Iterative polynomial-root-finding procedure with enhanced accuracy

Hashim A. Yamani (a) and Abdulaziz D. Alhaidari (b) *

(a) Mabuth-2414, Medina 42362-6959, Saudi Arabia

(b) Saudi Center for Theoretical Physics, P.O. Box 32741, Jeddah 21438, Saudi Arabia

Abstract. We devise a simple but remarkably accurate iterative routine for calculating the roots of a polynomial of any degree. We demonstrate that our results have significant improvement in accuracy over those obtained by methods used in popular software packages.

Keywords: polynomial roots, iterative procedure, recursion relation, deflated polynomial, Fibonacci numbers.

1. Introduction

The search for algebraic formulas for the zeros of higher order polynomial equations stopped at the fourth order (the quartic equation) because in 1820 the Norwegian mathematician Abel showed that no such formula exists for degrees higher than 4 [1]. Thus, numerical routines for finding the roots of polynomials of degrees higher than four were needed and the race to come up with the most efficient routine (in convergence, accuracy, speed and stability) was on and still going. In this work, we develop a highly accurate numerical procedure for calculating the roots of a polynomial of any degree. The procedure is inspired by the known simple scheme of finding the asymptotic limit ratio of consecutive Fibonacci number [2]. When the scheme is reversed, it leads to a procedure for finding numerical values of the zeros of quadratic, cubic and higher equations. For the benefit of readers who are not familiar with the Fibonacci numbers and/or the scheme of finding their asymptotic limit ratio, we give an introduction in the Appendix where we also illustrate that by reversing the scheme we could develop a numerical routine for finding the zeros of quadratic and higher equations.

We write the target equation, whose zeros we seek to find, in terms of a monic polynomial of degree \( n \) as follows

\[
P_n(z) = z^n + a_{n-1}z^{n-1} + a_{n-2}z^{n-2} + \ldots + a_1z + a_0 = 0.
\] (1)

We then divide this equation by \( z^{n-1} \) and replace the \( z \)’s in the resulting equation by \( n \) ratios of consecutive terms \( Q_{k+1}/Q_k \) of an \( (n+1) \)-term recursion relation for \( \{Q_k\} \) associated with (1). In the Appendix, we show how this is done for \( n = 2 \) and \( n = 3 \). Starting with an arbitrarily chosen \( n \) initial values \( \{Q_k\}_{k=0}^{n-1} \), we have also shown that the resulting iterative scheme, when

* Corresponding Author: email address: haidari@sctp.org.sa
augmented by an averaging process, gives accurate numerical evaluation for one of the zeros of Eq. (1). If we call this approximate root \( \bar{z}_0 \), then we can factor it out and write

\[ P_n(z) \approx (z - \bar{z}_0) \tilde{P}_{n-1}(z) \]

where \( \tilde{P}_{n-1}(z) \) is called the \textit{deflated polynomial}, which has the same form as \( P_n(z) \) but with \( n \mapsto n - 1 \) and with new coefficients \( \{\tilde{a}_m\}_{m=0}^{n-2} \). The procedure is then repeated for the deflated polynomial of degree \( n - 1 \)

\[ \tilde{P}_{n-1}(z) = P_n(z)/(z - \bar{z}_0) = z^{n-1} + \tilde{a}_{n-2}z^{n-2} + \ldots + \tilde{a}_1z + \tilde{a}_0, \quad (2) \]

to find an approximation for the second root \( \bar{z}_1 \), and so on. Inspired by the success of this procedure, as outlined in the Appendix, we wondered whether we could do away with the starting recursion relation associated with Eq. (1) and just begin by an arbitrary initial guess values for the roots as \( \{\bar{z}_m\}_{m=0}^{n-1} \) and improve on \( \bar{z}_0 \) by rewriting Eq. (2) as

\[ \bar{z}_0 = z - \frac{P_n(z)}{\tilde{P}_{n-1}(z)} = z - \frac{P_n(z)}{(z - \bar{z}_1)(z - \bar{z}_2)(z - \bar{z}_3)\ldots(z - \bar{z}_{n-1})}, \quad (3) \]

Then we start an iterative sequence by the replacement \( z \mapsto R_k \) and \( \bar{z}_0 \mapsto R_{k+1} \) for \( k = 0,1,\ldots,N-1 \) where \( N \) is some large enough integer. Consequently, the resulting iterative sequence has very close resemblance to Eq. (A5) and (A9) in the Appendix. In the following section, we show that this scheme does in fact lead to a remarkably accurate procedure for finding the roots of a general polynomial of any degree that compares favorably with other routines in popular computational software packages. We demonstrate the accuracy of the procedure by giving an illustrative example of a polynomial of degree 20 with complex coefficients and show that the results obtained are superior to those found by some of the built-in routines in the popular computational software Mathcad®.

2. The procedure

We start by the observation construed from the investigation carried out in the Appendix and advanced in the previous section, which is that the iterative procedure for finding subsequent roots of quadratic and cubic equations in terms of previously obtained ones (with modest level of accuracy) is rapidly convergent and with progressively improved accuracy. Consequently, we can do away with the averaging procedure used therein. Moreover, due to the rapid convergence, it would be conceivable that by starting with reasonable but otherwise arbitrary initial guessed values of the zeros, we could obtain highly accurate values after some small number of rounds of iteration.

Therefore, the problem that we are set to solve is finding the roots \( \{z_m\}_{m=0}^{n-1} \) of the monic polynomial of degree \( n \) given by Eq. (1), which could be rewritten as \( P_n(z) = (z - z_0)(z - z_1)\ldots(z - z_{n-1}) \). We start with a set of guessed values \( \{\bar{z}_m\}_{m=0}^{n-1} \) for the roots and then improve on \( \bar{z}_0 \)
3. Results and discussion

As illustration, we used the computational software package Mathcad® (version 14) to write a routine (Khandug) implementing the procedure in the previous section as shown in Figure 1. We compare the accuracy of the routine to that of the built-in \texttt{polyroots} function in Mathcad, which utilizes either the “La Guerre (LG)” method or the “Companion Matrix (CM)” method [3]. We took $n = 20$ and made an arbitrary choice of the complex coefficients $\{a_m\}_{m=0}^{19}$. We have chosen the initial guessed values for the roots $\{z_m\}_{m=0}^{19}$ as 20 equally spaced points on the
unit circle in the complex plane. The routine executed one iteration step given by Eq. (6) for each root (i.e., \( N = 1 \)) and then the whole procedure was repeated 17 rounds to reach the desired accuracy. Table I shows a comparison of the accuracy of the roots obtained by our iterative routine \texttt{Khandug} to those obtained by \texttt{polyroots} using the CM and LG methods separately. The second column in Table I is a measure of how good (or bad) our initial guessed values by giving \( \left( \left| P_n(\zeta_n) \right| \right)_{m=0}^{19} \). It is clear from Table I that the accuracy of our results is superior to those obtained by the two methods in Mathcad. Other tests using the popular computational software Mathematica\textregistered do also confirm the preeminence of our routine [4]. We also compared our results to those obtained by two FORTRAN routines from the SLATEC Common Mathematical Library [5], \texttt{CPZERO} and \texttt{CPQE79}, with the same conclusion that \texttt{Khandug} is not just more accurate but also the fastest [6]. It should be mentioned that we could have also reached the same desired accuracy as in Table I by performing 30 iterations for each root (\( N = 30 \)) and then repeating the procedure for three rounds instead of 17. However, this takes a total of 1800 iteration step whereas for Table I it is only 340. We show in Table II the lowest iteration sequences in which the process converges to the desired accuracy for the same initial guessed roots.

In the fifth and sixth columns of Table I, it is peculiar to see that Mathcad produced a relatively large number for one of the roots, which is at least nine or eleven orders of magnitudes larger than the largest of the other numbers in the corresponding column. Both large numbers correspond to the root “\(-0.09074+5.81549i\)” whose accuracy in our calculation is even better than \(10^{-307}\). The same peculiar result is produced by Mathematica [4]. This result is not a precision-related issue since it persists even after increasing precision to the maximum allowed by our computing hardware and software. Of course, if we repeat the calculation with increased precision, the value for this number decreases but its relatively large size to the rest of the numbers in the column remains the same. This is possibly due to the polarity in the distribution of the roots where this particular root is far from the rest that are in close proximity to each other. The important question now is as follows: Considering the results produced by Mathcad in Table I and not being aware of our result, how can one decide whether the root corresponding to this relatively large number is in fact a true root? We believe that the success of our scheme in avoiding such peculiarity is in the robustness of the iterative procedure that resulted in improvement of ALL roots on equal footings whereas in Mathcad, it seems that the isolated root is not treated equitably and the accumulated errors are dumped into that root. Of course, the error is relatively small but substitution in such high degree polynomial with complex coefficients does magnify the error greatly. Nonetheless, we should also note that our routine does have similar behavior if the degree of the polynomial is very large.

We believe that a rigorous assessment of the utility, accuracy and convergence properties of our routine by specialists in the field of computing software development will shed light on its utility as a viable alternative to other methods. Such assessment may lead to an improvement on the efficiency of the routine. For example, the total number of iterations, \( K \), could be minimized in lieu of the iteration rounds \( J \) and/or the iteration steps \( N \). Additionally, a better algorithm to choose the initial guessed roots may also be developed. Nonetheless, we describe here a simple but meaningful efficiency test using the polynomial example above as shown in Table III. The Table gives a rough estimate of the convergence, accuracy and speed of the routine as explained in the Table caption. In another test that demonstrates robustness of the
scheme, we used a random number generator with uniform distribution to construct the complex coefficients of a polynomial with degree 99 [7]. The real and imaginary parts of the coefficients were distributed randomly in the range [−5, +5]. Convergence is reached for several choices of initial guess values and number of iterations and with accuracy superior to that of Mathcad.

To illustrate how the accuracy of the individual roots develop with iteration, we plot in Figure 2 $|P_n(z_m)|$ corresponding to $N = 1$ of Table II for each iteration step until the desired accuracy is reached after 17 rounds. Figure 3 shows the same for the two cases corresponding to $N = 17$ and $N = 30$ in Table II but for clarity we show only the first 10 roots. Additional figures are included in the supplementary material linked to this work.

Finally, we make the following relevant observations:

1. Our scheme does not accept initial guessed roots $\{z_m\}_{m=0}^{n-1}$ that are degenerate (i.e., two or more guessed roots are equal). Otherwise, divergences will occur. For that reason, we placed them separated on a circle centered at the origin of the complex plane. On the other hand, the scheme is successful in finding degenerate roots.

2. Convergence improves if we place the initial guessed roots separated on an outward or inward spiral. Table IV is a reproduction of Table II with the exception that instead of placing the guessed roots on the unit circle, we fan them out on a $2\pi$-spiral starting from $x = 0.5$ to $x = 1.5$. It is evident that this configuration leads to more possibilities for reaching convergence with less total number of iterations. However, numerical divergences may occur for very large degrees unless we place the initial guessed roots on the unit circle.

3. The improvement in accuracy of our results over those obtained by Mathcad becomes more evident if the distribution of the zeros is such that one or more are located far enough from the rest in the complex plane.

4. All results obtained by our routine in this section were under the condition of “hard” convergence where all $\{|P_n(z_m)|\}_{m=0}^{n}$ are required to be smaller than a given accuracy threshold. However, “soft” convergence allows for more possibilities to obtain accurate results if we demand that only the average of the set $\{|P_n(z_m)|\}_{m=0}^{n}$ remains constant. In the procedure Khandug shown by Figure 1, we adopted this soft convergence. On the other hand, accuracy increases in soft convergence if we include only the most accurate portion of the set $\{|P_n(z_m)|\}_{m=0}^{n}$ (the lower end portion of the sorted set) in the averaging rather than the whole set.

5. For a given set of coefficients $\{a_m\}_{m=0}^{n-1}$ there exist an optimum number of iterations $N = N_c$ to achieve maximum accuracy in the values of the computed roots. Increasing $N$ beyond $N_c$ will result in diminished accuracy.

6. The hope is that further effort by professionals in software programming will result in enhancing the efficiency of our proposed scheme while maintaining the superior accuracy that it enjoys.
Acknowledgements:

We appreciate very much the assistance by Ahmed Jellal, Abdellatif Kamal, Hasan Abdullah and Abdelhadi Bahaoui in the calculation using Mathematica and FORTRAN. We are also grateful to “Al-Khandug School, Al-Madinah” for hospitality during the progress of this work.

Appendix: Solving quadratic and higher equations à la Fibonacci

In this Appendix, we show that the known simple scheme of finding the asymptotic limit ratio of consecutive Fibonacci numbers can be reversed to find numerical values of the zeros of quadratic, cubic and higher equations. The Fibonacci number sequence is very simple [2]. It starts with the first two members (usually $F_0 = 0$ and $F_1 = 1$) and then proceed to propagate as

$$ F_{n+2} = F_n + F_{n+1}, \quad n = 0, 1, \ldots $$  \hspace{1cm} (A1)

It is a known fact that for high index $n$, the ratio $(F_{n+1}/F_n)$ accurately approximates the golden ratio $(1 + \sqrt{5})/2$, regardless of the starting values, $F_0$ and $F_1$ of the sequence. Writing this ratio as $(R_n = F_{n+1}/F_n)$, then dividing (A1) by $F_{n+1}$ gives

$$ R_{n+1} = \frac{1}{R_n} + 1 \hspace{1cm} (A2) $$

The above argument could be developed in reverse to find accurate numerical values of the zeros of general quadratic, or higher, equations. If we are interested in solving for the zeros of the quadratic equation $F(z) = \beta z + \gamma = 0$, the solution satisfies the equation

$$ z = -\frac{\gamma}{z} - \beta \hspace{1cm} (A3) $$

We now assume that the solution to this equation emerges as the ratio $(R_n = Q_{n+1}/Q_n)$ of a sequence $\{Q_n\}$ in the asymptotic limit. Implementing this idea and generalizing (A2), we demand that this sequence satisfies the following iterative equation corresponding to (A3)

$$ R_{n+1} = -\frac{\gamma}{R_n} - \beta \hspace{1cm} (A4) $$

Thus, the sequence is generated from initial values of $Q_0$ and $Q_1$ together with the recursion relation $Q_{n+2} = -\gamma Q_n - \beta Q_{n+1}$. Hence, the asymptotic limit of the ratio $Q_{n+1}/Q_n$ does not depend on the initial values, thus leaving us with the freedom to choose values we deem suitable. Now, suppose that after having generated the sequence $\{Q_n\}$ and having chosen a value $R$ as stable converged value of the sequence $\{R_n\}$, we then take $R$ to stand for the first zero of the function...
If \( \bar{W} \) is a good approximation to the second zero, we have \( F(z) \approx (z - \bar{R})(z - \bar{W}) \). This means that \( \bar{W} \approx z - \frac{F(z)}{z - \bar{R}} \). We then search for a converging sequence \( \{W_k\} \) that is generated by the equation

\[
W_{n+1} = W_n - \frac{F(W_n)}{W_n - \bar{R}} \tag{A5}
\]

We are free to choose any suitable initial value. We note that the quality of the results can be measured by how small the quantities \( |F(\bar{R})| \) and \( |F(\bar{W})| \).

We also note in passing that for complex zeros, the recursion (A4) with real coefficients \( \beta \) and \( \gamma \) may not converge and the resulting numbers of the sequence are located on an ellipse-like shape in the complex plane as shown in Fig. 4, where we also indicate the position of the exact root by a solid square. Taking the average \( \langle R \rangle \equiv \frac{1}{N-M+1} \sum_{k=M}^{N} R_k \) for large enough \( M \) and even larger \( N \) yields a modest approximation to the root of the quadratic equation. Now we equate this value with \( \bar{R} \) in Eq. (A5) to generate the sequence associated with the second root. Doing so, we get remarkably much more accurate result for \( \bar{W} \) as evidenced by the very low value of the accuracy measure \( |F(\bar{W})| \). Encouraged by this boost in accuracy, we could try to find an improved version of the first zero \( \langle R \rangle \) using the new found value of \( \bar{W} \) and using the same equation (A5) but with the exchange \( \{R_k\} \leftrightarrow \{W_k\} \) and \( \bar{R} \leftrightarrow \bar{W} \). The result is a substantially improved value of the first zero where \( |F(\bar{R})| << |F(\langle R \rangle)| \).

The above procedure could be extended to find the three solutions that satisfy the cubic equation: \( G(z) = z^3 + \beta z^2 + \gamma z + \delta = 0 \). We start by dividing this equation by \( z^2 \) to obtain

\[
z = -\beta - \frac{\gamma}{z} - \frac{\delta}{z^2} \tag{A6}
\]

We then assume that the solution can be obtained by the converged ratio \( R_n \equiv \frac{Q_{n+1}}{Q_n} \) of the sequence of quantities \( \{Q_k\} \). We choose to make the following substitutions in Eq. (A6)

\[
R_{n+1} = -\beta - \frac{\gamma}{R_{n+1}} - \frac{\delta}{R_{n+1} R_n}, \tag{A7}
\]

which is a four-term recursion relation for the set \( \{Q_k\} \). To generate the sequence that ends with \( \bar{R} \), we only need to choose suitable initial values for \( R_0 \) and \( R_1 \). If we now extract the factor \( (z - \bar{R}) \) from \( G(z) \), we are then left with a quadratic equation \( \tilde{F}(z) \) that satisfies the relation

\[
\tilde{F}(z) = \frac{G(z)}{z - \bar{R}} = z^2 + \tilde{\beta} z + \tilde{\gamma} \tag{A8}
\]
Now, we apply the technique used to solve for the zeros of the quadratic equation and search for the second zero \( W \) using the asymptotically converging sequence \( \{W_k\} \) of Eq. (A5) with \( F(z) \mapsto \tilde{F}(z) \). Finally for the third zero of the cubic equation \( G(z) = 0 \), we generate the sequence \( \{L_k\} \) satisfying the recursion relation

\[
L_{n+1} = L_n - \frac{G(L_n)}{(L_n - \tilde{W})(L_n - \tilde{R})}
\] (A9)

The above scheme can be generalized to compute the zeros of higher degree equations. Note that the procedure of finding the first zero, which is exemplified by Eq. (A4) and Eq. (A7) for the quadratic and cubic equations, respectively is not new (see, for example, Ref. [8]). On the other hand, computing the remaining zeros using an iterative procedure based on factorization like that in Eq. (A5) and (A9) is not fully and successfully utilized. In this paper, we overcome this deficiency and develop a scheme inspired by the findings in this Appendix whereby we develop a highly accurate routine to calculate the roots of a polynomial of any degree.

References:

[1] Al Cuoco, *Mathematical connections: A companion for teachers and others* (The Mathematical Association of America & EDC, 2005) Sec. 2.5

[2] References are abound about the Fibonacci number system. Our favorite is the exposition in Keith Ball, *Strange Curves, Counting Rabbits and Other Mathematical Explorations* (Princeton University Press, 2003) Chap.8

[3] The Mathcad code for our subroutine Khandug is available upon request from the authors.

[4] Private communication with Ahmed Jellal (Laboratory of Theoretical Physics, Chouaib Doukkali University, Morocco) and Hasan M. Abdullah (Physics Department, King Fahd University of Petroleum and Minerals, Saudi Arabia)

[5] SLATEC CML is a comprehensive software library for general-purpose mathematical and statistical routines written in Fortran 77: http://www.netlib.org/slatec/

[6] Private communication with Abdelhadi Bahaoui (Laboratory of Theoretical Physics, Chouaib Doukkali University, Morocco)

[7] We used this particular polynomial degree because our version of Mathcad does not allow for degrees larger than 99 for the built-in function `polyroots`.

[8] B. A. Brousseau, *Linear Recursion Relations - Lesson Eight: Asymptotic Ratios in Recursion Relations*, The Fibonacci Quarterly 8, 311 (1970)
Figures Caption

Fig. 1: The procedure Khandug with “soft” convergence that could be turned into a computational routine using any convenient programming language. The output of the routine consists of the computed roots \( \{ \xi_m \}_{m=0}^n \) and the number of iteration rounds \( J \).

Fig. 2 (color online): Plot of \( \log_{10}\left| P_n(\xi_m) \right| \) as it develops with iteration until the desired accuracy is reached for the individual roots of the polynomial example with \( N=1 \). Part a (Part b) is for the first (last) 10 roots.

Fig. 3 (color online): Same as Figure 1 but for \( N=17 \) (part a) and \( N=30 \) (part b). For brevity, we show traces for the first 10 roots only.

Fig. 4: The sequence \( R_n = Q_{n+1}/Q_n \) (solid circles) calculated using the iterative procedure (A4) starting with some complex seed values \( Q_0 \) and \( Q_1 \). The exact solution is indicated by the solid square.

Tables Caption:

Table I: Comparison of the accuracy of our iterative routine Khandug to that of the built-in function polyroots in Mathcad® with the CM and LG methods. The polynomial coefficients \( \{ a_m \}_{m=0}^n \) for \( n=20 \) are listed in the first column. The roots obtained by our iterative procedure are shown in the third column. The accuracy measure is given by the sorted set \( \left\{ \left| P_n(\xi_m) \right| \right\}_{m=0}^{n-1} \) for the roots obtained by each of the three methods. The “0.0000” is a number less than \( 10^{-307} \). The individual roots in the third column are not in correspondence with the entries in the last three due to sorting.

Table II: Alternative convergent iteration sequences for finding the roots of the example polynomial to the desired accuracy shown in Table I and for the same set of initial guess roots, which are equally spaced on the unit circle in the complex plane. The number of rounds of the iteration procedure is \( J \) whereas \( K \) is the total number of iteration steps: \( K = n \cdot N \cdot J \).

Table III: Efficiency analysis of the accuracy and speed of convergence of our root-finding routine for the example given in Section 3. \( r \) is the radius of the circle on which the initial guess roots are placed at equal separation. We vary \( r \) from 0.2 to 2.2 in 10 steps until convergence is reached (exception is the \( N=5 \) case, where we took 50 steps). The minimum, maximum and average of \( \left\{ \left| P_n(\xi_m) \right| \right\}_{m=0}^{n-1} \) (not including 0.000) are shown for each choice of \( N \). The total number of iteration steps is \( K = n \cdot N \cdot J \), where \( J \) is the number of rounds of the iterative procedure to reach the desired accuracy given in Table I.

Table IV: Reproduction of Table II with the exception that instead of placing the initial guess roots on the unit circle, we fan them out on a \( 2\pi \)-spiral that starts from \( x=0.5 \) to \( x=1.5 \). Improved convergence is evident.
## Table I

| $\{a_m\}_{m=0}^n$ | $\{|P_n(z_m)|\}_{m=0}^{n-1}$ | $\{|\tau_m\}_{m=0}^n$ | Khandug ($10^{-14}$) | polyoots (CM) ($10^{-13}$) | polyroots (LG) ($10^{-5}$) |
|------------------|-----------------------------|-----------------------|----------------------|-----------------------------|-----------------------------|
| 2+3i             | -0.09074+5.81549i           | 0.0000                | 0.0157               | 1.5590×10^{-10}             |
| -1               | 0.89105+0.21904i            | 0.0366                | 0.0673               | 2.1105×10^{-9}              |
| -3-7i            | 0.54897+0.52784i            | 0.0744                | 0.0703               | 0.0291                      |
| 5                | 0.74874+0.81131i            | 0.0828                | 0.0735               | 0.0317                      |
| 7+3i             | 0.34122+1.15526i            | 0.1038                | 0.0742               | 0.0322                      |
| 1+i              | -0.05401+0.89515i           | 0.1559                | 0.0955               | 0.0337                      |
| 4+2i             | -0.43302+1.16667i           | 0.1675                | 0.1971               | 0.0358                      |
| -5i              | 0.56340-0.21258i            | 0.1691                | 0.5974               | 0.0449                      |
| -7               | -0.68143+0.72467i           | 0.1939                | 0.6239               | 0.0464                      |
| i                | -0.67024+0.23777i           | 0.2726                | 0.7419               | 0.1612                      |
| 2+8i             | -1.19293+0.33327i           | 0.4453                | 0.9601               | 0.5237                      |
| 2                | -0.77633-0.08160i           | 1.0893                | 1.8591               | 0.6895                      |
| -7               | -0.89380-0.36960i           | 1.7588                | 1.9249               | 0.7331                      |
| 8-2i             | -0.75081-0.62499i           | 2.3110                | 4.5950               | 0.9291                      |
| 6                | -0.49219-0.87189i           | 2.3832                | 5.4118               | 1.7956                      |
| 5+4i             | 0.02587-1.00949i            | 2.7716                | 12.3385              | 1.7994                      |
| 2                | 0.12709-1.15164i            | 3.8264                | 20.7626              | 6.8793                      |
| 3                | 0.61401-0.77163i            | 4.1431                | 39.8159              | 10.1299                     |
| -1+i             | 1.17894-0.49591i            | 4.9770                | 48.3157              | 10.3193                     |
| -6i              | 0.99621-0.29716i            | 10.0486               | 7.6034×10^{12}       | 3.9375×10^{10}              |
Table II

| N  | 1  | 17 | 19 | 21 | 30 | 34 | 42 | 50 | 59 | 66 | 71 | 72 | 88 | 97 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| J  | 17 | 4  | 4  | 3  | 3  | 4  | 3  | 3  | 3  | 2  | 3  | 3  | 3  | 3  |
| K  | 340| 1360| 1520| 1260| 1800| 2720| 2520| 3000| 3540| 3960| 2840| 4320| 5280| 5820|

Table III

| N  | r   | J  | K    | Min (10^{-16}) | Max (10^{-13}) | Average (10^{-14}) |
|----|-----|----|------|----------------|----------------|---------------------|
| 1  | 1.00| 17 | 340  | 3.6649        | 1.0049        | 1.8427               |
| 2  | 0.60| 15 | 600  | 7.4378        | 1.6218        | 2.4595               |
| 3  | 1.00| 4  | 240  | 7.3648        | 1.0049        | 1.9200               |
| 5  | 0.96| 4  | 400  | 3.6649        | 1.6218        | 2.1839               |
| 10 | 0.80| 4  | 800  | 3.6649        | 1.0049        | 1.9894               |
| 15 | 0.40| 5  | 1500 | 3.6649        | 1.0049        | 2.1596               |
| 20 | 0.80| 4  | 1600 | 3.6649        | 1.0049        | 2.0259               |
| 50 | 0.80| 3  | 3000 | 3.6649        | 1.0049        | 1.9706               |
| 100| 0.80| 4  | 8000 | 3.6649        | 1.0049        | 2.1408               |
| 1000|0.80| 3  | 60000| 3.6649       | 1.0049        | 2.0380               |

Table IV

| N  | 4  | 18 | 24 | 26 | 30 | 31 | 33 | 38 | 48 | 51 | 55 | 56 | 57 | 58 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| J  | 6  | 4  | 3  | 4  | 3  | 4  | 3  | 4  | 3  | 2  | 2  | 2  | 2  |    |
| K  | 480| 1440 | 1440| 2080| 1800| 1860| 2640| 2280| 3840| 3060| 2200| 2240| 2280| 2320|
Polynomial root-finding scheme “Khandug”

Define: \( P(z) = z^N + a_N z^{N-1} + a_{N-1} z^{N-2} + \ldots + a_2 z + a_0 \)

Input:
1. The complex coefficients \( \{a_n\} \), which could be read out from a file.
2. The number of iterations for each root: \( N \)
3. The maximum number of iteration rounds: \( J_{\text{max}} \)
4. The initial radius of the spiral: \( r_{\text{in}} \)
5. The final radius of the spiral: \( r_{\text{fin}} \)

Start Khandug

\( n \) = the number of elements of the set \( \{a_n\} \).

If \( n = 1 \) then:
   - Return \([-a_0]\)
   - Exit routine
End If

If \( n = 2 \) then:
   - Return \( \left[ \frac{1}{2} \left( -a_0 \pm \sqrt{a_0^2 - 4a_1} \right) \right] \)
   - Exit routine
End If

\( \varepsilon = 1 \)
\( J = 0 \)
For \( m = 0, 1, \ldots, n-1 \)
   \( z_m = \left[ \frac{T_{u,v} + \frac{m}{n} \left(T_{u,v} - T_{u,m} \right)} {2} \right] e^{2\pi i n/m} \)
   \( T_{u,v} = |P(z_m)| \)
End For
\( i = \text{Average}(T) \)
While \( 
\varepsilon > 0
\)
   For \( i = 0, 1, \ldots, n-1 \)
      \( R = z_i \)
      For \( k = 1, 2, \ldots, N \)
         \( F = P(R) \)
         \( G = \prod_{j=1}^{n} (R - z_j) \quad , i = 0 \)
         \( G = \prod_{j=2}^{n} (R - z_j) \quad , i = n-1 \)
         \( R \leftarrow R \frac{F}{G} \quad \text{otherwise} \)
      End For
      \( z_i = R \)
      \( S_i = |P(R)| \)
   End For
   \( s = \text{Average}(S) \)
   \( J \leftarrow J + 1 \)
   \( \varepsilon \leftarrow |s - i| \)
   Exit While loop if \( \varepsilon = 0 \) or \( J \geq J_{\text{max}} \)
\( t \leftarrow s \)
End While

If \( \varepsilon = 0 \) then:
   - Return \( \left[ \{z_n\}^{-1}_{i=1} , J \right] \)
   - Exit routine
End if

Return error(“No Convergence: Change \( N \), \( J_{\text{max}} \), \( r_{\text{in}} \), and/or \( r_{\text{fin}} \)"

End Khandug

---

Fig. 1
