УДК 519.6

ВЫЧИСЛЕНИЕ НУЛЕЙ ПЕРВОГО ПОРЯДКА КОМПЛЕКСНЫХ АНАЛИТИЧЕСКИХ ФУНКЦИЙ С БОЛЬШИМИ ВЕЛИЧИНАМИ ПРОИЗВОДНЫХ

В. В. Протопопов[1]

Существуют практически важные типы комплексных аналитических функций с большими значениями модулей производных в нулях. Вычисление нулей таких функций представляет проблему для традиционно используемых для этого методов вследствие больших значений производных. Другой эффективный алгоритм для вычисления нулей этого типа разработан на базе контурного интегрирования аргумента комплексной функции. Изменения аргумента вдоль контура значительно меньше изменений производной, что делает предложенный алгоритм эффективным. Положение максимума приращения аргумента вдоль контура интегрирования дает начальное приближение для вычисляемого нуля, а его точное значение определяется последующим уточнением.

Ключевые слова: аналитические функции, нули функций, контурное интегрирование, функции комплексного аргумента.

1. Introduction. In recent years the methods of computational electrodynamics became an essential part of many practical technologies in semiconductor industry, photonics, and radio communications. In particular, numerical methods for modeling the propagation of electromagnetic waves through layered structures and for computing the reflectivity of diffraction gratings or the polarization properties of wire grid polarizers require solving the so-called Rytov dispersion equation [1]. This equation allows one to determine the complex propagation parameter z with which an electromagnetic wave of wavelength λ can propagate through the layered structure composed of two layers:

$$F(z) = \beta(1 - e^{i\beta})(1 + e^{i\gamma}) + \gamma(1 + e^{i\beta})(1 - e^{i\gamma}) = 0, \tag{1}$$

where $\beta = \dfrac{2\pi a}{\lambda}\sqrt{\varepsilon_1 - z}$, $\gamma = \dfrac{2\pi b}{\lambda}\sqrt{\varepsilon_2 - z}$, a and b are the widths of the layers, and ε_1 and ε_2 are their complex permittivities. In general, equation (1) has an infinite sequence of roots. Let z_0 be the first-order root of this sequence. Then,

$$F(z) = F'(z_0)(z - z_0) + \varepsilon(z)(z - z_0), \tag{2}$$

where F' is the function derivative and $\lim\limits_{z\to z_0} \varepsilon(z) = 0$.

When applied to highly absorptive media such as metals, some roots may have extremely large modulus of the derivative $\bigl|F'(z_0)\bigr|$. To realize the possible scale of this situation, consider the following example: $\lambda = 632.8$ nm, $a = b = 400$ nm, $\varepsilon_1 = 1$, and $\varepsilon_2 = 3.57 - i\,4.36$ (chromium). Then one of the zeros is located approximately at $z_0 = -3.886359 - i\,0.5272905$ and $F(z_0) = 1.310818 - i\,7.959988$.

Changing the argument by only the 6th digit after the decimal point, i.e., putting $z_1 = z_0 - 10^{-6}$, we get the function value $F(z_1) = -38.67308 + i\,13.85763$. Thus, at this zero we have

$$\bigl|F'(z_0)\bigr| = \left|\frac{F(z_1) - F(z_0)}{z_1 - z_0}\right| = 4.55 \times 10^7.$$

For brevity, the zeros with large moduli of the derivatives are below called deep zeros. Clearly, the deep zeros may present difficulties for the efficient computation of their values by standard methods. To understand more clearly why this is so, a brief overview of the standard methods is presented in the next section.

2. Traditional methods for computing zeros. If we are given a polynomial, then all its zeros can be found by using a process called deflation. Suppose any one of the roots z_1 is known. Then, dividing the polynomial by $z - z_1$ and applying the algorithm used to find the first root, it is possible to find any second

[1] Samsung Electronics Co., Ltd, 416, Maetan-3Dong, Yeongtong-Gu, Suwon-City, Gyeonggi-Do, South Korea, 443-742; e-mail: proto.vladimir@samsung.com

root. Applying this procedure iteratively, we eventually find all the roots of the polynomial. The peculiarity of this procedure is that we do not care about the region in which the roots are located and, since the number of roots is finite, all the roots can be found consecutively one by one. The Muller algorithm [2] can be applied to find the first root as well as to find all consecutive roots.

This algorithm requires three guess points to start the search. Then the quadratic polynomial is fitted to these points and the single zero of this polynomial nearest to the last guess point is found. The process continues until a required tolerance is achieved. Working well for polynomials, the Muller algorithm with deflation cannot be applied to our problem by the two reasons: in our case the number of roots is infinite and the algorithm quickly diverges out of a prescribed region, leaving some roots undiscovered. Instead of the Muller algorithm, the cubically converging Halley method [3] can be used. However, this method works even worse in the case of deep roots, since it requires function derivatives.

In 1967 L. M. Delves and J. N. Lyness [4] proposed the algorithm based on the contour integration of the logarithmic derivative of a function; nowadays this algorithm becomes a standard for the most of the algorithms for finding zeros in a prescribed region. In it the number of zeros lying inside the contour C is determined by the formula

$$N = \frac{1}{2\pi i}\int\limits_C \frac{F'}{F}\,dz, \tag{3}$$

whereas the zeros themselves are determined by solving the generalized eigenvalue problem for the matrices composed of the elements $s_k = \dfrac{1}{2\pi i}\displaystyle\int\limits_C z^k\,\frac{F'}{F}\,dz$. However, an attempt to apply this concept to the functions of type (1) fails, since the derivatives happen to be so large that the execution of the code stops, unable to complete integration accurately. In order to overcome this difficulty, consider another approach discussed in the next section.

3. The basic idea of the method for computing deep zeros. Instead of (2), the number of zeros within the contour C may be computed using the principle of argument [5] by integrating the function argument over the contour: $N = \dfrac{1}{2\pi}\displaystyle\int\limits_C d\Big\{\arg\big[F(z)\big]\Big\}$. During the integration we have to compute the values of the argument variances $d\Big\{\arg\big[F(z)\big]\Big\}$ and to sum them over to complete integration. Since the argument variances are of orders of magnitude smaller than any possible function variances, there are no computational difficulties to complete such an integration. This is the first advantage of the method we proposed.

Additionally we can examine the behavior of the variances as a function of the running argument along C in order to obtain some estimates for the zeros. In particular, such an estimate is most efficient when there is only one zero within C. Consider the example illustrated in Fig. 1.

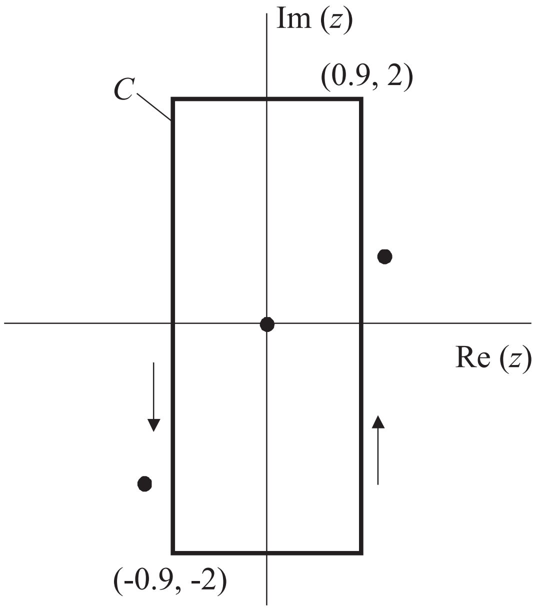

Fig. 1. Integration over the rectangle with the left lower corner at $(-0.9, -2)$ and right upper corner at $(0.9, 2)$

Let the function be $F(z) = \big(z - (1, 0.8)\big)\big(z + (1, 1.5)\big)z$ with the zeros at $(-1, -1.5)$, $(0, 0)$, and $(1, 0.8)$. For this case the argument variances along the vertical sides of the rectangle are shown in Fig. 2. On the rout along the left side of C, we have the leftmost zero at the right and the rightmost zero at the left, while on the rout along the right side of C the leftmost zero remains at the left and the rightmost remains at the right. Thus, the argument variances caused by the zeros situated outside the contour will always be of opposite signs along the parallel sides of the contour. This means that it is possible to isolate the central root by summing the variances along the opposite sides of the contour. This situation is illustrated by the solid line in Fig. 2. The maximum of the sum yields an estimate of the vertical coordinate of the central root. The same procedure can be applied for the horizontal sides of the contour. Thus, the second advantage of our method is that it is possible to approximately locate the inner zero simultaneously with counting the number of zeros within the contour.

When the contouring gives $N > 1$, the location of the inner

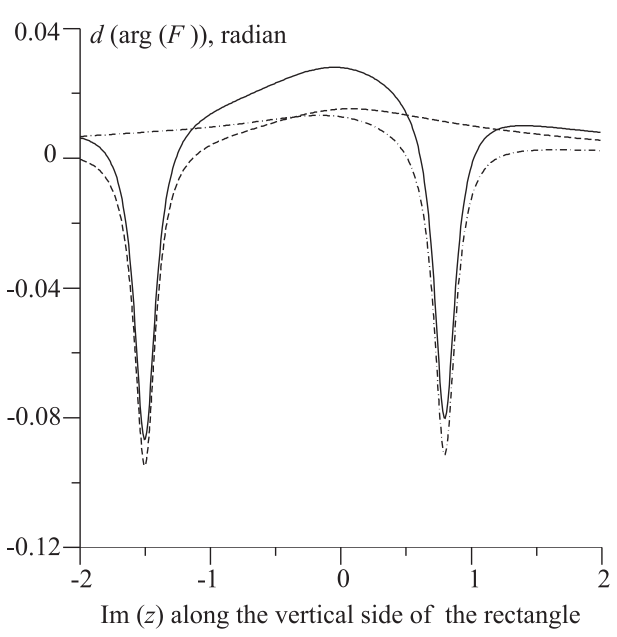

Fig. 2. Function argument variances along the left (dashed line) and right (dashed-dotted line) sides of the contour. The sum of them is shown by the solid line

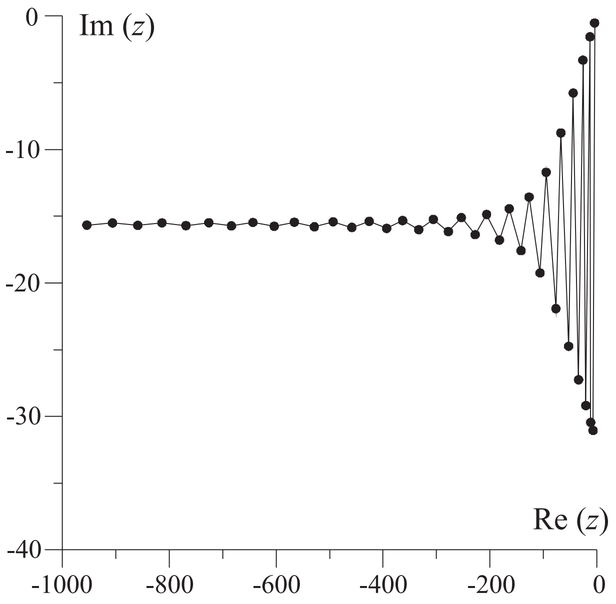

Fig. 3. Zeros of the function (1) connected by lines in order from left to right

zero will be obviously incorrect, so that in this case the result must be ignored, the contour narrowed, and the procedure repeated. When we have arrived at $N = 1$, we may get a sufficiently good estimate for the zero that can be used as an input for the refinement algorithm, such as the Muller method. If this zero happens to be deep, however, it is likely that the refinement algorithm will converge to another root outside the contour. In this case the procedure must be repeated with a new contour inside the initial one, and so on until the refinement algorithm converges within the contour. This is the basic idea of the method.

A few comments should be done regarding the practical implementation of the method. First, the argument can be computed unambiguously only within the $[0, 2\pi]$ interval. If it is necessary to compute the argument outside this interval with preserving the continuity of the function, then a procedure of stitching the values over the 2π discontinuities should be applied. In our case we have to calculate only small variances of the function argument and, consequently, there is no need in stitching over the 2π discontinuities. Therefore, it is possible to greatly simplify computations by using the following analytic form for the variance:

$$d\big[\arg(F)\big] = d\left\{\arctan\left[\frac{\mathrm{Im}\,(F)}{\mathrm{Re}\,(F)}\right]\right\} = \frac{1}{|F|^2}\Big\{\mathrm{Re}\,(F)\,d\big[\mathrm{Im}(F)\big] - \mathrm{Im}\,(F)\,d\big[\mathrm{Re}(F)\big]\Big\}.$$

Second, during integration of (3), the adaptive step change should be implemented in order to keep a necessary accuracy. For the sake of speed, it is worth to initially choose rather small number of subdivisions on the contour, say 100, and to increase it only if the function argument variance becomes too large. Finally, it should be noted that our code was written in Fortran and the refinement algorithm was chosen to be the Muller method implemented as the ZANLY subroutine of the IMSL library [6].

4. The algorithm.

1. Select the rectangle in which the zeros should be found: input the left lower corner and the upper right corner.

2. Compute the number N0 of zeros in the rectangle using (3). If N0=0, then quit with notification. If N0>1, then assign N=N0.

3. If N>1, then divide the original rectangle in two halves and compute N in the left rectangle. If N>1, then repeat.

4. If N=0, then swap the neighboring rectangles and repeat.

5. If N=1, then input a zero guess. Draw a maximum possible square around the zero guess and compute the integral (3) along it. If N=0, then perform a new subdivision within the current rectangle and repeat.

6. If N=1, then input a zero guess. Repeat 5 until the side of the square is less than TOL.

7. Call ZANLY and compute the zero. If zero is outside the square, then repeat 5.

8. If the zero is inside the square, then assign its ordinal number and save. If the zero ordinal number is less than N0, then repeat with 3, otherwise quit.

If at step 3 we choose the left rectangle, then the zeros will be numbered consecutively from their lowest real parts. Otherwise, the roots will be numbered counting from the largest real parts. The parameter TOL specifies the maximum radius of the region in which the ZANLY subroutine starts finding a zero. For a better convergence, TOL must be as small as possible, of the order of 10^{-3}.

5. Example of computation. Compute all zeros of function (1) with the parameters defined in Section 1 in the rectangle $(-1000, -35), (-0.1, -0.1)$. It is worth to first compute the results using single precision complex arithmetic in order to identify poorly determined zeros if they exist. The screen shot of the computer monitor is presented below.

```
Enter max value of phase difference (0.1 recommended)
0.1
Enter tolerance for root search (0.001 recommended)
0.001
Enter left lower corner
Enter Real Z1
-1000
Enter Imag Z1
-35
Enter right upper corner
Enter Real Z2
-0.1
Enter Imag Z2
-0.1
NUMBER OF ROOTS IN THE RECTANGLE: 39
root N 1 z: (-954.0978,-15.67441) F(z): (-7.4205680E-05,-2.1356277E-04)
root N 2 z: (-906.1952,-15.51160) F(z): (2.5762793E-06,7.2566904E-06)
root N 3 z: (-858.9707,-15.68921) F(z): (1.4134132E-05,-1.4892459E-04)
root N 4 z: (-813.6056,-15.50243) F(z): (-3.2297787E-05,8.8813154E-05)
root N 5 z: (-768.8445,-15.70736) F(z): (4.1521358E-05,-1.3975592E-04)
root N 6 z: (-726.0241,-15.49073) F(z): (1.5818043E-06,-4.4978238E-04)
root N 7 z: (-683.7178,-15.72994) F(z): (-8.8210363E-05,-1.9440080E-04)
root N 8 z: (-643.4515,-15.47551) F(z): (1.1982391E-05,-1.0729567E-04)
root N 9 z: (-603.5889,-15.75864) F(z): (3.2039232E-05,-1.3793542E-04)
root N 10 z: (-565.8890,-15.45525) F(z): (-4.7004301E-07,-1.4687330E-04)
root N 11 z: (-528.4549,-15.79584) F(z): (-4.3780467E-05,1.2451979E-04)
root N 12 z: (-493.3387,-15.42763) F(z): (1.0861555E-04,-2.2022270E-04)
root N 13 z: (-458.3121,-15.84539) F(z): (-5.1389587E-05,1.1799941E-04)
root N 14 z: (-425.8033,-15.38878) F(z): (-1.0057975E-06,5.5186033E-06)
root N 15 z: (-393.1548,-15.91345) F(z): (-3.5499619E-05,-2.0716935E-04)
root N 16 z: (-363.2877,-15.33224) F(z): (2.3619109E-06,4.3366649E-06)
root N 17 z: (-332.9739,-16.01065) F(z): (6.2358831E-06,-4.2859221E-05)
root N 18 z: (-305.7995,-15.24641) F(z): (-3.7556754E-05,8.2025239E-05)
root N 19 z: (-277.7549,-16.15638) F(z): (1.1361901E-04,1.1719217E-04)
root N 20 z: (-253.3523,-15.10928) F(z): (-7.1983745E-06,-1.9075919E-04)
root N 21 z: (-227.4732,-16.38908) F(z): (-8.8826157E-05,1.3628372E-04)
root N 22 z: (-205.9709,-14.87594) F(z): (-7.4295538E-05,-5.3421903E-05)
root N 23 z: (-182.0886,-16.79391) F(z): (-3.3633223E-06,1.9763254E-05)
root N 24 z: (-163.7012,-14.44615) F(z): (-1.4449892E-05,2.0809252E-04)
root N 25 z: (-141.5558,-17.58361) F(z): (-7.4822219E-06,-4.3670525E-06)
root N 26 z: (-126.6134,-13.57415) F(z): (-2.6654969E-05,-4.5174281E-07)
root N 27 z: (-106.0432,-19.25433) F(z): (1.3292051E-05,-4.8897433E-05)
root N 28 z: (-94.62355,-11.71493) F(z): (2.5323325E-05,-1.0517285E-04)
root N 29 z: (-76.38768,-21.93226) F(z): (-4.9749078E-06,-3.0144029E-05)
root N 30 z: (-67.06992,-8.763555) F(z): (4.1825924E-04,-3.7350567E-04)
root N 31 z: (-52.58426,-24.75001) F(z): (1.0822387E-06,4.7372432E-06)
root N 32 z: (-44.02319,-5.777972) F(z): (2.5991166E-03,-3.3440022E-04)
root N 33 z: (-33.97283,-27.25909) F(z): (-1.6793897E-06,-1.1425575E-05)
root N 34 z: (-25.90372,-3.311430) F(z): (8.9612361E-03,-1.4696946E-02)
root N 35 z: (-20.26427,-29.19605) F(z): (1.5921086E-06,-5.3371036E-06)
root N 36 z: (-12.61823,-1.565052) F(z): (-0.2869364,-1.150095)
root N 37 z: (-11.27158,-30.45336) F(z): (3.4242094E-06,4.6095661E-06)
root N 38 z: (-6.819347,-31.05658) F(z): (-1.4922171E-05,-4.3640553E-06)
root N 39 z: (-3.886359,-0.5272903) F(z): (5.136904,-2.359382)
Press any key to continue
```

Zeros of function (1) computed with double precision complex arithmetic

N	Re (z)	Im (z)	$\left\|F(z)\right\|$
1	-9.540977595801235E+02	-1.567441892790350E+01	3.019806626980426E-13
2	-9.061951600258290E+02	-1.551160305353389E+01	4.407656439090835E-13
3	-8.589707521042559E+02	-1.568921246934664E+01	1.058356923037507E-12
4	-8.136055862893412E+02	-1.550243436871120E+01	8.316689495813723E-14
5	-7.688445659287260E+02	-1.570735247162385E+01	1.314504061156185E-13
6	-7.260240578166776E+02	-1.549073181136085E+01	4.532460716807538E-13
7	-6.837178786816493E+02	-1.572995328406492E+01	4.096727173092940E-13
8	-6.434514478657668E+02	-1.547550549695704E+01	6.224046131325608E-13
9	-6.035888527659774E+02	-1.575863892051898E+01	4.069369499369587E-13
10	-5.658890251610026E+02	-1.545525325693889E+01	1.020173347323357E-12
11	-5.284548728772937E+02	-1.579585045153972E+01	7.105516174887867E-13
12	-4.933386849445413E+02	-1.542762137124049E+01	6.597055448962145E-13
13	-4.583121140260413E+02	-1.584539230377967E+01	4.041982792082277E-13
14	-4.258033493600928E+02	-1.538878321613366E+01	4.392884133218002E-13
15	-3.931548055532865E+02	-1.591345425086025E+01	0.000000000000000E+00
16	-3.632876895465475E+02	-1.533224430719856E+01	1.286847625882703E-13
17	-3.329739321316593E+02	-1.601064508944336E+01	1.621843951300168E-13
18	-3.057994857658682E+02	-1.524640918914081E+01	1.040890542596457E-12
19	-2.777548756077671E+02	-1.615635926790910E+01	1.734655884282248E-13
20	-2.533523024042593E+02	-1.510928122338324E+01	1.082266451513747E-13
21	-2.274731795106826E+02	-1.638909466263839E+01	7.553172887309252E-14
22	-2.059708868136810E+02	-1.487594868718368E+01	1.558341878610840E-13
23	-1.820885593730619E+02	-1.679391235425890E+01	4.130936831003462E-13
24	-1.637012061928949E+02	-1.444614731698818E+01	1.336196464944824E-08
25	-1.415558316332628E+02	-1.758361327207739E+01	9.532931148469529E-14
26	-1.266134255576757E+02	-1.357415035972105E+01	4.130936831003462E-13
27	-1.060432102502674E+02	-1.925432687791370E+01	7.140866117879973E-14
28	-9.462355534667954E+01	-1.171492952110427E+01	1.171857100421693E-13
29	-7.638767757771870E+01	-2.193226709880078E+01	1.151683528066849E-11
30	-6.706992463373996E+01	-8.763556585803229E+00	2.470564346833809E-12
31	-5.258426206300209E+01	-2.475000745523743E+01	1.275379677289189E-12
32	-4.402319010608897E+01	-5.777972533683895E+00	0.000000000000000E+00
33	-3.397282848790300E+01	-2.725909520841754E+01	5.376231353587555E-14
34	-2.590371986838409E+01	-3.311430324622187E+00	4.152398636035003E-10
35	-2.026426450128012E+01	-2.919605094406653E+01	1.687916891722083E-08
36	-1.261822742683489E+01	-1.565052066804964E+00	0.000000000000000E+00
37	-1.127158308567147E+01	-3.045335690247878E+01	3.113695637850019E-14
38	-6.819347270830086E+00	-3.105658038915582E+01	1.902136147497003E-10
39	-3.886359268593807E+00	-5.272903785357645E-01	5.823134822479450E-05

The above result is also presented graphically in Fig. 3, showing the oscillating character of the sequence of zeros. The computation time on the Pentium4 3GHz computer is equal to 0.94 s, including displaying and writing the result into a file. It can be seen that in this sequence there are two deep zeros whose residuals are of order unity: $N = 36$ and $N = 39$. This is because the single precision arithmetic cannot compute more accurate values of an very abruptly varying function.

Double precision complex arithmetic substantially increase the accuracy of computation as is shown in the table. The last column of the table contains the values of function moduli computed at zeros. It is noteworthy that in this case the computation time is even shorter (0.89 s) than with single precision arithmetic. Probably, this is due to a better convergence of the Muller algorithm near deep zeros when double precision is used.

СПИСОК ЛИТЕРАТУРЫ

1. *Rytov S.M.* Electromagnetic properties of finely layered structure // J. of Experim. and Theor. Physics. 1955. **29**, N 5 (11). 605–616.
2. *Muller D.E.* A method for solving algebraic equations using an automatic computer // Mathem. Tables and Aids to Computation. 1956. **10**. 208–215.
3. *Traub J.F.* Iterative methods for the solution of equations. Englewood Cliffs: Prentice-Hall, 1964.
4. *Davis L.M., Lyness J.N.* A numerical method for locating the zeros of an analytic function // Mathematics and Computation. 1967. **21**, N 100. 543–560.
5. *Carpentier M.P., Dos Santos A.F.* Solution of equations involving analytic functions // J. of Computational Physics. 1982. **45**, N 2. 210–220.
6. Visual Numerics. IMSL. Fortran Subroutines for Mathematical Applications. Math/Library, Vol. 1 & 2. Ch. 7. 841–842.

Поступила в редакцию
5.10.2007