$

例 4.

$$f(z) = e^z - z^3$$

如取 $x_1 = 7$，$x_2 = 10$，$y_1 = -24$，$y_2 = 24$，则此区域内有 6 个根为

$$z_{1,2} = 8.670633 \pm 15.77071i,$$
$$z_{3,4} = 7.343272 \pm 8.931157i,$$
$$z_{5,6} = 9.572886 \pm 22.34781i.$$

如取 $x_1 = -1$，$x_2 = 10$，$y_1 = -25$，$y_2 = 25$，则此区域内有 10 个根为

$$z_1 = 1.857171 - 3.390842 \times 10^{-6}i,$$
$$z_{2,3} = -0.5543857 \pm 0.6193916i,$$
$$z_{4,5} = 7.343259 \pm 8.930980i,$$
$$z_{6,7} = 8.670634 \pm 15.77088i,$$
$$z_{8,9} = 9.572884 \pm 22.34776i,$$
$$z_{10} = 4.536404 - 3.390842 \times 10^{-6}i.$$

从前面的例子可以看出，该算法求得区域内超越方程的所有根，无一遗漏．所求根与精确解的相对误差一般都小于 10^{-5}．这主要是受到所使用的 286 微机的运算速度和有效位数的影响．如在小型机或中型机上计算，可以获得更高的精度．

衷心感谢中山大学王则柯教授的帮助．

参 考 文 献

[1] 王则柯,单纯不动点算法基础,中山大学出版社, 1986.

[2] Kuhn. H., in Fixed Points: Algorithms and Applications, Academic Press, 1977, 11--40.

[3] 王则柯,徐森林,逼近零点与计算复杂性理论,中国科学 (A), 1(1984),8--15.

[4] 王则柯, Kuhn 算法的程序实施及数值实验,数值计算与计算机应用, 2:3(1981),175--181.

[5] 徐森林,王则柯,代数方程组和计算复杂性理论,科学出版社, 1989.