Computer Physics Communications 124 (2000) 212–232

Computer Physics Communications

www.elsevier.nl/locate/cpc

# ZEAL: A mathematical software package for computing zeros of analytic functions

P. Kravanja [a,1], M. Van Barel [a,2], O. Ragos [b,3], M.N. Vrahatis [b,4], F.A. Zafiropoulos [b,5]

[a] *Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200 A, B-3001 Heverlee, Belgium*
[b] *Department of Mathematics and University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR-261.10 Patras, Greece*

Received 3 December 1998; accepted 28 June 1999

**Abstract**

We present a reliable and portable software package for computing zeros of analytic functions. The package is named ZEAL (ZEros of AnaLytic functions). Given a rectangular region $W$ in the complex plane and a function $f : W \to \mathbb{C}$ that is analytic in $W$ and does not have zeros on the boundary of $W$, ZEAL localizes and computes *all* the zeros of $f$ that lie inside $W$, together with their respective multiplicities. ZEAL is based on the theory of formal orthogonal polynomials. It proceeds by evaluating numerically certain integrals along the boundary of $W$ involving the logarithmic derivative $f'/f$ and by solving generalized eigenvalue problems. The multiplicities are computed by solving a linear system of equations that has Vandermonde structure. ZEAL is written in Fortran 90. 

*PACS:* 02.30.Dk; 02.60.Cb
*AMS classification*: 65H05
*Keywords:* Analytic functions; Zeros; Multiplicities; Quadrature method; Formal orthogonal polynomials; Isolation of zeros; Computation of zeros

## PROGRAM SUMMARY

*Title of program:* ZEAL

*Catalogue identifier:* ADKW

*Program Summary URL:*
http://cpc.cs.qub.ac.uk/summaries/ADKW

*Program obtainable from:* CPC Program Library, Queen's University of Belfast, N. Ireland

*Computers:* Hewlett-Packard 9000 B160L, IBM RS6000 7012, Sun Microsystems SPARC Ultra-2 m1170 and Silicon Graphics Origin 2000

*Operating systems under which the program has been tested:* UNIX (HPUX 11.0, AIX 3.2.5, SunOS 5.5.1 and IRIX64 6.4)

[1] E-mail: Peter.Kravanja@na-net.ornl.gov
[2] E-mail: Marc.VanBarel@cs.kuleuven.ac.be
[3] E-mail: ragos@math.upatras.gr
[4] E-mail: vrahatis@math.upatras.gr
[5] E-mail: phikapa@math.upatras.gr

0010-4655/00/$ – see front matter 
PII: S0010-4655(99)00429-4

*Programming language used:* Fortran 90

*Memory required to execute with typical data:* Less than 500 Kbytes

*No. of bits in a word:* 32 bits

*No. of processors used:* One

*Has the code been vectorised?:* No

*No. of bytes in distributed program, including test data, etc.:* 252 832

*Distribution format:* uuencoded compressed tar file

*Keywords:* Analytic functions, zeros, multiplicities, quadrature method, formal orthogonal polynomials, isolation of zeros, computation of zeros

*Nature of physical problem*
ZEAL is a general purpose package for computing zeros of analytic functions. It can be used in various physical applications. More precisely, given a rectangular region $W$ in the complex plane and an analytic function $f : W \to \mathbb{C}$, such that no zero of $f$ lies on the boundary of $W$, ZEAL calculates all the zeros of $f$ that lie inside $W$, together with their respective multiplicities.

*Method of solution*
The package ZEAL uses an integral formula to compute the total number of zeros (counting multiplicities) of $f$ that lie inside $W$. Then, by using the same procedure, the region $W$ is subdivided into subregions that contain at most $M$ zeros (again counting multiplicities), where the value of $M$ is specified by the user. Approximations for these zeros are calculated via an algorithm that is based on numerical integration along the boundaries of the subregions and generalized eigenvalue problems. The multiplicities of the zeros are calculated by solving a Vandermonde system. The approximations for the zeros are refined via the modified Newton's method, which takes into account the multiplicity of a zero and converges quadratically.

*Restrictions on the complexity of the problem*
The function $f$ has to be analytic in the rectangular region $W$. Both $f$ and its derivative $f'$ are needed. The edges of $W$ have to be parallel to the coordinate axes. The boundary of $W$ is not allowed to contain zeros of $f$. Since, in the sequel, $W$ is repeatedly subdivided, the boundaries of the obtained subregions should not contain zeros of $f$. The possibility of such a situation can be minimized if $W$ is not symmetric with respect to the axes. ZEAL is not specifically designed to handle clusters of zeros. However, if $f$ has one or more clusters of zeros and the input parameter `EPS_STOP` is given a proper (problem-dependent) value, then ZEAL will compute approximations for the centres of the clusters. The "multiplicity" of a centre is equal to the total number of zeros that belong to the corresponding cluster.

*Typical running time*
The following table gives the running times (in seconds) for the test runs of Section 4:

| | System library | Code included in ZEAL |
|---|---|---|
| Test run # 1 | 0.13 (0.13) | 0.18 (0.14) |
| Test run # 2 | 1.82 (1.04) | 1.65 (1.14) |
| Test run # 3 | 1.55 (1.14) | 1.60 (1.42) |
| Test run # 4 | 1.52 (1.15) | 1.62 (1.35) |
| Test run # 5 | 0.65 (0.41) | 1.62 (0.37) |
| Test run # 6 | 1.56 (0.56) | 1.54 (0.53) |

The calculations have been done on a SUN SPARC Ultra-2 m1170. We have used the subroutine ETIME. The parenthesized running times correspond to optimized compiling.

ZEAL uses a number of routines from the BLAS and LAPACK libraries. These Fortran 77 routines are distributed together with ZEAL to enable the user to compile them in case the BLAS and LAPACK libraries are not available on his/her computer system. The column labelled "System library" gives the running times in case ZEAL uses the BLAS and LAPACK libraries that are installed on our SUN computer. The column labelled "Code included in ZEAL" gives the running times in case the Fortran 77 routines distributed with ZEAL are used.

# LONG WRITE-UP

## 1. Introduction

Let $W$ be a rectangular region in $\mathbb{C}$, $f : W \to \mathbb{C}$ analytic in the closure of $W$ and $\gamma$ the (positively oriented) boundary of $W$. Suppose that $\gamma$ does not pass through any zero of $f$ and that the edges of $\gamma$ are parallel to the coordinate axes. We present a reliable and portable software package for computing *all* the zeros of $f$ that lie in

the interior of $\gamma$, together with their respective multiplicities [6]. Our package is named ZEAL (ZEros of AnaLytic functions) and is written in Fortran 90.

Our approach to the problem of computing all the zeros of an analytic function that lie in the interior of a Jordan curve can be seen as a continuation of the pioneering work of Delves and Lyness [1] and the corresponding Fortran 77 implementation written by Botten, Craig and McPhedran [2].

Let $N$ denote the total number of zeros of $f$ that lie in the interior of $\gamma$, i.e., the number of zeros where each zero is counted according to its multiplicity. Suppose from now on that $N > 0$. Delves and Lyness considered the sequence $Z_1, \ldots, Z_N$ that consists of all the zeros of $f$ that lie inside $\gamma$. Each zero is repeated according to its multiplicity. An easy calculation shows that the logarithmic derivative $f'/f$ has a simple pole at each zero of $f$ with residue equal to the multiplicity of the zero. Cauchy's Theorem implies that

$$N = \frac{1}{2\pi i} \int_\gamma \frac{f'(z)}{f(z)}\, dz. \tag{1}$$

This formula enables us to calculate $N$ via numerical integration. Methods for the determination of zeros of analytic functions that are based on the numerical evaluation of integrals are called *quadrature methods*. A review of such methods is given by Ioakimidis [3]. Delves and Lyness considered the integrals

$$s_p := \frac{1}{2\pi i} \int_\gamma z^p \frac{f'(z)}{f(z)}\, dz, \quad p = 0, 1, 2, \ldots.$$

The residue theorem implies that the $s_p$'s are equal to the *Newton sums* of the unknown zeros,

$$s_p = Z_1^p + \cdots + Z_N^p, \quad p = 0, 1, 2, \ldots. \tag{2}$$

These $s_p$'s can again be calculated via numerical integration along $\gamma$.

Delves and Lyness considered the monic polynomial of degree $N$ that has zeros $Z_1, \ldots, Z_N$,

$$P_N(z) := \prod_{k=1}^{N} (z - Z_k) =: z^N + \sigma_1 z^{N-1} + \cdots + \sigma_N.$$

They called $P_N(z)$ the *associated polynomial* for the interior of $\gamma$. Its coefficients can be calculated via Newton's identities. An elegant proof is given by Carpentier and Dos Santos [4].

**Theorem 1** (Newton's identities).

$$\begin{aligned} &s_1 + \sigma_1 = 0, \\ &s_2 + s_1\sigma_1 + 2\sigma_2 = 0, \\ &\qquad\vdots \\ &s_N + s_{N-1}\sigma_1 + \cdots + s_1\sigma_{N-1} + N\sigma_N = 0. \end{aligned}$$

In this way they reduced the problem to the easier problem of computing the zeros of a polynomial. Unfortunately, the map from the Newton sums $s_1, \ldots, s_N$ to the coefficients $\sigma_1, \ldots, \sigma_N$ is usually ill-conditioned. Also, the polynomials that arise in practice may be such that small changes in the coefficients produce much larger

[6] The assumptions that $W$ is a rectangular region in the complex plane and that the edges of its boundary $\gamma$ are parallel to the coordinate axes are of course not essential from a theoretical point of view. They merely represent the specific choice that we have made while developing our package. In fact, the algorithm for computing all the zeros of $f$ that lie in the interior of $\gamma$ that we will discuss in Section 2 can be used for an arbitrary simply connected region $W$ and an arbitrary positively oriented Jordan curve $\gamma$.

changes in some of the zeros. This ill-conditioning of the map between the coefficients of a polynomial and its zeros has been investigated by Wilkinson [5]. The location of the zeros determines their sensitivity to perturbations of the coefficients. Multiple zeros and very close zeros are extremely sensitive, but even a group of moderately close zeros can result in severe ill-conditioning. Wilkinson states that ill-conditioning in polynomials cannot be overcome without, at some stage of the computation, resorting to high precision arithmetic.

If $f$ has many zeros in the interior of $\gamma$, then the associated polynomial is of high degree and could be very ill-conditioned. Therefore, if $N$ is large, one has to calculate the coefficients $\sigma_1, \ldots, \sigma_N$, and thus the integrals $s_1, \ldots, s_N$, very accurately. To avoid the use of high precision arithmetic and to reduce the number of integrand evaluations needed to approximate the $s_p$'s, Delves and Lyness suggested to construct and solve the associated polynomial only if its degree is smaller than or equal to a preassigned number $M$. Otherwise, the interior of $\gamma$ is subdivided or covered with a finite covering and the smaller regions are treated in turn. The choice of $M$ involves a trade-off. If $M$ is increased, then fewer regions have to be scanned. However, if $M$ is chosen too large, then the resulting associated polynomial may be ill-conditioned. Delves and Lyness chose $M = 5$.

Instead of using Newton's identities to construct the associated polynomial, Li [6] considered (2) as a system of polynomial equations. He used a homotopy continuation method to solve this system.

In this contribution we consider the mutually distinct zeros and their respective multiplicities *separately*. This is the approach that has recently been taken by Kravanja et al. [7]. Their quadrature method is a generalization of the method of Delves and Lyness. It is again based on the numerical evaluation of integrals along $\gamma$ that involve the logarithmic derivative $f'/f$, but by using the theory of formal orthogonal polynomials they have been able to obtain more accurate approximations for the zeros. Therefore, one may give the parameter $M$, i.e., the number of zeros that are calculated simultaneously, a larger value. Moreover, the algorithm of [7] does not require initial approximations for the zeros – it is self-starting – and it provides not only accurate approximations for the zeros but also the values of the corresponding multiplicities. Our Fortran 90 implementation of this algorithm is at the heart of ZEAL. Numerical approximations for the integrals are computed via the well-known quadrature package QUADPACK [8].

ZEAL's user interface is inspired by that of the Fortran 77 package ZEBEC, which is a package for computing simple zeros of Bessel functions that has been recently written by Kravanja et al. [9]. Once the user has specified the rectangular region $W$, the analytic function $f$ and its derivative $f'$, and the value of the parameter $M$, he/she can ask ZEAL to compute only the total number of zeros of $f$ that lie inside $W$, to isolate subregions of $W$ that contain at most $M$ zeros, to compute all the zeros of $f$ that lie inside $W$ or to compute only a specified number of zeros (together with their respective multiplicities). The results can be written on separate files. All these options will be discussed in detail below.

This paper is organized as follows. In Section 2 we give an overview of the algorithm for computing zeros of analytic functions that has been proposed by Kravanja et al. [7]. In Section 3 we discuss the structure and the user interface of our package ZEAL. Numerical examples are presented in Section 4.

## 2. Computing zeros of analytic functions

Let $n$ denote the number of mutually distinct zeros of $f$ that lie inside $\gamma$. Let $z_1, \ldots, z_n$ be these zeros and $\nu_1, \ldots, \nu_n$ their respective multiplicities. The quadrature method that Kravanja et al. [7] have recently proposed generalizes the approach of Delves and Lyness. Our implementation of their algorithm forms the central part of ZEAL. By using the theory of formal orthogonal polynomials, they have shown how the mutually distinct zeros can be calculated by solving generalized eigenvalue problems. The value of $n$ is determined indirectly. Once $n$ and $z_1, \ldots, z_n$ have been found, the problem becomes linear and the multiplicities $\nu_1, \ldots, \nu_n$ are computed by solving a linear system of equations that has Vandermonde structure. In this section we will give a brief summary of these results. For more details (including proofs and a pseudo-code formulation of the algorithm), we refer to [7].

Let $\mathcal{P}$ be the linear space of polynomials with complex coefficients. One defines a symmetric bilinear form

$$\langle \cdot, \cdot \rangle : \mathcal{P} \times \mathcal{P} \to \mathbb{C}$$

by setting

$$\langle \phi, \psi \rangle := \frac{1}{2\pi i} \int_{\gamma} \phi(z)\psi(z)\frac{f'(z)}{f(z)}\,dz = \sum_{k=1}^{n} \nu_k \phi(z_k)\psi(z_k) \tag{3}$$

for any two polynomials $\phi, \psi \in \mathcal{P}$. The latter equality follows from the fact that $f'/f$ has a simple pole at $z_k$ with residue $\nu_k$ for $k = 1, \ldots, n$. Note that $\langle \cdot, \cdot \rangle$ can be evaluated via numerical integration along $\gamma$. In what follows, we will assume that all the "inner products" $\langle \phi, \psi \rangle$ that are needed have been calculated. Let $s_p := \langle 1, z^p \rangle$ for $p = 0, 1, 2, \ldots$. These ordinary moments are equal to the *Newton sums* of the unknown zeros,

$$s_p = \sum_{k=1}^{n} \nu_k z_k^p, \quad p = 0, 1, 2, \ldots.$$

In particular, $s_0 = \nu_1 + \cdots + \nu_n = N$, the total number of zeros. Hence, we may assume that the value of $N$ is known. Let $H_k$ be the $k \times k$ Hankel matrix

$$H_k := \left[s_{p+q}\right]_{p,q=0}^{k-1} = \begin{bmatrix} s_0 & s_1 & \cdots & s_{k-1} \\ s_1 & & \iddots & \vdots \\ \vdots & \iddots & & \vdots \\ s_{k-1} & \cdots & \cdots & s_{2k-2} \end{bmatrix}$$

for $k = 1, 2, \ldots$. A monic polynomial $\varphi_t$ of degree $t \geqslant 0$ that satisfies

$$\left\langle z^k, \varphi_t(z) \right\rangle = 0, \quad k = 0, 1, \ldots, t-1, \tag{4}$$

is called a *formal orthogonal polynomial* (FOP) of degree $t$. (Observe that condition (4) is void for $t = 0$.) The adjective *formal* emphasizes the fact that, in general, the form $\langle \cdot, \cdot \rangle$ does not define a true inner product. An important consequence of this fact is that, in contrast to polynomials that are orthogonal with respect to a true inner product, formal orthogonal polynomials need not exist or need not be unique for every degree. (For details, see Draux [10,11], Gutknecht [12,13] or Gragg and Gutknecht [14].) If (4) is satisfied and $\varphi_t$ is unique, then $\varphi_t$ is called a *regular* FOP and $t$ a *regular index*. If we set

$$\varphi_t(z) =: u_{0,t} + u_{1,t}z + \cdots + u_{t-1,t}z^{t-1} + z^t,$$

then condition (4) translates into the *Yule–Walker* system

$$\begin{bmatrix} s_0 & s_1 & \cdots & s_{t-1} \\ s_1 & & \iddots & \vdots \\ \vdots & \iddots & & \vdots \\ s_{t-1} & \cdots & \cdots & s_{2t-2} \end{bmatrix} \begin{bmatrix} u_{0,t} \\ u_{1,t} \\ \vdots \\ u_{t-1,t} \end{bmatrix} = - \begin{bmatrix} s_t \\ s_{t+1} \\ \vdots \\ s_{2t-1} \end{bmatrix}. \tag{5}$$

Hence, the regular FOP of degree $t \geqslant 1$ exists if and only if the matrix $H_t$ is nonsingular.

The following theorem characterizes $n$, the number of mutually distinct zeros. It enables one, theoretically at least, to calculate $n$ as rank $H_N$.

**Theorem 2.** *$n = \operatorname{rank} H_{n+p}$ for every nonnegative integer $p$. In particular, $n = \operatorname{rank} H_N$.*

Therefore, $H_n$ is nonsingular whereas $H_t$ is singular for $t > n$. Note that $H_1 = [s_0]$ is nonsingular by assumption. The regular FOP of degree 1 exists and is given by $\varphi_1(z) = z - \mu$ where

$$\mu := \frac{s_1}{s_0} = \frac{\sum_{k=1}^{n} \nu_k z_k}{\sum_{k=1}^{n} \nu_k}$$

is the arithmetic mean of the zeros. Theorem 2 implies that the regular FOP $\varphi_n$ of degree $n$ exists and tells us also that regular FOPs of degree larger than $n$ do not exist. The polynomial $\varphi_n$ is easily seen to be

$$\varphi_n(z) = (z - z_1) \cdot \cdots \cdot (z - z_n). \tag{6}$$

It is the monic polynomial of degree $n$ that has $z_1, \ldots, z_n$ as simple zeros. This polynomial has the peculiar property that it is orthogonal to *all* polynomials (including itself),

$$\langle z^k, \varphi_n(z) \rangle = 0, \quad k = 0, 1, 2, \ldots .$$

Once $n$ is known, the mutually distinct zeros $z_1, \ldots, z_n$ can be calculated by solving a generalized eigenvalue problem. Indeed, let $H_n^<$ be the Hankel matrix

$$H_n^< := \begin{bmatrix} s_1 & s_2 & \cdots & s_n \\ s_2 & & \iddots & \vdots \\ \vdots & \iddots & & \vdots \\ s_n & \cdots & \cdots & s_{2n-1} \end{bmatrix}.$$

**Theorem 3.** *The eigenvalues of the pencil $H_n^< - \lambda H_n$ are given by $z_1, \ldots, z_n$.*

Once $z_1, \ldots, z_n$ have been found, the multiplicities $\nu_1, \ldots, \nu_n$ can be computed by solving the Vandermonde system

$$\begin{bmatrix} 1 & \cdots & 1 \\ z_1 & \cdots & z_n \\ \vdots & & \vdots \\ z_1^{n-1} & \cdots & z_n^{n-1} \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \vdots \\ \nu_n \end{bmatrix} = \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{bmatrix}. \tag{7}$$

**Note.** Vandermonde matrices are often very ill-conditioned [15,16]. In our case, however, the components of the solution vector of (7) are known to be integers, and therefore there is no problem, even if the linear system (7) happens to be ill-conditioned, as long as the computed approximations for the components of the solution vector have an absolute error that is less than 0.5.

Theorems 2 and 3 suggest the following approach to compute $n$ and $z_1, \ldots, z_n$. Start by computing the total number of zeros $N$. Next, compute $s_1, \ldots, s_{2N-2}$. As already mentioned, this can be done via numerical integration along $\gamma$. The number of mutually distinct zeros is then calculated as the rank of $H_N$, $n = \operatorname{rank} H_N$. Finally, the zeros $z_1, \ldots, z_n$ are obtained by solving a generalized eigenvalue problem. However, this approach has several disadvantages:

– Theoretically the $N - n$ smallest singular values of $H_N$ are equal to zero. In practice, this will not be the case, and it may be difficult to determine the rank of $H_N$ and hence the value of $n$ in case the gap between the computed approximations for the zero singular values and the nonzero singular values is too small.

– The approximations for $z_1, \ldots, z_n$ obtained via Theorem 3 may not be very accurate. Indeed, the mapping from the Newton sums to the zeros and their respective multiplicities,

$$(s_0, s_1, \ldots, s_{2n-1}) \mapsto (z_1, \ldots, z_n, \nu_1, \ldots, \nu_n), \tag{8}$$

is usually very ill-conditioned. (See, e.g., the papers by Gautschi [17–19] who studied the conditioning of (8) in the context of Gauss quadrature formulae.) A classical adage in numerical analysis says that one should avoid the use of ordinary moments.

In [7] Kravanja et al. have proposed an algorithm that gives more accurate approximations for $z_1, \ldots, z_n$. The idea is the following. The inner products that appear in the Hankel matrices $H_n$ and $H_n^<$ are related to the standard monomial basis. Why not consider a different basis? In other words, why not try to use modified moments instead of ordinary moments? The fact that

$$H_n = \left[\langle z^p, z^q\rangle\right]_{p,q=0}^{n-1} \quad \text{and} \quad H_n^< = \left[\langle z^p, zz^q\rangle\right]_{p,q=0}^{n-1}$$

suggests that one should consider the matrices

$$\left[\langle \psi_p, \psi_q\rangle\right]_{p,q=0}^{n-1} \quad \text{and} \quad \left[\langle \psi_p, \psi_1\psi_q\rangle\right]_{p,q=0}^{n-1}, \tag{9}$$

where $\psi_k$ is a polynomial of degree $k$ for $k = 0, 1, \ldots, n-1$. Of course, even if one succeeds in writing Theorem 3 in terms of the matrices that appear in (9), the question remains which polynomials $\psi_k$ to choose. Kravanja et al. [7] have found that very accurate results are obtained if one uses the formal orthogonal polynomials. In other words, the zeros of $\varphi_n(z)$ will be computed from inner products that involve $\varphi_0(z), \varphi_1(z), \ldots, \varphi_{n-1}(z)$. The value of $n$ will be determined indirectly. Before we can explain this in more detail, we have to say a few words about the orthogonality properties of FOPs.

If $H_n$ is strongly nonsingular, i.e., if all its leading principal submatrices are nonsingular, then we have a full set $\{\varphi_0, \varphi_1, \ldots, \varphi_n\}$ of regular FOPs.

What happens if $H_n$ is not strongly nonsingular? By filling up the gaps in the sequence of existing regular FOPs it is possible to define a sequence $\{\varphi_t\}_{t=0}^{\infty}$, with $\varphi_t$ a monic polynomial of degree $t$, such that if these polynomials are grouped into blocks according to the sequence of regular indices, then polynomials belonging to different blocks are orthogonal with respect to $\langle\cdot,\cdot\rangle$. More precisely, define $\{\varphi_t\}_{t=0}^{\infty}$ as follows. If $t$ is a regular index, then let $\varphi_t$ be the regular FOP of degree $t$. Else define $\varphi_t$ as $\varphi_r\psi_{t,r}$ where $r$ is the largest regular index less than $t$ and $\psi_{t,r}$ is an arbitrary monic polynomial of degree $t-r$. In the latter case $\varphi_t$ is called an *inner polynomial*. These polynomials $\{\varphi_t\}_{t=0}^{\infty}$ can be grouped into blocks. Each block starts with a regular FOP and the remaining polynomials are inner polynomials. Note that the last block has infinite length. The block orthogonality property is then expressed by the fact that the *Gram matrix* $G_n := [\langle\varphi_r, \varphi_s\rangle]_{r,s=0}^{n-1}$ is block diagonal. The diagonal blocks are nonsingular, symmetric and zero above the main antidiagonal. (See Bultheel and Van Barel [20] for more details.)

Theorem 3 can be interpreted in the following way: the zeros of the regular FOP of degree $n$ can be calculated by solving a generalized eigenvalue problem. The following theorem shows that this zero/eigenvalue property holds for all regular FOPs. This will enable us to compute regular FOPs in their product representation. The theorem also provides a solution to the problem of how to switch from ordinary moments to modified moments. Define the matrices $G_k$ and $G_k^{(1)}$ as

$$G_k := \left[\langle\varphi_r, \varphi_s\rangle\right]_{r,s=0}^{k-1} \quad \text{and} \quad G_k^{(1)} := \left[\langle\varphi_r, \varphi_1\varphi_s\rangle\right]_{r,s=0}^{k-1}$$

for $k = 1, 2, \ldots$.

**Theorem 4.** *Let $t \geqslant 1$ be a regular index and let $z_{t,1}, \ldots, z_{t,t}$ be the zeros of the regular FOP $\varphi_t$. Then the eigenvalues of the pencil $G_t^{(1)} - \lambda G_t$ are given by $\varphi_1(z_{t,1}), \ldots, \varphi_1(z_{t,t})$. In other words, they are given by $z_{t,1} - \mu, \ldots, z_{t,t} - \mu$ where $\mu = s_1/s_0$.*

**Corollary 1.** *The eigenvalues of* $G_n^{(1)} - \lambda G_n$ *are given by* $z_1 - \mu, \ldots, z_n - \mu$ *where* $\mu = s_1/s_0$.

Regular FOPs are characterized by the fact that the determinant of a Hankel matrix is different from zero, while inner polynomials correspond to singular Hankel matrices. To decide whether $\varphi_t(z)$ should be defined as a regular FOP or as an inner polynomial, one could calculate the determinant of $H_t$ and check if it is equal to zero. However, from a numerical point of view such a test "is equal to zero" does not make sense. Because of rounding errors (both in the evaluation of $\langle\cdot,\cdot\rangle$ and in the calculation of the determinant) one would encounter only regular FOPs. Strictly speaking one could say that inner polynomials are not needed in numerical calculations. However, the opposite is true! Let us agree to call a regular FOP *well-conditioned* if its corresponding Yule–Walker system (5) is well-conditioned, and *ill-conditioned* otherwise. To obtain a numerically stable algorithm, it is crucial to generate only well-conditioned regular FOPs and to replace ill-conditioned regular FOPs by inner polynomials. Stable look-ahead solvers for linear systems of equations that have Hankel structure are based on this principle [21–23]. In this approach the diagonal blocks in $G_n$ are taken (slightly) larger than strictly necessary to avoid ill-conditioned blocks.

The algorithm for calculating $z_1, \ldots, z_n$ that Kravanja et al. [7] have proposed proceeds by computing the polynomials $\varphi_0(z), \varphi_1(z), \ldots, \varphi_n(z)$ in their product representation, starting with $\varphi_0(z) \leftarrow 1$ and $\varphi_1(z) \leftarrow z - \mu$. At each step, to decide whether it is numerically feasible to generate the next polynomial in the sequence as a regular FOP, the algorithm uses a heuristic method. By doing a large number of numerical experiments, Kravanja et al. have reached the conclusion that their heuristic approach leads to accurate results. For more details, we refer to [7].

How does one obtain the value of $n$? Theorem 2 and Eqs. (3) and (6) imply the following.

**Theorem 5.** *Let* $t \geqslant n$. *Then* $\varphi_t(z_k) = 0$ *for* $k = 1, \ldots, n$ *and* $\langle z^p, \varphi_t(z)\rangle = 0$ *for all* $p \geqslant 0$.

The value of $n$ can be determined as follows. Suppose that the algorithm has just generated a (well-conditioned) regular FOP $\varphi_r(z)$. To check whether $n = r$, the algorithm scans the sequence

$$\big(\big|\langle (z-\mu)^\tau \varphi_r(z), \varphi_r(z)\rangle\big|\big)_{\tau=0}^{N-1-r}.$$

If all the elements are "sufficiently small", then the algorithm concludes that indeed $n = r$ and it stops. As we will explain in Section 3, the value of EPS_STOP, one of the input parameters of ZEAL, is used in this test.

As we have already mentioned, once $n$ and (approximations for) $z_1, \ldots, z_n$ have been found, the multiplicities $\nu_1, \ldots, \nu_n$ are computed by solving the Vandermonde system (7).

This concludes our discussion of the algorithm of Kravanja et al. More details can be found in [7].

## 3. The package ZEAL

Given a rectangle whose edges are parallel to the coordinate axes and a positive integer $M$, we take the following approach.

- We calculate the total number of zeros that lie inside this rectangle.
- Via consecutive subdivisions we obtain a set of subrectangles, each of which contains at most $M$ zeros (counting multiplicities).
- For each of these subrectangles, we calculate approximations for the zeros that lie inside it, together with their respective multiplicities.
- The approximations for the zeros are refined iteratively via the modified Newton's method.

As the function $f$ may have zeros on the boundary of the rectangular box specified by the user, ZEAL starts by perturbing this box. For this purpose a tolerance is used that is taken to be proportional to a power of the machine precision, for example, 10 times the square root of the machine precision. The box is then slightly enlarged in an

asymmetrical way. The reason for this asymmetric perturbation is to eliminate the possibility of having a zero close to or on any boundary of the consecutive subdivisions.

The total number of zeros of $f$ that lie inside the perturbed box is obtained in the same way as in the package ZEBEC [9]. The real part of the integral in (1) is written as a sum of four integrals, where each integral corresponds to one of the edges of the rectangular region. Approximations for these integrals are calculated via the adaptive integrator DQAG from the package QUADPACK [8]. A zero near one of the edges of the rectangle causes the integrand of the corresponding integral to have a "peak". The closer the zero lies to the edge, the sharper this peak is. If the zero lies on the edge, then the integral is divergent. DQAG uses adaptive strategies that enable it to cope with such peaks efficiently. However, if a zero lies too close to an edge (the corresponding peak is too sharp), then DQAG warns us that it has problems in calculating the integral. Our algorithm then slightly moves this edge and restarts. By enlarging the user's box, we may of course include additional zeros. We have decided not to discard any of these zeros ourselves. Rather, we provide the user with the box that eventually has been considered, all the zeros that lie inside this box, and leave it to him/her to filter out unwanted zeros.

If the starting box (as perturbed by ZEAL) contains less than $M$ zeros (counting multiplicities), then approximations for these zeros and the values of their respective multiplicities are computed via our implementation of the algorithm of Kravanja et al. [7]. Otherwise, the longest edges of the box are halved and the box is subdivided into two equal boxes. The number of zeros in each of these boxes is calculated via numerical integration. If DQAG detects a zero near the inner edge, then this edge is shifted, a process that results in an asymmetric subdivision of the box. Then the two smaller boxes are examined. A box that does not contain any zero is abandoned. If a box contains less than $M$ zeros, then approximations for these zeros are calculated, together with their respective multiplicities. A box that contains more than $M$ zeros is subdivided again. This process is repeated until a set of boxes has been found, each of which contains at most $M$ zeros, and approximations for all these zeros as well as the values of their respective multiplicities have been computed. The approximations for the zeros are then refined via the modified Newton's method, which takes into account the multiplicity of a zero and converges quadratically.

As we have already mentioned, if the function $f$ has a zero on the boundary of the initial rectangular region $W$ or on the boundary of one of the subregions of $W$ that the algorithm uses to isolate groups of zeros, then this may cause numerical integration problems. In some cases the small perturbations that ZEAL applies to the edges of the rectangular boxes do not solve these problems. QUADPACK uses certain heuristic strategies. They work very well but nevertheless can fail. For example, we have observed that in case the function has several zeros very close (at a distance less than $10^{-7}$) to the boundary, then DQAG may give a incorrect total number of zeros without giving any warning message. We advise the user to consider an initial region that is not symmetric with respect to the axes. In this way the possibility of having an imaginary or a real zero on the boundary of a subregion is minimized. If, in spite of this, during consecutive subdivisions there is a zero on a boundary, we recommend that the user perturbs the initial box asymmetrically.

### *3.1. The structure of ZEAL*

The package ZEAL (ZEros of AnaLytic functions) contains about 6 500 lines of code including comments. It is written in Fortran 90 and has been tested on various UNIX machines.

ZEAL consists of 11 parts, namely, the main program `Main` and the modules

- `Precision_Module`, in which the user can specify the precision to which the floating point calculations are to be done,
- `Zeal_Module`, which contains the main subroutine `ZEAL` and also the subroutines `CHECK_INPUT` and `ERROR_EXIT`,
- `Zeros_Module`, which contains the subroutines `APPROXIMATE` and `INPROD`,
- `Refine_Module`, which contains the subroutines `REFINE` and `NEWTON`,
- `Split_Module`, which contains the subroutines `INBOX` and `SPLITBOX`,

– and finally the modules `Error_Module`, `Quad_Module`, `Zeal_Input_Module`, `Function_Input_Module` and `Integration_Input_Module`.

ZEAL also requires the subroutine DQAG from the package QUADPACK [8] and a number of subprograms from the BLAS and LAPACK libraries [24].

The user can specify the values of the input parameters by editing the modules `Zeal_Input_Module`, `Function_Input_Module`, `Integration_Input_Module`. This will be discussed in Section 3.2.

The main program `Main` has the following form:

```
PROGRAM Main

  USE Precision_Module
  USE Zeal_Module

  IMPLICIT NONE

  INTEGER :: TOTALNUMBER, DISTINCTNUMBER, REFINEDNUMBER
  INTEGER, DIMENSION(:), POINTER          :: MULTIPLICITIES
  LOGICAL, DIMENSION(:), POINTER          :: REFINEMENT_OK
  COMPLEX(KIND=DP), DIMENSION(:), POINTER :: ZEROS, FZEROS

  CALL ZEAL(TOTALNUMBER,DISTINCTNUMBER,ZEROS,FZEROS,       &
            MULTIPLICITIES,REFINEDNUMBER,REFINEMENT_OK)

END PROGRAM Main
```

The subroutine `ZEAL` returns the total number of zeros of the given function that lie inside the given rectangular region, the number of mutually distinct zeros, the refined approximations for the zeros and the values that the function takes at these points, the corresponding multiplicities, the number of approximations for the zeros (as computed by the subroutine `APPROXIMATE`) that ZEAL has been able to refine successfully via the modified Newton's method, and finally for each computed zero a logical variable that indicates whether this refinement procedure has been successful or not.

In the design of ZEAL we have followed the recommendations for precision level maintenance described by Buckley [25]. The parameter `DP` that appears in the declaration of the variables `ZEROS` and `FZEROS` is defined in `Precision_Module`,

```
INTEGER, PARAMETER :: DP = SELECTED_REAL_KIND(15,70)
```

It determines the precision to which all the floating point calculations are to be done. Its current value corresponds to Fortran 77's `DOUBLE PRECISION`[7].

Let us briefly describe the various parts of ZEAL.

The subroutine `INBOX` calculates the total number of zeros that lie inside the rectangular box given by the user. If some of the zeros lie too close to the boundary of this box and the quadrature routine DQAG fails, then `INBOX` perturbs the box slightly and enlarges it.

[7] This observation is important for the following reason. As documented in its `makefile`, ZEAL uses certain Fortran 77 routines from QUADPACK, BLAS and LAPACK. To enable the user to compile the necessary routines from BLAS and LAPACK in case these libraries are not available on his/her computer system, we have included them with our distribution of ZEAL. However, we have included only the `DOUBLE PRECISION` version of these routines and hence they should be replaced by the corresponding `SINGLE PRECISION` routines in case a change to `DP` requires this. The same holds for the subroutine DQAG from QUADPACK.

The subroutine SPLITBOX takes a box and splits it into two boxes. A symmetric splitting, which proceeds by halving the longest edges, is tried first. If the calculation of the integral along the inner edge fails, then it is assumed that some of the zeros lie too close to this edge and the inner edge is shifted.

The subroutine APPROXIMATE contains our implementation of the algorithm of Kravanja et al. [7]. The symmetric bilinear form (3) is evaluated via the subroutine INPROD.

The subroutine NEWTON contains our implementation of the modified Newton's method. The subroutine REFINE calls NEWTON to refine the approximations for the zeros that APPROXIMATE has computed. If NEWTON fails, then REFINE tries again from a nearby point. If after eight attempts NEWTON still fails, then REFINE indicates that it has been unable to refine the given approximation successfully.

The subroutine ZEAL forms the main part of the package. ZEAL starts by calling CHECK_INPUT to check if the input parameters specified by the user are proper. Next it calls INBOX. If there are no zeros inside the user's box, then the program stops. If there are less than $M$ zeros inside the box (where the value of $M$ can be specified in Zeal_Input_Module), then APPROXIMATE and REFINE are called. Else, the box is given to SPLITBOX. The two boxes returned by SPLITBOX are examined. A box that does not contain any zero is abandoned. A box that contains less than $M$ zeros is given to APPROXIMATE and REFINE. A box that contains more than $M$ zeros is put in a list. Then ZEAL takes a next box from this list and calls SPLITBOX. This procedure is repeated until all the zeros have been computed, together with their respective multiplicities.

The program execution terminates normally after the completion of its task. This type of termination is indicated by the value 1 of the variable INFO, which is a global variable declared in the module Error_Module. If the value of this parameter is different from 1, then the termination of the program is abnormal. The cases of abnormal termination are the following:

INFO=0 Improper input parameters.
INFO=2 The procedure for the calculation of the total number of zeros has failed.
INFO=3 The procedure for the isolation of the zeros has failed.
INFO=4 The procedure for the computation of the zeros has failed.

### *3.2. ZEAL's user interface*

The user can specify the input parameters by editing three different files. (This splitting was done to speed up the recompilation in case only a few parameters are changed.)

In the module Zeal_Input_Module the following parameters have to be set:

LV a real array of length 2 that contains the $x$- and $y$-coordinates of the left lower vertex of the rectangle that is to be examined.

H a real array of length 2 that specifies the size of this rectangle along the $x$- and $y$-direction.

M an integer that determines the maximum number of zeros (counting multiplicities) that are considered within a subrectangle. M has to be larger than the maximum of the multiplicities of the zeros. A recommended value is 5.

ICON an integer in $\{1, \ldots, 4\}$ that specifies which calculations are to be done:

- 1 calculation of the total number of zeros, only,
- 2 calculation of the total number of zeros and isolation of a set of subrectangles, each of which contains at most M zeros,
- 3 calculation of the total number of zeros and computation of all the zeros, together with their respective multiplicities,
- 4 calculation of the total number of zeros and computation of NR zeros, together with their respective multiplicities.

Note that if ICON=4 the user must also supply the desired number of zeros NR. In the other cases (ICON=1,2,3) a value of NR may be supplied but it will not be used by the package.

VERBOSE a logical variable. ZEAL is allowed to print information (concerning the user's input and the computed results) if and only if VERBOSE is equal to .TRUE.

FILES a logical variable. If FILES is set equal to .TRUE. then ZEAL generates the files zeros.dat and mult.dat. They contain the computed approximations for the zeros as well as their respective multiplicities. ZEAL also writes the file fzeros.dat, which contains the values that the function takes at the computed approximations for the zeros.

IFAIL an integer that determines how errors are to be handled. We follow the NAG convention:

- 1 *soft silent error* – control is returned to the calling program.
- -1 *soft noisy error* – an error message is printed and control is returned to the calling program.
- 0 *hard noisy error* – an error message is printed and the program is stopped.

These parameters determine the geometry of the rectangular region that is to be considered and the type of calculation that ZEAL will perform.

In the module Integration_Input_Module the following parameters have to be set:

NUMABS a real variable that determines the absolute accuracy to which the integrals that calculate the number of zeros are to be evaluated. If NUMABS = 0.0_DP, then only a relative criterion will be used.

NUMREL a real variable that determines the relative accuracy to which the integrals that calculate the number of zeros are to be evaluated. If NUMREL = 0.0_DP, then only an absolute criterion will be used.

If NUMABS and NUMREL are both too small, then the numerical integration may be time-consuming. If they are both too large, then the calculated number of zeros may be wrong. The default values of NUMABS and NUMREL are 0.07_DP and 0.0_DP, respectively. These variables are used by QUADPACK.

INTABS a real variable that determines the absolute accuracy to which the integrals that are used to compute approximations for the zeros are to be calculated. If INTABS = 0.0_DP, then only a relative criterion will be used.

INTREL a real variable that determines the relative accuracy to which the integrals that are used to compute approximations for the zeros are to be calculated. If INTREL = 0.0_DP, then only an absolute criterion will be used.

If INTABS and INTREL are both too small, then the numerical integration may be time-consuming. If they are both too large, then the approximations for the zeros may be very inaccurate and Newton's method, which is used to refine these approximations (see NEWTONZ and NEWTONF), may fail. The default values of INTABS and INTREL are 0.0_DP and 1.0E-12_DP, respectively. These variables are used by QUADPACK.

EPS_STOP a real variable that is used in the stopping criterion that determines the value of $n$, the number of mutually distinct zeros. If EPS_STOP is too large, then the computed value of $n$ may be smaller than the actual number of distinct zeros. If EPS_STOP is too small, then the computed value of $n$ may be larger than the actual number of distinct zeros, especially in case the function has many multiple zeros. A recommended value is 1.0E-08_DP.

These parameters are related to numerical integration.

Finally, in the module Function_Input_Module the user has to specify two parameters that are used in the stopping criteria for Newton's method, the function $f$ whose zeros ZEAL has to compute as well as its first derivative. He or she also has to give some information about the analyticity of $f$ inside the considered region.

The parameters used to control the Newton's process are the following:

NEWTONZ and NEWTONF are real variables which should be specified in case ICON = 3 or 4. They are used as follows. The modified Newton's method, which takes into account the multiplicity of a zero and converges quadratically, is used to refine the calculated approximations for the zeros. The iteration stops if the relative distance between two successive approximations is at most NEWTONZ or the absolute value of the function at the last approximation is at most NEWTONF or if a maximum number of iterations (say, 20) is exceeded.

The considered function $f$ and its first derivative $f'$ should be defined via the subroutine FDF, which takes the following form:

```
SUBROUTINE FDF(Z,F,DF)

COMPLEX(KIND=DP), INTENT(IN)   :: Z
COMPLEX(KIND=DP), INTENT(OUT)  :: F, DF

F = ...
DF = ...

END SUBROUTINE FDF
```

If any cases where $f$ is not analytic are known, they have to be specified using the logical function VALREG. Given a rectangular region specified by its left lower vertex and the sizes of its edges, VALREG decides whether the function $f$ is analytic inside this region or not. ZEAL uses this information to decide whether it is allowed to move the edge of a box or not. VALREG has the following form:

```
FUNCTION VALREG(LV,H)

LOGICAL VALREG
REAL(KIND=DP), INTENT(IN) :: LV(2), H(2)

VALREG = ...

END FUNCTION VALREG
```

For example, if $f$ is analytic in the entire complex plane, then one may use the statement

```
VALREG = .TRUE.
```

If $f$ has a branch cut along the non-positive real axis, then one may write

```
VALREG = .NOT. (LV(2)*(LV(2)+H(2)) <= 0.0_DP .AND.   &
                LV(1) <= 0.0_DP)
```

This concludes our discussion of the structure of ZEAL and its user interface.

## 4. A few examples of how to use ZEAL

We will now discuss a few numerical examples.
Suppose that $f(z) = e^{3z} + 2z\cos z - 1$ and that

$$W = \{z \in \mathbb{C}\colon\ -2 \leqslant \operatorname{Re} z \leqslant 2,\ -2 \leqslant \operatorname{Im} z \leqslant 3\}.$$

In other words, $W$ is the rectangular region $[-2, 2] \times [-2, 3]$. Therefore, we have to define the input parameters LV and H as

`LV = (/-2.0_DP,-2.0_DP/)` and `H = (/ 4.0_DP, 5.0_DP/)`.

We set `M = 5`. The logical variable VERBOSE is set to `.TRUE.` We start by calculating only the total number of zeros, `ICON = 1`. ZEAL outputs the following.

**TEST RUN OUTPUT # 1**

```
This is ZEAL. Version of June 1999.

Input:

LV     =    -2.00000000000000      -2.00000000000000
H      =     4.00000000000000       5.00000000000000

M      =    5
ICON   =    1

FILES  =    T

================================================================

Results:

The following box has been considered:

  LV =  -2.00000016391277       -2.00000019371510
  H  =   4.00000035762787        5.00000041723251

Total number of zeros inside this box         =    4
```

The function has four zeros inside the given box. We now ask ZEAL to compute approximations for all these zeros, `ICON=3`.

**TEST RUN OUTPUT # 2**

```
This is ZEAL. Version of June 1999.

Input:

LV     =    -2.00000000000000      -2.00000000000000
H      =     4.00000000000000       5.00000000000000

M      =    5
ICON   =    3

FILES  =    T

================================================================

Results:

The following box has been considered:

  LV =  -2.00000016391277       -2.00000019371510
  H  =   4.00000035762787        5.00000041723251

Total number of zeros inside this box         =    4
Number of boxes containing at most   5  zeros =    1
```

```
These boxes are given by:
-------------------------

  1)  LV =  -2.00000016391277       -2.00000019371510
      H  =   4.00000035762787        5.00000041723251

      Total number of zeros inside this box =    4

Final approximations for the zeros and verification:
---------------------------------------------------

  1)  Number of mutually distinct zeros inside this box   =    4

      z    = (-1.84423395326221     ,-0.729696337329436E-29 )
      f(z) = ( 0.222044604925031E-15, 0.297690930716218E-28 )
      multiplicity =    1

      z    = ( 0.530894930292930    , 1.33179187675112     )
      f(z) = ( 0.888178419700125E-15, 0.222044604925031E-14 )
      multiplicity =    1

      z    = ( 0.530894930292930    ,-1.33179187675112     )
      f(z) = (-0.266453525910038E-14,-0.444089209850063E-15 )
      multiplicity =    1

      z    = ( 0.277555756299546E-16, 0.732694008769276E-26 )
      f(z) = ( 0.00000000000000     , 0.366347004384638E-25 )
      multiplicity =    1
```

If we set `M = 2`, then ZEAL outputs the following.

**TEST RUN OUTPUT # 3**

```
This is ZEAL. Version of June 1999.

Input:

LV     =    -2.00000000000000      -2.00000000000000
H      =     4.00000000000000       5.00000000000000

M      =    2
ICON   =    3

FILES  =    T

================================================================

Results:

The following box has been considered:

  LV =  -2.00000016391277       -2.00000019371510
  H  =   4.00000035762787        5.00000041723251
```

```
Total number of zeros inside this box         =    4

Number of boxes containing at most   2  zeros =    3

These boxes are given by:
-------------------------

  1)  LV =  -2.00000016391277        0.500000014901161
      H  =   4.00000035762787        2.50000020861626

      Total number of zeros inside this box =    1

  2)  LV =  -2.00000016391277       -2.000000019371510
      H  =   2.00000017881393        2.50000020861626

      Total number of zeros inside this box =    2

  3)  LV =   0.149011611938477E-07  -2.000000019371510
      H  =   2.00000017881393        2.50000020861626

      Total number of zeros inside this box =    1

Final approximations for the zeros and verification:
----------------------------------------------------

  1)  Number of mutually distinct zeros inside this box   =    1

      z    = ( 0.530894930292931    , 1.33179187675112     )
      f(z) = ( 0.888178419700125E-15,-0.177635683940025E-14 )
      multiplicity =    1

  2)  Number of mutually distinct zeros inside this box   =    2

      z    = (-1.84423395326221     ,-0.551251254781237E-21 )
      f(z) = ( 0.222044604925031E-15, 0.224891493487409E-20 )
      multiplicity =    1

      z    = (-0.501336236251204E-20,-0.135361644895767E-20 )
      f(z) = ( 0.000000000000000     ,-0.676808224478837E-20 )
      multiplicity =    1

  3)  Number of mutually distinct zeros inside this box   =    1

      z    = ( 0.530894930292931    ,-1.33179187675112     )
      f(z) = ( 0.888178419700125E-15,-0.444089209850063E-15 )
      multiplicity =    1
```

Finally, suppose that we want ZEAL to compute only two zeros. We set `ICON=4` and `NR=2`.

**TEST RUN OUTPUT # 4**

```
This is ZEAL. Version of June 1999.
```

```
Input:

LV     =    -2.00000000000000      -2.00000000000000
H      =     4.00000000000000       5.00000000000000

M      =    2
NR     =    2
ICON   =    4

FILES  =    T

================================================================

Results:

The following box has been considered:

  LV =  -2.00000016391277       -2.00000019371510
  H  =   4.00000035762787        5.00000041723251

Total number of zeros inside this box         =    4

Number of boxes containing at most   2  zeros =    3

These boxes are given by:
------------------------

  1)  LV =  -2.00000016391277        0.500000014901161
      H  =   4.00000035762787        2.50000020861626

      Total number of zeros inside this box =    1

  2)  LV =  -2.00000016391277       -2.00000019371510
      H  =   2.00000017881393        2.50000020861626

      Total number of zeros inside this box =    2

  3)  LV =   0.149011611938477E-07  -2.00000019371510
      H  =   2.00000017881393        2.50000020861626

      Total number of zeros inside this box =    1

Requested number of mutually distinct zeros =    2

Final approximations for the zeros and verification:
---------------------------------------------------

  1)  Number of mutually distinct zeros inside this box   =    1

      z    = ( 0.530894930292931    , 1.33179187675112     )
      f(z) = ( 0.888178419700125E-15,-0.177635683940025E-14 )
      multiplicity =    1
```

```
  2)  Number of mutually distinct zeros inside this box   =    2

      z    = (-1.84423395326221    , -0.551251254781237E-21 )
      f(z) = ( 0.222044604925031E-15, 0.224891493487409E-20 )
      multiplicity =    1
```

Suppose that $f(z) = z^2(z-1)(z-2)(z-3)(z-4) + z\sin z$ and let $W$ be the rectangular region determined by

`LV = (/-0.5_DP,-0.5_DP/)` and `H = (/ 6.0_DP, 2.0_DP/)`.

Note that $f$ has a double zero at the origin. We set `M=5` and `ICON=3`.

**TEST RUN OUTPUT #5**

```
 This is ZEAL. Version of June 1999.

 Input:

 LV     =   -0.500000000000000     -0.500000000000000
 H      =    6.00000000000000       2.00000000000000

 M      =    5
 ICON   =    3

 FILES  =    T

 ================================================================

 Results:

 The following box has been considered:

   LV = -0.500000163912773      -0.500000193715096
   H  =  6.00000035762787        2.00000041723251

 Total number of zeros inside this box         =    6

 Number of boxes containing at most   5  zeros =    2

 These boxes are given by:
 -----------------------

   1)  LV = -0.500000163912773      -0.500000193715096
       H  =  3.00000017881393        2.00000041723251

       Total number of zeros inside this box =    4

   2)  LV =  2.50000001490116       -0.500000193715096
       H  =  3.00000017881393        2.00000041723251

       Total number of zeros inside this box =    2
```

```
Final approximations for the zeros and verification:
---------------------------------------------------

  1)  Number of mutually distinct zeros inside this box   =    3

      z    = ( -0.555111512312578E-15, 0.774442308175991E-15 )
      f(z) = ( -0.729030243977503E-29,-0.214950920445209E-28 )
      multiplicity =    2

      z    = (  1.18906588973011     , 0.372342347264318E-27 )
      f(z) = (  0.00000000000000     ,-0.147405864729133E-26 )
      multiplicity =    1

      z    = (  1.72843498616506     , 0.189326617253043E-27 )
      f(z) = (  0.666133814775094E-15, 0.904485799870076E-27 )
      multiplicity =    1

  2)  Number of mutually distinct zeros inside this box   =    2

      z    = (  3.01990732809571    ,  0.481205152184817E-28 )
      f(z) = ( -0.105471187339390E-14,-0.104420423665203E-26 )
      multiplicity =    1

      z    = (  4.03038191606047    ,  0.394430452610506E-28 )
      f(z) = (  0.150990331349021E-13, 0.421365501391524E-26 )
      multiplicity =    1
```

Finally, suppose that $f(z) = z^2(z-2)^2[e^{2z}\cos z + z^3 - 1 - \sin z]$ and let $W$ be the region determined by

`LV = (/-1.0_DP,-1.0_DP/)` and `H = (/ 4.0_DP, 2.0_DP/)`.

Note that $f$ has a triple zero at the origin and a double zero at $z = 2$. We set `M=5` and `ICON=3`.

**TEST RUN OUTPUT # 6**

```
This is ZEAL. Version of June 1999.

Input:

LV     =    -1.00000000000000      -1.00000000000000
H      =     4.00000000000000       2.00000000000000

M      =    5
ICON   =    3

FILES  =    T

================================================================

Results:

The following box has been considered:
```

```
  LV =  -1.00000016391277       -1.00000019371510
  H  =   4.00000035762787        2.00000041723251

Total number of zeros inside this box         =    8

Number of boxes containing at most   5  zeros =    2

These boxes are given by:
------------------------

  1)  LV =  -1.00000016391277       -1.00000019371510
      H  =   2.00000017881393        2.00000041723251

      Total number of zeros inside this box =    5

  2)  LV =   1.00000001490116       -1.00000019371510
      H  =   2.00000017881393        2.00000041723251

      Total number of zeros inside this box =    3

Final approximations for the zeros and verification:
----------------------------------------------------

  1)  Number of mutually distinct zeros inside this box   =    3

      z    = (-0.460714119728972    ,-0.625427769347768     )
      f(z) = ( 0.139986423568309E-14, 0.948597781037581E-14 )
      multiplicity =    1

      z    = (-0.749400541621981E-15, 0.243022572576853E-15 )
      f(z) = (-0.120813528135193E-44, 0.162080855465761E-44 )
      multiplicity =    3

      z    = (-0.460714119728972    , 0.625427769347768     )
      f(z) = (-0.383075548142431E-15,-0.579735693200359E-14 )
      multiplicity =    1

  2)  Number of mutually distinct zeros inside this box   =    2

      z    = ( 2.00000000000000    ,  0.114762494171498E-14 )
      f(z) = (-0.122427664771369E-26,-0.678041409962963E-27 )
      multiplicity =    2

      z    = ( 1.66468286974552    , -0.863802691217008E-28 )
      f(z) = ( 0.276741773088933E-15, 0.662785838981764E-27 )
      multiplicity =    1
```

## 5. Concluding remarks

We have applied ZEAL to various analytic functions and rectangular regions and have found that it behaves predictably and accurately. ZEAL calculates the total number of zeros that lie inside the given box and then

computes approximations for these zeros, together with their respective multiplicities. Our package does not require initial approximations for the zeros.

The user will appreciate the flexibility offered by the input parameter `ICON`. If nothing is known about the zeros that lie inside the given box, one may call ZEAL with `ICON = 1` to obtain the total number of zeros. Then one may proceed with `ICON = 3` to compute approximations for all these zeros, or, if less than the total number of zeros are required, with `ICON = 4` and `NR` equal to the requested number of zeros. If only a set of boxes is required, each of which contains less than `M` zeros (counting multiplicities), then one may set `ICON = 2`.

## Acknowledgements

One of the authors (P.K.) would like to thank the Department of Mathematics of the University of Patras (Patras, Greece) for its hospitality, during which time this paper was initiated, and Vlaamse Leergangen Leuven for supporting his visit to Patras. He also wishes to acknowledge the support granted to him by the Flemish Institute for the Promotion of Scientific and Technological Research in the Industry (IWT).

This research was partially supported by projects # G.0278.97 (“Orthogonal Systems and their Applications”) and # G.0261.96 (“Counting and Computing all Isolated Solutions of Systems of Nonlinear Equations”) of the Fund for Scientific Research – Flanders (FWO – Vlaanderen).

## References

[1] L.M. Delves, J.N. Lyness, Math. Comput. 21 (1967) 543.
[2] L.C. Botten, M.S. Craig, R.C. McPhedran, Comput. Phys. Commun. 29 (1983) 245.
[3] N.I. Ioakimidis, Quadrature methods for the determination of zeros of transcendental functions – a review, in: Numerical Integration: Recent Developments, Software and Applications, P. Keast, G. Fairweather, eds. (Reidel, Dordrecht, 1987) pp. 61–82.
[4] M.P. Carpentier, A.F.D. Santos, J. Comput. Phys. 45 (1982) 210.
[5] J.H. Wilkinson, Numer. Math. 1 (1959) 150.
[6] T.-Y. Li, SIAM J. Numer. Anal. 20 (1983) 865.
[7] P. Kravanja, T. Sakurai, M. Van Barel, BIT 39 (4) (1999) 646.
[8] R. Piessens, E. de Doncker-Kapenga, C.W. Überhuber, D.K. Kahaner, QUADPACK: A Subroutine Package for Automatic Integration, Springer Series in Computational Mathematics, Vol. 1 (Springer, Berlin, 1983).
[9] P. Kravanja, O. Ragos, M.N. Vrahatis, F.A. Zafiropoulos, Comput. Phys. Commun. 113 (1998) 220.
[10] A. Draux, Polynômes Orthogonaux Formels – Applications, Lecture Notes in Mathematics, Vol. 974 (Springer, Berlin, 1983).
[11] A. Draux, Numer. Algorithms 11 (1996) 143.
[12] M.H. Gutknecht, SIAM J. Matrix Anal. Appl. 13 (1992) 594.
[13] M.H. Gutknecht, SIAM J. Matrix Anal. Appl. 15 (1994) 15.
[14] W.B. Gragg, M.H. Gutknecht, Stable look-ahead versions of the Euclidean and Chebyshev algorithms, in: Approximation, Computation: A Festschrift in Honor of Walter Gautschi, R.V.M. Zahar, ed. (Birkhäuser, Basel, 1994) pp. 231–260.
[15] W. Gautschi, How (un)stable are Vandermonde systems?, in: Asymptotic and Computational Analysis: Conference in Honor of Frank W.J. Olver’s 65th Birthday, Lecture Notes in Pure and Applied Mathematics, Vol. 124, R. Wong, ed. (Marcel Dekker, Basel, 1990) pp. 193–210.
[16] W. Gautschi, G. Inglese, Numer. Math. 52 (1988) 241.
[17] W. Gautschi, Math. Comput. 24 (1970) 245.
[18] W. Gautschi, SIAM J. Sci. Stat. Comput. 3 (1982) 289.
[19] W. Gautschi, Numer. Math. 48 (1986) 369.
[20] A. Bultheel, M. Van Barel, Linear Algebra, Rational Approximation and Orthogonal Polynomials, Studies in Computational Mathematics, Vol. 6 (North-Holland, Amsterdam, 1997).
[21] A.W. Bojanczyk, G. Heinig, J. Complexity 10 (1994) 142.
[22] S. Cabay, R. Meleshko, SIAM J. Matrix Anal. Appl. 14 (1993) 735.
[23] R.W. Freund, H. Zha, Numer. Math. 64 (1993) 295.
[24] E. Anderson et al., LAPACK Users’ Guide (SIAM, Philadelphia, PA, 1994).
[25] A.G. Buckley, ACM Trans. Math. Software 20 (1994) 308.