PINPOINTING THE REAL ZEROS OF ANALYTIC FUNCTIONS

Soufiane Noureddine
Dept. of Mathematics and Computer Science, University of Lethbridge
4401 University Drive, Lethbridge, T1K 3M4, AB, Canada
soufiane.noureddine@uleth.ca

Abdelaziz Fellah
Dept. of Computer Science, University of Sharjah
P.O. Box: 27272, Sharjah, United Arab Emirates
fellah@sharjah.ac.ae

Abstract We present a new algorithm for extracting the real zeros of an analytic function. The algorithm makes use of known results of analytic function theory. The distinguishing feature of the algorithm is its ability to handle only zeros the analyst is interested in and to work with sufficient accuracy in cases where the number of zeros is extremely large. In the course of our discussion, we introduce two novel concepts: sweeping functions and size of polynomials.

Keywords: Zeros of analytic functions, Principle of argument, Bisection methods, Quadrature.

1. Introduction and Motivation

Perhaps there is no topic in applied mathematics that has occupied scientists more than solving equations. The task of solving equations concerned not only (pure) mathematicians, but also computer scientists, physicists, engineers, and scientists in many other disciplines. In computer science, for example, the topic is strongly emphasized in numerical methods of computing, and makes itself well visible in different application domains such as robotics, computer graphics, and computer vision, only to mention some.

One might ask now, why introduce a new method for finding zeros? In fact, the method presented here addresses a usually unmentioned and/or underestimated problem: *What are the values of the zeros we are interested in and how many are they*? It turns out that both questions are not easily answered. As a matter of fact, all known methods for zero finding will contribute only to a

certain extent to the solution of both problems. The trouble in working with well-known methods arises partly because (a) They cannot solve your problem. (b) They could solve your problem in some instances only. (c) They can solve your problem but you are misusing these methods. Following this classification we can easily notice that known methods are either of type (a) or (b) regarding the problem of finding real zeros of an analytic function. Methods that easily work if only real zeros are needed is therefore obvious.

Restrictions on the kind of zeros a user may be interested in is natural in real applications. For example, Henrici (Henrici, 1974) mentioned the stability of polynomials in control theory, which translates in zeros having negative real parts. The equation for modeling heat propagation (Greenleaf, 1972) is another example for an application where one is only interested in the real zeros contained in some domain.

We will restrict our discussion to real zeros of *analytic* functions. The rationale behind this assumption is that analytic functions are very common in practice and fortunately very well understood. We will be able, therefore, to make extensive use of the complex analysis results related to analytic functions. The problem statement of this paper can be stated as follows:

Given an analytic function $f(z)$, where z is a complex variable, under which conditions is it possible to locate the real zeros of $f(z)$ in a given real interval $[a, b]$? If said conditions are satisfied, then find an efficient and reliable algorithm that locates the real zeros in $[a, b]$ and/or their number.

An important aspect in this connection is that we are interested in *real zeros* (contained in some domain) only. There are algorithms that solve the same problem without the restriction to real zeros (Delves and Lyness, 1967(a); Lehmer, 1962). In fact, the intention of both algorithms is to determine good approximations for all zeros of an analytic function in a systematic way (but not simultaneously). In many applications, the function under study would be modeling some real world behavior and in many instances the pure complex zeros of $f(z)$ will not bear any meaning with regard to the modeled behavior. Thus, it is worthwhile to think about a general method for this type of applications.

The second assumption is liberating. We will omit the usual assumption tacitly made in this kind of work that the number of zeros N is tractable, that is, N is not too large. This will restrict our solution space even further but the authors are convinced that this step is needed in many cases (this will become clearer as we proceed in the paper). Other works like that of (Delves and Lyness, 1967(a)) and those based thereupon actually address the tractability of the number of zeros in a given domain, too. But the main solution approach there is bisection. The domain under examination is shortened step by step until it only includes a small number of zeros. Then, some direct method is used to locate the zeros, for example, Delves and Lyness (Delves and Lyness,

1967(a)) used polynomial root-finding. Bearing in mind that these methods attempt to locate *a*ll the zeros of the given function $f(z)$, it is obvious that the bisection algorithm, though extremely powerful for finding few zeros, is of no help if the number of zeros N is too high, *i.e.*, $N = 2^k$ for some k.

The price for the last assumption is that in our case it is not reasonable anymore to ask for the locations of all zeros. We will have rather to content ourselves with the determination of few zeros in question, or even coarser, with the mere information that the given interval contains at least one real zero. Obviously, if we can assure by some means that the number of zeros in the respective interval is not too large then precisely determining their locations can be easily achieved using a bisection scheme. Thus, we will not exclude the simpler case where N is relatively small in our discussion.

The paper is organized as follows. In the next section, we review some of the well-known zero finding methods. In Section 3, we sketch the general principles of our solution approach in a generic way. The formal algorithm of pinpointing the real zeros of analytic function is presented in Section 4. Section 5 reports on some experiments obtained in a prototypical implementation of the algorithm. Finally, in Section 6 we briefly elaborate on further investigations and conclude the paper with some critical remarks.

2. Related Work

Without claiming to be exhaustive, let us very briefly review some of the well-known zero finding methods and assess them with respect to our problem. Roughly, these methods can be classified as follows:

Rule of Signs for Polynomials

The Descartes rule of signs (see e.g., (Henrici, 1974)) is one of the oldest methods to quickly get a first rough upper bound of the number of real zeros of real polynomials (and easily generalizes to other entire real-valued functions with real coefficients). Sometimes the rule is used to get lower bounds for non-real zeros, too (Drucker, 1979). This rule simply says that the number of real zeros (counting multiplicities) in a real interval $[a, b]$ is either the number of coefficients sign changes, say C, of a certain polynomial (gained from the original one) or C minus an even number. Combined with a bisection scheme, for example, the method can be extended to a technique for the determination of real zeros of polynomials (e.g., using Vincent's algorithm (Vincent)). The main problem of this method, however, is that it is based on upper bounds of the number of zeros. Thus, the method is only conditionally applicable (in the sense of (Henrici, 1974)). A nice feature of this rule, however, is that the complexity of the determination of this upper bound *does not depend on the degree of the polynomial.* This feature however has no major practical relevance, since

a bisection scheme potentially leads to a high number of coefficients anyway. Better is to resort to Sturm sequences (Henrici, 1974) which takes $O(n^2)$ steps where n the degree of the input polynomial. Although exact, Sturm sequences are inefficient if n is too large.

Iterative Methods

One clearly thinks here of Fixed-Point methods with the well-known representative, namely, Newton's method (Henrici, 1974; Douglas and Burden, 2003). The latter has the nice feature of being quadratically convergent in contrast to the general fixed-point method, whose (potential) convergence is only linear. Newton's method converges locally only. That is, the first guess of the zero should be good enough. Rather than heuristically guessing the value of the zero, advanced methods use Newton's method only in the late stages of the zero finding process. Newton's method is not always convergent and converges to real zeros only, if the initial guess is real. The need for the derivative makes Newton's method sometimes inconvenient. Related methods are the Laguerre's method (Henrici, 1974), which may cubically converge to complex zeros even with a real starting value. If all zeros are real (e.g., eigenvalues of a symmetric matrix), the method is even globally convergent (Foster, 1981). However, Laguerre's method and more general methods like Schroeder's iteration methods (Henrici, 1974), which have arbitrarily large convergence rates, need more function evaluations per iteration, a fact that makes them less useful in practice. There are also iterative methods that determine *all* zeros of an analytic function simultaneously. Our method is somehow completely different, in the sense that we only select some of the zeros to be determined. Again, the main idea is to have an initial guess that is refined iteratively. If all zeros are known to be simple (Petcovic et. al., 1995) gives an iterative method that uses numerical integration for the determination of all zeros. Other similar methods have been presented in (Petcovic and Marjanovic, 1991; Atanassova, 1994; Olver, 1952).

Bisection Methods

The main idea of these methods is already present in simple algorithms like Regula Falsi and Secant methods (see any numerical analysis textbook, e.g., (Douglas and Burden, 2003)). They always converge to the bracketed zeros (if any). More general algorithms work equally well in the field of complex numbers (and other fields as well). Henrici (Henrici, 1974) provided a simple algorithm for determining the winding number of a closed Jordan curve in the complex plane. His algorithm, though only conditionally reliable, is the basis of many bisection methods for zero determination. Lehmer (Ying and Katz, 1988) used a direct numerical quadrature for the determination of the number

of zeros of an analytic function. He combined his basic algorithm with a bisection method that is based on circular contours. Many other known algorithms are based on the Lehmer's approach. The best representative is that of (Delves and Lyness, 1967(a); Delves and Lyness, 1967(b)). They handled the zero finding problem in a rigorous manner, used a scheme akin to Lehmer's method with an emphasis on entailed numerical errors, treated circles and squares as contours, and also provided a derivative-free method based on the ideas of Henrici. Li (Li, 1983) gave later another variant of Delves and Lyness method using homotopy continuation methods. Herlocker and Ely (Herlocker and Ely, 1995) exploited automatic differentiation and interval arithmetic to bound the error of numerical quadrature for the determining the zeros. An excellent treatment for the zeros of analytic functions which is based on numerical quadrature can be found in (Kravanja, Barel and Van, 2000).

For the above mentioned reason, our algorithm shall be based on a bisection scheme that exploits numerical quadrature. All known algorithms go into trouble if N (the number of zeros counting multiplicities) is too large. Our algorithm tries to mitigate this problem found in other algorithms. Also, the algorithm in its current form is able to extract the real zeros of an analytic function with relatively high accuracy. This feature is not found in other known algorithms. Moreover, under some circumstances the presented concepts generalize easily to the extraction of zeros contained in any bounded domain of the complex plane. It will be explained in the course of our treatment that the solution needs the (numerical) evaluation of some singular integrals. The reader should, however, keep in mind that we cannot make use of the available arsenal of methods that work well for this type of integrals for the simple reason that those methods assume the explicit knowledge of the singularities in question. Unfortunately, the singularities in our problem are the zeros to be evaluated, so they cannot be assumed to be known in advance.

3. Assumptions and Approach

Our approach will be based on the fundamental concept of complex analysis, that is, the so-called *Principle of Argument*. See for example, (Kravanja, Barel and Van, 2000 ; Delves and Lyness, 1967(a); Lehmer, 1962). The principle is stated as follows:

PRINCIPLE OF ARGUMENT 3.1 *Let $f(z)$ be a complex-valued function meromorphic in a simply connected region of the complex plane and γ a positively oriented Jordan curve not passing through any zero or pole of $f(z)$, then the number of zeros N and the number of poles P (counting multiplicities) of $f(z)$ in the interior of γ satisfy:*

$$N - P = \frac{1}{2\pi I} \oint_{\gamma} \frac{\frac{d}{dz} f(z)}{f(z)} dz \tag{1}$$

where $I^2 = -1$.

If the function $f(z)$ is analytic within the respective region, the principle of argument can be used to determine the number of zeros of $f(z)$ inside the curve γ. Even if $f(z)$ has some poles, and if we can write $f(z) = g(z) * h(z)$ where $g(z)$ does not have zeros and $h(z)$ does not have any poles, then the zeros of $f(z)$ are precisely those of $h(z)$, and the principle of argument can be used in this case for $h(z)$ instead of $f(z)$.

The principle of argument is not as elegant as it may seem, since the integrand in (1) is the logarithmic derivative of $f(z)$; the anti-derivative of which is the $\log(f(z))$, which is a multi-valued function in the field of complex numbers. Therefore, unless the considered function is relatively simple, the integral in (1) cannot be computed in closed form. Since we are to give an algorithm, that is, an automated procedure, for finding the zeros of $f(z)$, we can by no means assume that the function $f(z)$ (which is the input of our algorithm) is simple. We, therefore, shall have to resort to a numerical approximation of the above integral.

Using a quadrature scheme to evaluate the above integral will have a great impact on the permissible domain of functions in our context. We only are allowed to assume that the function $f(z)$ is analytic. However, this is not sufficient to get useful results using quadrature methods. The given function $f(z)$ is not only to be analytic but for practical reasons we also require that its logarithmic derivative is treatable by the used quadrature method. More formally, let

$$g(z) = f'(z)/f(z)$$

where $I(g)$ is the exact value of the integral in (1), and $R(g)$ is an approximated value achieved using the quadrature rule R, then the error, $E(g)$, made in using R to approximate $I(g)$ is:

$$E(g) = I(g) - R(g) \tag{2}$$

Thus, for a given quadrature rule R, the quadrature algorithm will only be useful if $E(g)$ is small enough. Hence, the algorithm to be presented in the next section will give good results if and only if $f(z)$ is an analytic function such that $E(g)$ is small enough. The fact that the sought value of the integral is an integer, makes extremely precise evaluation of $I(g)$ unnecessary. Even better, as we alluded to earlier, the algorithm we are going to present is targeted to functions where the number of zeros $I(g)$ is expected to be a large one, and we are often interested (if at all) in locating only some of the zeros. Thus, the approximation rule will be sufficient in our case for more problem instances than is usually the case[1].

[1]Often we will be interested only in knowing whether $I(g)$ is 0 or not.

Now, let us turn our attention to the main idea of the algorithm to be presented in Section 4. In more general terms, the problem statement could be rephrased as follows:

Given a function f analytic in a domain D and a bounded connected subdomain D' of D that does not have zeros of f on its boundary, find those zeros of f that lie inside D'. In principle, D' does not need to be part of the real axis (but this the case we are interested in). To find the zeros lying in D', we make use of what we call a *(zero) sweeping function* w. The function w should be analytic and reversible in D. Hence, we require that w^{-1} be defined throughout D. This calls for using a simple transformation like a (bi)linear one as a sweeping function. However, there is no need to restrict w to this kind of functions. The rationale behind the introduction of such a function w is to be able to "filter" the zeros of interest from the (perhaps many) zeros existing nearby. (The exact meaning of this filtering will be clear as we proceed). Analytically, we will treat the transformed function $f(w(z))$ instead of $f(z)$. We must however make sure that:
(a) The zeros of $f(w(z))$ are easier to approximate that those of $f(z)$.
(b) $w(z)$ and $w^{-1}(z)$ are easily computable for any z in the corresponding domain.

The reader should keep in mind that we introduced the sweeping function w with the only aim of separating the zeros we are interested in from the rest of the zeros. We emphasize that the function w is not related to any quadrature method we shall use to pinpoint the zeros, for example, by virtue of the principle of argument. Hence, the function w is inherently related the geometry of the zeros of f but has no direct relation to the evaluation process of these zeros. The function w is not just a substitutive transformation for integrals, but on the other hand, it is a general consequence of the requirements (a) and (b) that integrative numerical evaluation is more reliable using such a function w.

Having stated the main requirements and introduced the sweeping function, we should now give some simple examples prior to discussing the general algorithm. Suppose we are given the function $f(z) = (z-1)(z-(1+\epsilon))g(z)$ for small $\epsilon > 0$. Since we know that $f(z)$ has at least two zeros which are very close to each other, the numerical separation of the zeros of $f(z)$ would be a very instable process. Instead, we could define the sweeping function $w(z) = z/M$ with $M * \epsilon >> 1$ and solve the problem $f(w(z)) = 0$. The problem $f(w(z)) = 0$ is much more stable than the problem $f(z) = 0$. We could have used $w(z) = \log(z)$ in this example which would also lead to a more stable process. However, if the desired zeros are complex, the function $\log(z)$ is to be used with usual restrictions because of its discontinuity.

Sometimes we may want to use multiplicative sweeping functions. If we know that the function f has well-separated zeros in a large connected domain, it is worthwhile to down-size the domain for practical reasons of numerical

integrability. This can be done via sweeping functions of the form $w(z) = M * z$ or $w(z) = exp(z)$. However, we emphasize again that the decision about which sweeping function to use is related to the geometry of the zeros of f and needs thorough analysis in general. We cannot expect that this analysis be automated for a large variety of functions and/or domains. The case when $f(z)$ is a polynomial is of primary interest in applications. Our method is unique in handling polynomials in the sense that the algorithm's complexity is not dependent on the degree of the polynomial. This is best understood by means of the following important definition.

DEFINITION 3.1 *Let a polynomial $p(z)$ with degree n be given in its Taylor representation*

$$p(z) = \sum_{i=0}^{n} a_i x^i$$

We define the size of the $p(z)$ as follows:

$$s(p) = |\{a_i : a_i \neq 0\}|$$

Informally, $s(p)$ counts the number of terms in the $p(z)$. It is important to realize that the following simple fact always holds:

$$s(p) <= n + 1$$

Moreover we easily can develop polynomials with $s(p) << n$. Or in other words, cases where n is very large are never combined (for practical reasons) with a large $s(p)$. This is very plausible if you think that n may be $\Omega(2^k)$ for some parameter k. Because we somehow must mechanically (or manually) generate all terms of $p(z)$, we are forced to keep $s(p)$ in $O(q(k'))$ for some polynomial q and parameter k', even if n is of very high order, in order to be able to work properly with $p(z)$. Making an algorithm's complexity depend on $s(p)$ instead of n is thus a big advantage unless n is small. The reader should be aware of the fact that zero-finding algorithms for polynomials in general work in $O(q(n))$ where $q(n)$ is some polynomial in the degree n (of $p(z)$). Only the Descartes rule of signs is an exception in this regard is as much as it can decide about the existence of zeros in the given interval in $\Omega(s(p))$ steps instead of $O(q(n))$ steps. However, even in this case if the number of sign changes is even (and not zero) the performance is deteriorated by (needed) additional bisections.

The algorithm presented in this paper is based on the measure $s(p)$ instead of n if $f(z)$ is a polynomial. Thus, in principle, the algorithm is able to work with high-degree polynomials as easily as with low-degree polynomials. However, we still have the problem of the accuracy of produced results if n is very large, since in this case the polynomial may oscillate too much in the considered interval.

4. The Algorithm

The algorithm presented in this section will focus on the filtering of the *real zeros* of a given analytic function. We will tackle the difficult case of functions having dense complex zeros around the real axis. Also, as stated earlier we do not assume that the number of zeros is small. Thus, we will be in general interested in locating some of the real zeros or merely in deciding whether or not the function has at least one real zero in the a given (real) interval $[a, b]$.

The main idea of the algorithm is as follows. Suppose we want to locate some real zeros of f and we know that any non-real zero z has the property that is, $Im(z) \geq \epsilon$. (This property means that the non-real zeros are not closer than ϵ to the real axis). Then, we define the sweeping function $w(z) = z/M$ with $M \geq 1/\epsilon$. In this way, the function $f(w(z))$ will have the same real zeros of $f(z)$ scaled up by the factor M in the interval $[M * a, M * b]$. Moreover, the function $w(z)$ has the properties (a) and (b) mentioned in Section 3.

Before investigating the approximation of the number of zeros, we should clarify one issue related to the use of the algorithm as an automated routine. We will require from the algorithm to give us a good result concerning the real zeros and their number for any given accuracy. This shall be achieved by making the value of ϵ a parameter of the algorithm that the user is able to specify. In so doing, the algorithm is effectively able to return the correct result to any desired accuracy. Clearly, the parameter ϵ will be in general determined after some analysis of the function under study. However, in the worst case, if namely the geometry of the zeros of the function f is not reliably known, the user can still use a number close to (but greater than) the smallest representable positive machine number if high accuracy is important. In a symbolic environment, the user is able to choose ϵ many orders of magnitudes smaller than typical machine accuracies. This last method though theoretically advisable might induce high numerical instability in practice. Actually, a good value of ϵ is in general not easily found.

The preceding considerations about $f(z)$ and the desired accuracy ϵ already partially determine the input of our algorithm. Thus given a function and an accuracy ϵ, the first steps of the algorithm is to perform the transformations:

(1) $M := 1/\epsilon$;
(2) $w(z) := z/M$;
(3) $[a, b] := [a * M, b * M]$;
(4) $f(z) := f(w(z))$.

We now focus on how to approximate the number of the real zeros in the *new* interval $[a, b]$. We will make use of the principle of the argument (see Section 3) with regard to the new function $f(z)$, that is, instead of (1), we will be using the following formula:

$$N = \frac{1}{2\pi I} \oint_{\gamma} \frac{f'(w(z))(\frac{d}{dz}w(z))}{f(w(z))} dz \tag{3}$$

However, we have to determine first a closed contour around the interval $[a, b]$. For reasons of accuracy, a circular contour is the best choice. Periodicity of the integrand calls for the use of the trapezium rule as a quadrature rule for the approximation of the integral. This rule is extremely reliable and of course (as usual) very efficient in case of periodicity. The reason lies in the well-known Euler-Maclaurin expansion of the trapezium rule. Unfortunately, we would be faced with a problem if we use a circular contour. The problem is best explained using Figure 1. We notice that after the above 4 steps of the algorithm have been used, the real zeros of $f(z)$ will move horizontally by the factor M. At the same time, the non-real zeros move horizontally as well as vertically again by the factor M. If we choose now a circular contour that includes all potential real zeros of $f(z)$, we cannot make sure that all non-real zeros are outside the domain within the contour. Hence, this solution is not feasible unless the factor M is not too large (that is non-real zeros are not too close to the real axis) in which case we may be able to decompose the contour into smaller (but not too many) adjacent circles and perform the numerical quadrature on each of them. However, this would contradict the main assumption of our problem, namely, that M is large in general.

Instead of a circular contour we will be using a rectangular one. A rectangular contour, though its numerical quadrature treatment is cumbersome and often less reliable, is the only halfway acceptable contour in our case. With careful shaping we are able to use a rectangular contour to achieve a desired accuracy. As indicated in Figure 2, the rectangular contour can be shaped so that:

(a) No potential real zeros of $f(z)$ lie outside the contour.

(b) No potential non-real zeros of $f(z)$ lie inside the contour.

Properties (a) and (b) clearly explain why we spoke of "filtering" the desired zeros by means of a sweeping function earlier in this paper. We can achieve the desired properties (a) and (b) by choosing B (see Fig. 2) in such a way that it is large enough as to avoid integration near the real axis and small enough as to avoid non-real singularities of the logarithmic derivative present under the integral sign in formula (3).

The next issue is to determine a good value of B. Obviously B must not be very small since then the contour would be close to potential real-valued singularities of the logarithmic derivative. Our approach for choosing B is preventive. When we fix the value of M, we tend to be very restrictive. That is, when the user specifies $\epsilon = 1/M$, what the algorithm will actually do is to set:

$$M := M + B \tag{4}$$

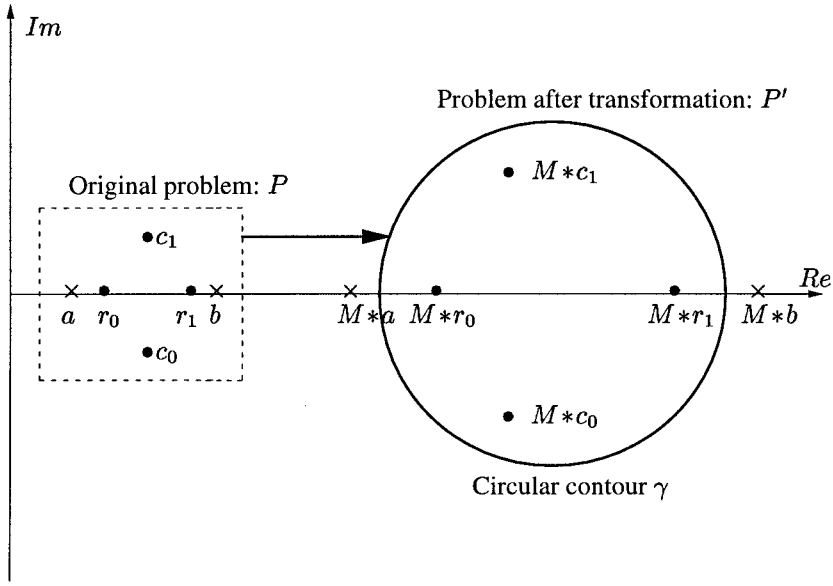

Figure 1. After transforming the problem via $w(z)$, some non-real zeros (here $M * c_0$ and $M * c_1$) may unavoidably remain in any circular contour containing all zeros in [M*a, M*b] (here $M * r_0$ and $M * r_1$).

The algorithm will therefore effectively work with $M + B$ instead of M. This has the nice side effect that we are now able to choose B freely from within the algorithm. You can, therefore, think of B as a configuration parameter of the algorithm itself. In any instance, B should be chosen relatively large compared to 1. In our experiments, the value of B was chosen to be in the range [1, 100]. However, larger values are acceptable too and may lead to better results when used. This method for the determination of B will not work if the user chooses ϵ to be very close to the smallest representable positive number in the machine. This case is deemed unusual in this treatment and should be avoided/prevented for reasons of numerical stability.

We still did not specify the quadrature rule to be used. We already mentioned that the trapezium rule is unsurpassed for circular contours. In our case, though the contour is rectangular, we decided to use the trapezium rule for reasons of efficiency. As should any automatic quadrature procedure, our quadrature algorithm is adaptive. Adaptivity is needed since we cannot expect the user to know a priori a minimum number of function evaluations needed to achieve the desired accuracy. Hence, we use an adaptive trapezium quadrature, with accuracy requirement *eps* in the range [0.1, 0.4]. This is sufficient to distinguish two consecutive integers. It is worth mentioning in this context that

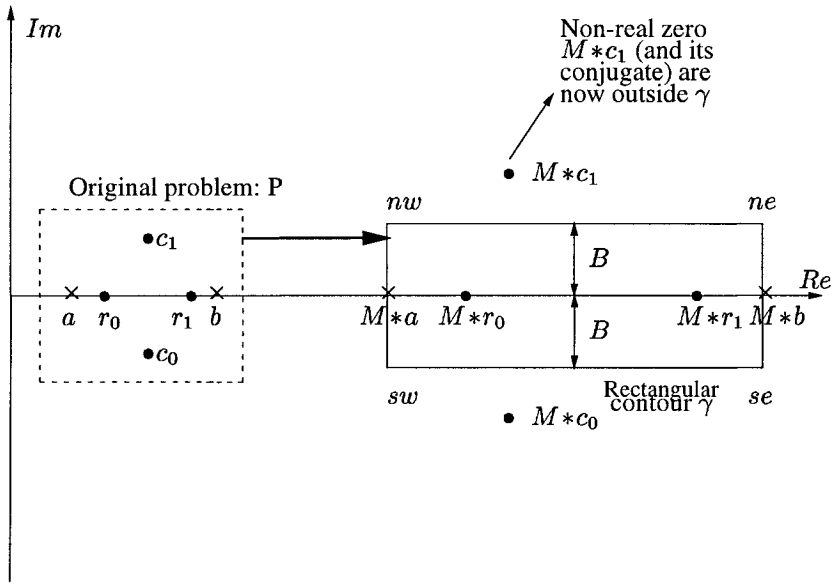

Figure 2. The rectangle contour γ with length $M * (b - a)$, width $2 * B$, and edges (sw, se, ne, and nw) encloses now only the real zeros of $f(z)$ scaled up by the factor M.

there is no need to let the user specify *eps*, since the algorithm is intended to approximate an integer-valued integral and not to be used as a general-purpose adaptive quadrature method. As with any adaptive method, the quadrature procedure relies on a subdivision of the treated interval that depends on the characteristics of the integrand in different sections of the interval. The implemented algorithm is therefore recursive. The adaptive trapezium quadrature procedure stops if either of the following conditions occurs:
(a) The required accuracy is reached in the respective sub-interval.
(b) The size of the treated subinterval (which is halved in each recursive call) is less than a pre-assigned threshold.

The threshold mentioned above is also configuration parameter of the algorithm and should be set as small as possible. It is also possible to consider this threshold as an input parameter of the algorithm, in which case the user is responsible for its determination. Obviously, only case (a) of above would lead to the desired result. However, in practice useful results can also be achieved in spite of the termination s enforcing criterion (b), if the threshold is chosen carefully.

Now we address one of the most crucial issues in the evaluation process of the integral. This issue deals with the problem that the integration interval

$[a, b]$ is per construction very large. It goes without saying that the error in the evaluation of the integral (see (2)) is among others super-proportional to the size of the interval[2]. In the adaptive version of the trapezium rule, this dependency will manifest itself in an almost non-terminating computation; a highly undesirable situation. To avoid this problem, we resort to the classical substitution technique. There are many ways to perform an interval-shrinking substitution and we decided to use the logarithm function. Thus, the integral in (3) is further transformed to:

$$N = \frac{1}{2\pi I} \oint_{\ln(\gamma)} \frac{f'(w(e^z))w'(e^z)e^z}{f(w(e^z))} dz \tag{5}$$

Last integral is more precisely the sum of four integrals along the sides of the rectangle[3]:

$$N_i = \frac{1}{2\pi I} \oint_{\ln(z_i)}^{\ln(z_{i+1})} \frac{f'(w(e^z))w'(e^z)e^z}{f(w(e^z))} dz \tag{6}$$

where $(z_i, z_{i+1}) \in \{(sw, se), (se, ne), (ne, nw), (nw, sw)\}, i \in \{1, 2, 3, 4\}$).

Thus, N of (5) is then (clearly, the $N_i's$ need not to be integers or even real):

$$N = N_1 + N_2 + N_3 + N_4$$

It is pertinent to point out here that this last integral transformation of equation (6) is a typical one and is different from the transformation made in (3). There, we needed to know the geometry of zeros in order to find a convenient function $w(z)$. However, the transformation here does not need any information about the integrand's zeros. In performing the transformation (6) we have to make sure that we use the correct branch of the logarithm for the different four integrals. However, for different integrals different branches can be used. The price paid in issuing the transformation in (6) is:

(1) The integrand is now more complex increasing the time for function evaluations.

(2) The integrand might lose some desirable properties after the transformation (less smooth, non-periodic, etc.).

(3) The second derivative of the integrand may highly increase in absolute values contra-acting the shrinkage of the integration interval in the error term.

[2]Precisely, the error of the $m-$point trapezium rule is $-((b-a)^3/12m^2)f''(\xi)$ for some ξ in (a, b).

[3]Thus the following quadrature discussion equally apply to all four integrals.

Using (6) the algorithm is able to unambiguously estimate the number of the zeros $N(a,b)$ in the interval $[a,b]$. If this number is zero, the algorithm terminates. If $N(a,b)$ is small, say $N(a,b) \leq 20$, the algorithm is reused recursively for the subintervals $[a,(a+b)/2]$ or $[(a+b)/2,b]$ depending on which subinterval includes more zeros (counting multiplicities). This bisection is performed as long as $N(a,b)$ is different from zero. In order to determine a zero with a specified accuracy ϵ_0, the algorithm uses this bisection method until $b-a \leq \epsilon_0$. Here, ϵ_0 is a user-supplied input of the algorithm. In this way, if the original interval [a, b] only includes a small number of zeros, those zeros are determined to any desired accuracy. The complexity of the bisection scheme for an interval [a, b] is $O(log(b-a))$. As mentioned in the Section 3, one of the strengths of the algorithm is that it can deal with functions having a large number of zeros as easily as with those with small number of zeros.

Let us shed some light on the performance and the numerical reliability of the algorithm. The numerical reliability of the algorithm is only limited by the convenience of the quadrature rule. Taking into consideration that the number we are seeking is an integer, we see that even a crude approximation is highly appreciated. There is an "apparently" good way of enhancing the numerical reliability of the quadrature process by using $f^2(z)$ instead of $f(z)$, thus halving the accuracy demands, since the new zeros are double zeros, and we only need to know $N(a,b)$ to 0.9 accuracy (say), since $N(a,b)$ is known to be an even number in this case. Unfortunately, this will not work, since formula (6) includes the logarithmic derivative making any exponentiation behave like a mere multiplication by a constant.

The decision for using the trapezium rule has also to do with performance. It is well known that in the trapezium rule pre-computed functions values need not be computed again if the integration interval is further refined. Gauss quadrature is extremely time consuming in this respect, since already computed function values are not reused. Moreover, if the function $f(z)$ is periodic and the size of the interval $[a,b]$ is a multiple of its period, we know that the trapezium rule is extremely reliable. Since the algorithm s quadrature process is adaptive, the number of needed function evaluations will in general depend on the function $f(z)$. The stopping criterion (b) (artificially) integrated in this process is only a mean to limit this number even if some accuracy is lost. The bisection procedure, however, is extremely fast ($O(log(b-a))$. Thus pinpointing a zero is done in logarithmic time (for example, versus binary search).

In calculating the values of the four integrals in (6), we are occasionally able to make the following optimizations:

1 If the function $f(z)$ is real on the real axis, it can be easily shown that the zeros of $f(z)$ will be either real or conjugate pairs. It is therefore needless to evaluate both integrals N_1 and N_3 of formula (6). We only

need to compute one integral, since their values will be conjugate:

$$N_1 + N_3 = 2 * Re(N_1) = 2 * Re(N_3)$$

2 If $f(z)$ is periodic and the size of the interval $[a, b]$ is a multiple of its period, there is no need to compute the integrals N_2 and N_4 at all, since their values will cancel. Thus, we see that periodicity of the integrand both lowers the reliability demands and enhances performance (a rare combination).

3 If $f(z)$ satisfies both above conditions, we merely need to compute one integral in formula (6), namely, N_1 or N_3.

Another minor optimization that would enhance performance is to only evaluate the real parts of the integrals in (6), that is:

$$N(a, b) = Re(N_1) + Re(N_2) + Re(N_3) + Re(N_4)$$

5. Examples and Experimental Results

We implemented the described algorithm in Maple 8.01. The actual implementation is straightforward in the sense that we did not take any precaution related to round-off errors. The results are promising though. Below, we give an account of some experiments with the implemented algorithm.

We have selected three types of functions: polynomials, trigonometric functions, and other more involved functions containing transcendental ones. If applicable, we indicate the source of the selected function. Also, in each case we indicate the respective interval $[a, b]$ and the values of the parameters M and B. These values are important since they reflect the needed accuracies of the user.

Polynomials

$$f_1(z)(z-4)(z-5)^3(z-4-\frac{1}{1000}I)(z-4+\frac{1}{1000}I)$$

This polynomial has 4 real zeros (counting multiplicities) and two conjugate complex zeros:

$$z_0 = 4,\ z_1 = 5,\ z_3 = 4 + \frac{1}{1000}I, \overline{z_3}$$

Notice that the conjugate pair is very close to the real zero z_0.

Applying the algorithm with $[a,b] = [3, 3+2\pi], M = 10000, B = 5$. We obtained the result of $4.0000416415776037389 + 0.I$. This is the expected result with enough accuracy.

We need to mention here that in the actual implementation it was not possible to work with complex zeros much closer than 10^{-3} to the real axis. This problem needs further investigation as pointed out in the conclusion.

The next polynomial is an extremely ill-conditioned one of (Delves and Lyness, 1967(a)):

$$\begin{aligned} f_2(z) \;=\; & 1250162561\, z^{16} + 385455882\, z^{15} + 845947696\, z^{14} \\ & +240775148\, z^{13} + 247926664\, z^{12} + 41018752\, z^{10} \\ & +9490840\, z^9 + 4178260\, z^8 + 837860\, z^7 + 267232\, z^6 \\ & +44184\, z^5 + 10416\, z^4 + 1288\, z^3 + 224\, z^2 + 16\, z + 2 \end{aligned}$$

Using the same parameters as in the preceding example, the algorithm gives the result of $-0.13505417 + 106{-}5 + 0.I$. This is also the expected result, since this polynomial does not have any real zero.

The result of the next polynomial shows one of the strongest features of the algorithm. The polynomial has a very large degree and we anticipate knowing the number of its real zeros in the mentioned interval. The polynomial is:

$$f_3(z) = (z-4)(z-5)^3(z-4-\frac{1}{10}I)^{10000}(z-4+\frac{1}{10}I)^{100000}$$

We used the algorithm for this polynomial with $M = 20, B = 1$. The result obtained is $3.992583 - 0.I$. This is an extremely good result, since out of 200004 zeros of $f_3(z)$ only 4 (counting multiplicities) are real. Notice that the 200000 complex zeros are close to real axis.

Trigonometric Functions

It is often beneficial to make the use of the periodicity of this type of functions. The following function is a simple trigonometric polynomial:

$$f_4(z) = cos(5z) - 2cos(2z) + cos(z) + 3$$

We obtained the correct result (*i.e.,* 2) with our algorithm:

$$2.001582398 + 0.636619772210^{-16} I$$

We used here:

$$[a,b] = [1, 2\pi+1], M = 100000, B = 100$$

The next function is based on the polynomial $f_2(z)$

$$f_5(z) = f_2(cos(z)+1) - 4$$

Maple shows that this (periodic) function has two simple real zeros (close to 3) in the interval $[0, 2\pi]$. Our algorithm's result confirms that:

$$1.997902195 + 0.359690171310^{-15}I$$

We used here the same values of a, b, M, and B as in the preceding example.

Other Functions

The next function is a mixed sine/exponential function.

$$f_6(z) = e^{(\frac{z}{4}-1)} + 10sin(\frac{z}{2} - 5)$$

Applying our algorithm with same the parameters as in the preceding example, we obtained the result of $1.006411448 - 0.186211283410^{-16}I$. This confirms the Maple plot function, which shows a single real root in the corresponding region.

The next function is used in practice to model heat propagation (Greenleaf, 1972) and one is interested in general in some of its (many) real zeros. Here, we used the instance:

$$f_7(z) = tan(z) - z$$

With $B = 20$ and without changing the other parameters, we obtained the result of $4.415547706 + 0.111408460110^{-17}I$. This is still an acceptable result for the true value 4.

The final example is based on the Bessel function. The function is often used in physical applications to model wave reflection (Kravanja, Barel and Van, 2000).

$$f_8(n, z) = J_n(z - 2\pi) + IJ_{n+1}(z - 2\pi)$$

It is known that if $n > 0$, $f_8(n, z)$ admits only one real zero multiplicity n at 2π with all remaining zeros non-real. We anticipate verifying this theoretical result using our algorithm for n in the range [1, 10]. The following table lists the results for the treated values of n.

N	Obtained Result
1	$1.321014932 + 0.0002922914206I$
2	$2.253817259 + 0.0002022027040I$
3	$3.208294409 + 0.0001329376255I$
4	$4.175709689 + 0.0000884997349I$
5	$5.151328999 + 0.0000605998921I$
6	$6.132658423 + 0.0000427817031I$
7	$7.117892159 + 0.0000310835358I$
8	$8.105992370 + 0.0000231751786I$
9	$9.096218075 + 0.0000176776976I$
10	$10.08803592 + 0.0000137573736I$

The algorithm's parameters in all these experiments were set as follows:

$$[a, b] = [2, 2\pi + 2], \quad M = 1000, \quad B = 10$$

For all values of n in [1, 10] we have acceptable results. One can also easily see that the experiments with the values 100 and 1000 for n and we obtained respectively:

$$100.0099893 + 0.2195117\ 10^{-7}I$$
$$1000.001071 + 0.36733\ 10^{-10}I$$

The sweeping function w that has been presented separates the set of zeros that the user is interested in from the remaining set of zeros. In fact, w is a filter as we assume the number of zeros is large. The algorithm, which is based on finding a rectangular contour in the interval $[a, b]$ is amenable to parallelism and can be implemented on a parallel computer. At each step, the rectangular contour is subdivided into sub-rectangles. The sub-rectangles being found to contain zeros are further subdivided. If a sub-rectangle does not contain a zero it is discarded. If we continue this subdivision process we will likely pinpoint the real zeroes. Each sub-rectangle generates a set of new sub-rectangles. Each sub-rectangle being considered for the next iteration should have the following properties: (a) no potential real zeros are outside the contour, (b) no potential non-real zeros are inside the contour. In addition to the configuration parameter B and the factor M, we can add other parameters (*i.e.,* efficiency, speedup, accuracy, reliability) that may influence the results positively. The number of processors available increases the number of subdivisions, however, the algorithm may not give the efficiency that would be expected. Our initial analysis indicates that it is likely possible to speed up the sequential algorithm that has been presented. This work is being investigated under these lines and will be our forthcoming research paper.

6. Conclusion

The main achievements of this work are:

(a) Addressing the problem of filtering interesting zeros of a given analytic function. (b) Introducing the concept of a sweeping function as a tool for such filtering. (c) Applying the concept to the selection of real zeros of an analytic function. (d) Constructively demonstrating the usefulness of the introduced concepts by means of an algorithm that proved to be able to handle a large type of analytic functions. (e) Assuring that the algorithm is able to handle extreme cases like polynomials with high degrees, analytic functions with large number of zeros, and zeros that are very close together.

There is no doubt that the described algorithm even in its current form will provide analysts with much-needed insights for their functions. However, in

the design of the algorithm we left one important item unanswered, namely, how can the user/analyst be guided in the process of finding good values of the prescribed accuracy (*i.e.*, M). This question is still open. Another more important question in this context is to study the influence of the parameters M and B (and other figuration parameters) on the accuracy of the algorithm's results. This may lead to dependencies on the used quadrature rule. It may outcome that a Gauss-like quadrature rule is needed to enhance independency of the algorithm's accuracy on the parameters M and B. All in all, we need more analysis and experiments in this respect and we need to know more about the overall conditioning of the algorithm with respect to its parameters.

We also introduced the concept of sweeping functions. We, however, did not provide any hint selecting a good such function. This problem is also open for the time being and needs more clarification. A related problem is how to find sweeping function for general domains in the complex plane. Recall that the present paper only dealt with domains of the real axis (real intervals).

Acknowledgments

We would like to thank the referees for their valuable comments that helped improve the paper.

References

Atanassova L. On the Simultaneous Determination of the Zeros of Analytic an Function Inside a Simple Smooth Closed Contour in the Complex Plane. *J. Comput. Appl. Math.*, 50:99–107, 1994.

Delves L. M., Lyness J. N. A Numerical Method for Locating the Zeros of Analytic Functions. *Math. of Comput.*, 21(100):543–560, 1967.

Delves L. M., Lyness J. N. On Numerical Contour Integration Round a Closed Contour. *Mathematics of Comput.*, 21(100):561–577, 1967.

Douglas J., Burden R. Numerical Methods. Thomson Books/Cole, 2003.

Drucker D. S. A Second Look at Descartes Rule of Signs. *Mathematics Magazine*, 52: 237–238, 1979.

Foster L. V. Generalizations of Laguerre's Method: Higher Order Methods. *SIAM Journal on Num. Analysis*, 18(6):1004–1018, 1981.

Greenleaf F. P. Complex Variables. W. B. Saunders Company, 1972.

Henrici P. Applied and Computation Complex Analysis. *Power Series- Integration-Conformal Mapping-Location of Zeros, Vol 1.* Wiley, 1974.

Herlocker J., Ely J. An Automated and Guaranteed Determination of the Number of Roots of Analytic Functions Interior to a Simple Closed Curve in the Complex Plane, *Reliable Comput.*, 3:239–250, 1995.

Kravanja, P., Barel, M. Van. Computing the Zeros of Analytic Functions, Series. *Lecture Notes in Mathematics, VII*, 2000.

Lehmer D. H. A Machine Method for Solving Polynomial Equations. *J. Assoc. Comput. Mach.*, 8:151–162, 1962.

Li T. On Locating All Zeros of an Analytic Function within a Bounded Domain by a Revised Delves/Lyness Method. *SIAM Journal on Numerical Analysis*, 20(4):865–871, 1983.

Olver H. W. The Evaluation of Zeros of High-Degree Polynomials. *Ph. Trans. of the Royal Soc. of London, (A)*, 244(885):385–415, 1952.

Petcovic M. S. et. al. Weierstrass Formula and Zero-Finding Methods. em Numerische Mathematik, 29: 353–372, 1995.

Petcovic M. S., Marjanovic Z. M. A Class of Simultaneous Methods for the Zeros of Analytic Functions. *Comput Appl. Math.*, 22(10): 79–87, 1991.

Vincent A. J. H. Sur la Resolution des Equations Numeriques. *Journal de Mathematiques Pures et Appliquees*, 341–371.

Wilf H. S. A Global Bisection Scheme for Computing the Zeros of Polynomials in the Complex Plane. *Journal of the Association for Computing Machinery*, 25(3): 415–420, 1978.

Ying X., Katz I. N. A Reliable Argument Principle Algorithm to Find the Number of Zeros of an Analytic Function in a Bounded Domain. *Numerische Mathematik*, 53:143–163, 1988.