
PERGAMON Computers and Mathematics with Applications 39 (2000) 95–105

An International Journal
computers & mathematics
with applications

www.elsevier.nl/locate/camwa

# An Algebraic Algorithm to Isolate Complex Polynomial Zeros Using Sturm Sequences

M. A. O. CAMARGO BRUNETTO
Departamento de Computação, UEL
Londrina, PR, 86051-970, Brazil

D. M. CLAUDIO
Instituto de Matemática, PUC-RS
Porto Alegre, RS, 90610-001, Brazil

V. TREVISAN
Instituto de Matemática, UFRGS
Porto Alegre, RS, 91509-900, Brazil

*(Received August 1998; revised and accepted August 1999)*

**Abstract**—In this paper, algorithms to enumerate and isolate complex polynomial roots are developed, analyzed, and implemented. We modified an algorithm due to Wilf, in which Sturm sequences and the principle of argument are used, by employing algebraic methods, aiming to enumerate zeros inside a rectangle in an exact way. Several improvements are introduced, such as dealing with zeros on the boundary of the rectangle. The performance of this new algorithm is evaluated in a theoretical as well as from a practical point of view, by means of experimental tests. The robustness of the algorithm is verified by using tests with ill-conditioned polynomials. We also compare the performance of this algorithm with the results of a recent paper, using different polynomial classes. 

**Keywords**—Polynomial zeros, Sturm sequences, Algebraic algorithm.

## 1. INTRODUCTION

There are several efficient numerical methods to compute complex polynomial roots, but they are sensitive to computational errors due to the use of a floating-pointing arithmetic. Alternative solutions have been proposed to minimize computational errors, such as, for example, inclusion methods, by means of intervals [1]. Most of the known methods to approximate polynomial roots require previous knowledge of isolated regions for zeros, that is, a set of disjoint regions (rectangles, for example), each one containing a single root of the given polynomial. The problem of isolating complex polynomial roots is closely related to the problem of enumerating polynomial roots. Often, the algorithms for isolating polynomial zeros require a procedure to count the number of roots inside a given region (enumeration). Next, the region will be bisected recursively, until the zeros are isolated. The resulting value of this process is an integer and numerical methods may fail to produce the right number.

Pinkert [2] provided an exact method for locating complex polynomial zeros, based on Routh's Theorem, which determines the number of polynomial zeros lying in the upper half-plane. In

Work herein is partially supported by CAPES and CNPq.

0898-1221/00/$ - see front matter  Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX
PII: S0898-1221(99)00336-3

order to determine the number of zeros inside a rectangle, he used Sturm sequences, with suitable polynomial transformations (like rotation and root squaring). Another method was proposed by Wilf [3], with a different approach. First, he solves the problem of enumeration by means of Sturm sequences and the principle of argument. Then, in order to isolate the roots, he applies a bisection procedure.

A problem that appears with Wilf's procedure is that the method requires the nonexistence of zeros on the boundary of the rectangle. If we are searching polynomial zeros in arbitrary rectangles, how to detect this condition? And still, how do we proceed if there is any zero on the boundary? The way proposed by Wilf may lead to errors due to the use of operations with floating-point numbers. If we work with a conventional floating-point system, what happens with the zeros near the boundaries? They may be taken as zeros on the boundary.

Recently, Collins and Krandick [4] proposed an algorithm also based on the principle of argument. Their approach avoids the use of Sturm sequences, requiring real zero isolation of transformed polynomials instead.

We present here an algorithm that is based on Wilf's method, but uses an algebraic approach. Furthermore, several improvements are added to the original algorithm, such as dealing with zeros on the rectangle boundary in an exact way. Optimization procedures are included to avoid unnecessary searches, and for special classes of polynomials, suitable subalgorithms are considered.

The new algorithm developed [5] results in an efficient computer implementation that shows competitiveness with existing algorithms.

A computer program was developed in the Computer Algebra System Macsyma, and several tests were made on a Sun Sparcstation under Unix. The development of the algorithm as well as the basic data structure are presented in Section 3. Section 4 discusses the improvements introduced to optimize the algorithm performance. Complexity analysis with theoretical and practical results are presented in Section 5. Conclusions are in Section 6.

## 2. MATHEMATICAL BASIS

The principle of argument is perhaps the most known result used to count equation roots inside a close region.

THEOREM 2.1. PRINCIPLE OF ARGUMENT. *Let $R$ be a closed curve with boundary $\delta$ $R$, e $p(z)$ a nonzero polynomial over $\delta$ $R$. Let $\Delta_{\delta R} \arg p(z)$ be the change in the continuous function $\arg p(z)$ when $z$ traverses $\delta$ $R$ in the counterclockwise direction. Then, the number $N$ of zeros of $p(z)$ inside $R$, considering their multiplicities is given by*

$$N = \frac{1}{2\Pi} \Delta_{\delta R} \arg p(z).$$

This value is an integer number and represents the winding number of image $p(z)$ gives around the origin when $z$ traverses $\delta R$ in the counterclockwise direction. For any argument function, $\tan \arg p(z) = (\Im p(z))/(\Re p(z))$, where $\Im p(z)$ is the imaginary part and $\Re p(z)$ is the real part of $p(z)$. The changes in $\arg p(z)$ can be obtained, counting the jumps at $(\Im\, p(z))/(\Re\, p(z))$ when $z$ traverses $\delta R$.

When $p(z)$ crosses the imaginary axis in a counterclockwise direction, $\tan \arg p(z)$ jumps from $+\infty$ to $-\infty$ and if this cross is in clockwise direction, the jump is from $-\infty$ to $+\infty$. The counting of these jumps can be made using Cauchy index.

DEFINITION 2.1. CAUCHY INDEX. *Let $a/b$ be a rational real function and $[\alpha, \beta]$ a real interval. The Cauchy index, denoted by $I_\alpha^\beta p/q$ is given by the difference between the number of points $\in [\alpha, \beta]$, when $p/q$ goes from $-\infty$ to $+\infty$ and the number of points when $p/q$ goes from $+\infty$ to $-\infty$.*

Cauchy index and principle of argument are linked through the following theorem (see [6]).

THEOREM 2.2. *Let $\delta R : z = z(x)$, $\alpha \leq x \leq B$ be a closed curve and $p$ is a polynomial defined over $\delta R$, then*

$$\Delta_{\delta R} \arg p(z) = -\pi I_{\alpha}^{\beta} \frac{\Im p}{\Re p}.$$

The Cauchy index can be computed by using Sturm sequences.

DEFINITION 2.2. STURM SEQUENCE. *A sequence $f_0, f_1, \ldots, f_m$ of real polynomials is a Sturm sequence for a real interval $[a, b]$ if the following conditions are satisfied.*

- *$f_m$ does not change the sign in $[a, b]$.*
- *For each $k$, so that $(1 \leq k \leq m-1)$ and each $x^* \in [a, b]$ of $f_k(x)$, $f_{k+1}(x^*)$, and $f_{k-1}(x^*) < 0$.*

The next theorem is the tool required to compute the Cauchy indeces (see [7]).

THEOREM 2.3. STURM THEOREM. *Let $f_0, f_1, \ldots, f_m$ be a Sturm sequence for $[a, b]$ and let $V(x)$ be the net sign variations through this sequence at the point $x$. Then*

$$I_a^b \frac{f_1(x)}{f_0(x)} = V(a) - V(b).$$

## 3. THE MODIFIED WILF ALGORITHM

In this section, we modify Wilf's procedure in order to produce exact numbers for the zeros inside a rectangle. The main idea of Wilf is to reuse Sturm sequences so that in each level, only two new sequences need to be computed. We also develop a method to detect and count the number of zeros on the rectangle boundary.

Let $R$ be a rectangle, with vertices $Q_1, Q_2, Q_3, Q_4$. If we enumerate the sides $S_i$, by traversing $R$ beginning at $Q_1$ in counterclockwise direction, we have $S_1 : \overline{Q_1Q_2}$, $S_2 : \overline{Q_2Q_3}$, $S_3 : \overline{Q_3Q_4}$, $S_4 : \overline{Q_4Q_1}$.

Each side of $R$ must be mapped into the $x$-axis, with the lower bound at the origin. So, we have for each side $j$ of the rectangle, $Pm_j = p(t_j(z))$, $1 \leq j \leq 4$, where

$$t_1 = Q_1 + z, \qquad t_2 = Q_2 + iz, \qquad t_3 = Q_3 - z, \qquad t_4 = Q_4 - iz.$$

Figure 1 shows the rectangle bisected, with the four subrectangles (I, II, III, IV), and the four vertices (Q1, Q2, Q3, Q4). Each side is associated with a Sturm sequence. The four basic Sturm sequences correspond to the four sides of the rectangle, and are denoted, respectively, by ST1, ST2, ST3, and ST4. The two additional Sturm sequences are associated with the inner line segments that compound the four subrectangles, beginning in the center of the rectangle. These sequences are denoted by ST5 (for the horizontal line) and ST6 (for the vertical line).

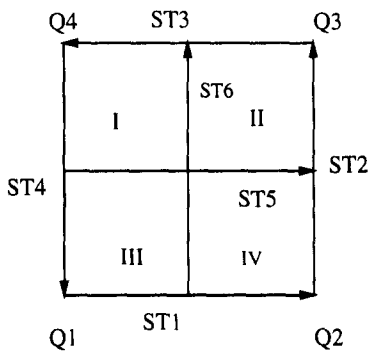


Figure 1. The rectangle and Sturm sequences.

The formula for counting the number of zeros inside a rectangle $R$ is given by

$$N = \frac{1}{2}\sum_{j=1}^{4}(V(S_j) - V(0)),$$

where $V(x)$ denotes the number of sign variation resulting from the evaluation of $x$ in all terms of the Sturm sequence associated.

This formula was obtained by the principle of argument and the Sturm theorem [3,6]. See more details in [5].

It is clear that for each subrectangle, it is possible to use the Sturm sequences of its ascendent rectangle, modifying the upper and lower endpoints of the associated interval, obtained by mapping the side of a rectangle.

In this way, at each bisection, the four sequences previously generated are used and it is necessary to generate only two new Sturm sequences. Given the six Sturm sequences, we can combine four of them for each subrectangle. So we have, for each subrectangle, the following combination:

I. ST5, ST6, ST3, ST4;
II. ST5, ST2, ST3, ST6;
III. ST1, ST6, ST5, ST4;
IV. ST1, ST2, ST5, ST6.

These associations do not change with the search level, while the interval associated with each Sturm sequence depends not only on the level, but also on the point that originated the rectangle and the previous level. Table 1 shows an example of the intervals considered in the first level of bisection, where $L$ is the length of the side of the current rectangle being searched.

Table 1. Intervals for the first level of bisection.

| Side | Subrectangles | | | |
|---|---|---|---|---|
| | I | II | III | IV |
| 1 | [−L,0] | [0,L] | [0,L] | [L, 2L] |
| 2 | [0, L] | [L, 2L] | [−L,0] | [0, L] |
| 3 | [L, 2L] | [0,L] | [0,−L] | [L,0] |
| 4 | [0, L] | [L, 0] | [L, 2L] | [0, −L] |

A brief comment should be made about Sturm sequences. It is known that these sequences may generate expressions with very large coefficients. In fact, it is known that the coefficient growth is exponential on the degree of $f$. There are two alternative ways to overcome this problem. Taking the primitive part of the terms in the sequence, the resulting polynomials are called *primitive Sturm sequences*, which assures that the coefficients generated are always integers and not large. Despite the additional cost to obtain the primitive part (that requires gcd computation), the total cost to generate the sequence decreases, since the arithmetic with integer is faster than with rational numbers.

The algorithm for isolating the roots of a square-free polynomial $p$ with integer coefficients is described by the procedure **Isolate** (Figure 2), which calls the procedure **Search** (Figure 3). The procedure **Search** does a recursive call and also uses the procedure **Count** (Figure 4).

Another method that divides each element of the sequence by a previously known factor, avoiding gcd computation, is a process called *subresultant algorithm* [8]. This alternative keeps the coefficient growth only slightly bigger than linear.

### 3.1. The Presence of Zeros on the Boundary

A typical problem that appears in using this process is the existence of zeros on the lines that divide a subrectangle.

**Procedure Isolate**
**Input**: $p$;
**Output**: $L$, a list of the rectangles or intervals isolating the roots.

1. $b := 2\max_{1\leq i\leq n} |a_{n-i}/a_n|^{1/i}$ a root bound of $p$. (See [8].)

2. $R$= initial rectangle with center at $(0,0)$ and side $2b$

3. for $j := 1\ldots,4$ do {For each side of $R$}

4. (a) Obtain $P_j$

   (b) $\alpha_j :=$ real part of $P_j$

   (c) $\beta_j :=$ imaginary part of $P_j$

   (d) $ST_j :=$ Sturm sequence with $f_0 = \alpha_j$ and $f_1 = \beta_j$
   {using Euclidean algorithm }

   (e) Compute the number of zeros inside $R$ {using formula (1)}

5. Return L = **Search(R,STL,F)**

Figure 2. The procedure **Isolate**.

**Procedure Search**
**Input**: A rectangle (R); a list with the four Sturm sequences (STL) and the level of the previous rectangle (R) {for the first time, this value is 0 }.
**Output**: A list with the identification of the isolated rectangles. Each rectangle is identified by its upper-left corner and its side length.

1. $z_c :=$ the center of R

2. $P_5 := p(z + z_c)$ $P_6 := p(iz + z_c)$

3. Generate Sturm sequences for this 2 transformed polynomials

4. for $k := 1\ldots,4$ do {for each subrectangle $R_k$}

   (a) $N :=$ **count**($R_k$,$STL_k$)

   (b) if $N = 1$ then $L_k = R_k$
   else $L_k =$ **search**($R_k$,$STL_k$,$F_k$)

5. Return concat($L_1, L_2, L_3, L_4$)

Figure 3. The procedure **Search**.

Formula (1) can be used to count zeros inside a rectangle if there are no zeros of $p(z)$ on the boundary of $R$. But, if it occurs, what do we do? In this case, we must have a special treatment. First, it must be verified whether there are zeros on the boundary. How to do this? The suggestion proposed by [3] was the following. As the right side of expression (1) is an integer, so the resultant term of that sum must be an even number. If that does not occur, it possibly due to a loss of significant digits in the computation. In each search level, the four subrectangles must

**Procedure Count(Q,ST)**
**Input**: Rectangle Q and the four Sturm sequences associated with each side of Q.
**Output**: Number of zeros inside Q, and the number of zeros on Q.

1. Identify the Sturm sequences that will be used (according to the association described before)
2. For $j : 1 \ldots 4$ do {for each side of Q}

   (a) identify the extremes of each interval (based on the suitable shifted endpoints, according to the specific Sturm sequence used)

   (b) Compute $V(S_j) - V(0))$

3. Return $1/2 \sum_{j=1}^{4} (V(S_j) - V(0))$

Figure 4. The procedure **Count**.

be checked for parity. If at least one subrectangle has that sum as an odd integer, it is assumed that there was loss of precision, likely by the presence of a zero of $p(z)$ on the boundary of the subrectangle or near it. The center of the father subrectangle is shifted and the four new sums are tested, repeating this process three times, and if still fails, return that rectangle with the number of zeros obtained. It is important to note that this approach, despite being interesting, may cause confusion about the zeros actually on the boundary with zeros close to the boundary.

In this work, we adopt a different strategy, that will be described in the following. For each side $j$, when the Sturm sequence is generated, the last term is (not considering the sign) the greatest common divisor (g.c.d.) of the polynomials $(\alpha_j, \beta_j)$, that we call $h_j$. If $h_j$ is a constant, then there are no zeros on the side $j$. If, on the other hand, $h_j$ has degree 1 or higher, there may be zeros on the side, but not necessarily. If any zero lies on the side $j$, it means that $Pm_j(z)$ has real zeros in the interval $[o, e]$. As $h_j = \gcd(\alpha_j, \beta_j)$ and $h_j$ is not constant, if $z_0$ is a zero of $h_j$, it will be a zero of $\alpha_j$ and $\beta_j$ simultaneously. In order to verify whether $h_j$ has zeros, we use the Sturm sequence for the real case, starting with $f_0 = h_j$ and $f_1 = h_j'$ (first derivative of $h_j$). If $h_j$ does not contain zeros, the procedure continues as in the case where $h_j$ is constant. In the opposite case, the associated side is assigned with its number of zeros (we call it $b_j$).

As we need a boundary free of zeros, we must divide $Pm_j(z)$ by $h_j$, in order to eliminate these zeros. So, the Sturm sequence for this side that contains zeros must begin with $f_0 = (\alpha_j / h_j)$ and $f_1 = (\beta_j / h_j)$.

In this way, it is possible to identify how many zeros are in each boundary of the searched rectangle. Formula (1) is then modified as follows:

$$N = \frac{1}{2} \left( \sum_{j=1}^{4} (V(S_j) - V(0)) - \sum_{j=1}^{4} b_j \right). \tag{2}$$

This modification introduced in the original algorithm provides an exact way to count the number of polynomial zeros in a rectangle, which will simplify the next step of isolating the polynomial zeros.

DATA STRUCTURE. The basic data structure used are lists. The input of the search procedure is given by the following items:

- the northwest corner of the rectangle R;
- the length of the side of R;
- a list with the six basic Sturm sequences is kept, from which the four suitable ones will be selected according to the current rectangle. For each Sturm sequence, we have in addition, its level, its origin point, a sublist that is empty if there are no zeros on the associated

boundary, and filled with a Sturm sequence for counting the zeros on that boundary. The output of the search procedure is given by a list of isolated rectangles and a list of isolated intervals (for the zeros found on the boundaries of some subrectangles during the search procedure).

The search procedure is made in depth, providing in this way, a natural recursive algorithm. Each rectangle with $N > 1$ must be bisected and the enumeration must be made in each subrectangle. The subrectangles with no zeros are rejected and those with only one zero are stored in a special list of isolated rectangles.

## 4. IMPROVING THE PERFORMANCE OF THE SEARCH

In order to save time, we introduce some improvements: for each rectangle bisected, when its total number of zeros is obtained, the search is stopped (at that level). Let us see an example. Considering the numeration of subrectangles adopted according to Figure 1, if a rectangle has four zeros distributed so that two zeros are lying in Rectangle I and two others are lying in Subrectangle II, the procedure searches only the first two subrectangles instead of all of them.

For each level, the last subrectangle has an advantage, because we can conclude previously that the number of zeros inside it is the difference between the total of zeros of the previous level rectangle and the total of zeros already found. This rectangle must be investigated, only if its total number of zeros is greater than one.

When the polynomial coefficients are real, the algorithm can be simplified, taking into account several well-known properties about this special class of polynomials. Here, real coefficients means that there are no imaginary numbers in them (actually, the coefficients are assumed to be rational numbers).

First, it is possible to verify if all zeros of the polynomial are reals. For this, a Sturm sequence starting with $p(z)$ and $p'(z)$ is generated. If all leading coefficients are positive, then all zeros of $p(z)$ are real according to [9, p. 176]. In this case, the Sturm algorithm can be applied for the real case.

Another interesting property that can be verified is the stability of a polynomial. A polynomial is stable (or Hurwitz) when all zeros have negative real part. An efficient algorithm to verify this condition can be found in [10]. For stable polynomials, it is necessary to search only the left side of the $y$-axis, that reduces the region to be searched.

Still, if the polynomial with real coefficients has complex zeros, these occur in conjugate pairs. Then the search is performed only in the upper half-plane.

To check the existence of zeros on the $x$-axis, we need only to compute $V(-\infty) - V(\infty)$ for the Sturm sequence starting with $p(z)$ and $p'(z)$. Similarly, the existence of roots on the $y$-axis can be verified, only changing $p(z)$ by $p(iz)$. In this way, unnecessary polynomial evaluation is avoided.

## 5. COMPUTATIONAL ANALYSIS AND NUMERICAL EXAMPLES

Here we analyze the proposed algorithm and also the Collins-Krandick algorithm [4]. Theoretical bounds as well as empirical results are presented. Let us consider the following notation.

$b$ is an upper bound for the polynomial root, that is $b$ is larger than any zero of $p$. We may consider, without loss of generality, that $b$ is a power of 2.

$d$ is the height of $p$, that is the maximum coefficient (in absolute value) of the polynomial $p$. $L(d)$ is the length of $d$, which means the number of digits.

### 5.1. Complexity of the Modified Wilf Algorithm

In order to establish the worst case complexity for the proposed algorithm, the complexity analysis of the subalgorithms will be considered.

To obtain the cost for enumerating zeros inside a rectangle, we must take into account the cost of Sturm sequence generation and the cost of evaluating elements of the sequence at the endpoints of intervals.

Let $n$ be the degree of the polynomial $p$ and $d$ the maximum norm of $p$.

The Sturm sequence generation costs at most

$$O\left(n^4 L(d)^2\right).$$

(here we use the primitive Sturm sequence), see [11].

The cost to evaluate a polynomial at a rational point is given by (see [11])

$$O\left(n^4 L(nd)^2\right).$$

As a Sturm sequence has at most $n+1$ terms, the cost to evaluate a Sturm sequence is given by

$$(n+1)\left(O\left(n^4 L(nd)^2\right)\right),$$

here we take $d$ as the maximum norm of the last term of the Sturm sequence.

So, the total cost of the enumeration is given by

$$O\left(n^5 L(nd)^2\right) + O\left(n^4 L(d)^2\right),$$

which is equivalent to

$$O\left(n^5 L(nd)^2\right),$$

since $L(nd) > L(d)$.

The number of bisections is bounded by $O(nL(nd))$, since the minimum root separation is given by $\text{sep}(p) > (1/2)(e^{1/2} n^{3/2} d)^{-n}$ and $\text{sep}(p)^{-1}$ is bounded by $O(nL(nd))$ [11].

So isolating roots by the modified Wilf algorithm costs at most

$$O(nL(nd)).O\left(n^5 L(nd)^2\right) \qquad \text{or} \qquad O\left(n^6 L(nd)^3\right). \tag{3}$$

### 5.2. Computational Complexity of the Collins-Krandick Algorithm

The main tool of the Collins-Krandick algorithm is the real root isolation by means of the Modified Uspensky algorithm, which costs at most (see [12])

$$O\left(n^5 L(nd)^2\right).$$

After the isolation, the isolating intervals which belong to zeros of different polynomials must be refined by bisection until they are disjoint. The worst case occurs when two equal intervals are as large as the initial interval, and continues overlapping after several bisections, or when all intervals are equal and need to be refined.

Let $\bar{b}$ be the upper root bound defined in [11], that is $2^l$, where $l = \max_{1 \le i \le n} |\log_2 a_{n-i}/a_n|$ and $d$ be taken as the maximum norm of the specialized polynomial.

Let $\epsilon$ be the minimum distance between two points of an interval, in such a way that they may be considered a single point.

The distance between the endpoints of the interval after $k$ bisections is $\bar{b} \cdot 2^{-k}$.

The initial distance between the endpoints of a rectangle segment is given by $2\bar{b}$, so we have $2\bar{b} \le 8d$.

The maximum number of bisections can be established by the relation

$$\frac{8d}{\epsilon} = \frac{2\bar{b}}{\bar{b}.2^{-k}},$$

from which we obtain

$$k = \log_2 \left( \frac{4d}{\epsilon} \right).$$

In the bisection procedure, only half interval is considered, so the number of bisections is bounded by

$$k = \left\lceil \log_2 \left( \frac{4d}{1/2} \epsilon \right) \right\rceil.$$

The constant $\epsilon$ can be roughly estimated by $\mathrm{sep}(p)$ (see [11]). So the number of bisections is bounded by $O(nL(nd))$.

Each refinement requires a bisection followed by polynomial evaluation at the middle point of the bisected interval. Each evaluation costs at most $O(n^4 L(nd)^2)$, and there are $k$ bisections. So, the cost for refining is $O(n(L(nd)).n^4 L(nd)^2)$.

So, enumerating roots by Collins-Krandick costs at most

$$O\left(n^5 L(nd)^2\right) + O\left(n^5 L(nd)^3\right) \qquad \text{or} \qquad 0\left(n^5 L(nd)^3\right).$$

As the number of bisections needed to isolate the roots is given by $O(nL(nd))$ (see [11]), then the cost of the isolation is bounded by

$$O\left(nL(nd)\right) O\left(n^5 L(nd)^3\right) \qquad \text{or} \qquad O\left(n^6 L(nd)^4\right). \tag{4}$$

### 5.3. Empirical Tests and Examples

The proposed algorithm was implemented in the computer algebra system Macsyma, and several tests were executed on a Sun Sparcstation under Unix. We present here a sample of these tests. In order to evaluate the performance of the algorithm, the enumeration phase of the Collins-Krandick algorithm was also implemented. The estimate for the isolation process was made by verifying the time consumed by the operations needed for one level of bisection for both algorithms. In the following tables, **t.Mod.Wilf** denotes the time (in seconds) obtained in the execution of the Modified Wilf algorithm. **t.Collins** denotes the time (in seconds) obtained in the execution of the Collins-Krandick algorithm. **rel.diff** denotes the relative difference between the two measures.

As an illustration of the performance of the algorithm devised, we present some examples of polynomials specially constructed for testing the algorithm.

$p1 : 1000z^{10} - 2500z^9 - 460800z^8 - 9133400z^7 - 50761800z^6 - 88653100z^5 - 53510400z^4$
$- 37313000z^3 - 197170000z^2 - 364800000z - 198000000;$

$p2 : \left(z - \left(1 + \frac{9}{10}i\right)\right) * \left(z - \left(1 + \frac{99}{100}i\right)\right) * \left(z - \left(1 + \frac{999}{1000}i\right)\right) * \left(z - \left(1 + \frac{9999}{10000}i\right)\right);$

$p3 : z^4 - (5 + 3i)z^3 + (6 + 7i)z^2 - (54 - 22i)z + (120 + 90i);$

$p4 : 470832z^3 - 665857(i + 1)z^2 - 1883328iz - 2663428(1 - i);$

$p5 : z^6 - 1.2z^5 + 0.04z^4 - 0.132z^3 = 0.7955z^2 - 1.43576z + 0.89496;$

$p6 : z^9 - 1.1z^8 + 1.12z^7 - 1.232z^6 + 2.084z^5 - 2.2924z^4 + 1.075648z^3 - 1.1832128z^2$
$+ 1.07910544z - 1.187015984;$

$p7 : 13652098z^4 - (38613965 + 38613965i)z^3 + (2i)z^2 + (-154455860 + 154455860i)z$
$+ 218433576.$

The time consumed for enumerating zeros in the initial region for these polynomials is given in Table 2.

Table 2. Time consumed to enumerate zeros of the polynomials.

| Polynomial | t.Mod.Wilf | t.Collins |
|---|---|---|
| p1 | 19.93 | 39.95 |
| p2 | 2.47 | 6.67 |
| p3 | 1.72 | 6.6 |
| p4 | 1.94 | 4.12 |
| p5 | 4.2 | 11.84 |
| p6 | 22.5 | 32.85 |
| p7 | 2.9 | 5.88 |

Several other experiments were made with squarefree polynomials, whose coefficients were randomly generated. With respect to the enumeration process, the following results were obtained. For polynomials with real coefficients, with 10 binary digits, considering degree 3 through 30, the proposed algorithm was faster for polynomials of degree 25 or less. The Collins-Krandick algorithm was faster for polynomials of degree greater than 25. For polynomials with complex coefficients, it was observed that the Collins-Krandick algorithm was faster for polynomials of degree 20 or more. Table 3 shows the observed results.

Table 3. Time consumed to enumerate zeros in rectangles for the modified Wilf algorithm and the Collins-Krandick algorithm.

| Degree | Real Coef. | | | Complex Coef. | | |
|---|---|---|---|---|---|---|
| | t.Mod.Wilf | t.Collins | Rel.Diff. | t.Mod.Wilf | t.Collins | Rel.Diff |
| 3 | 1.17 | 3.45 | 1.95 | 1.98 | 5.13 | 1.59 |
| 4 | 1.67 | 5.62 | 2.36 | 2.27 | 6.58 | 1.89 |
| 5 | 2.28 | 7.18 | 2.15 | 3.92 | 11.85 | 2.02 |
| 6 | 4.12 | 11.42 | 1.77 | 5.88 | 16.30 | 1.77 |
| 7 | 5.43 | 15.70 | 1.89 | 9.08 | 23.32 | 1.56 |
| 10 | 15.23 | 39.37 | 1.58 | 28.25 | 60.78 | 1.15 |
| 12 | 28.17 | 65.48 | 1.32 | 59.42 | 98.08 | 0.65 |
| 15 | 76.12 | 127.58 | 0.67 | 148.12 | 189.22 | 0.28 |
| 18 | 174.30 | 238.10 | 0.36 | 330.80 | 356.52 | 0.07 |
| 19 | 224.57 | 283.22 | 0.26 | 423.10 | 428.88 | 0.01 |
| 20 | 275.40 | 357.08 | 0.29 | 536.68 | 504.43 | −0.06 |
| 21 | 352.75 | 404.88 | 0.15 | 658.97 | 607.33 | −0.07 |
| 22 | 434.38 | 478.75 | 0.10 | 846.58 | 695.35 | −0.18 |
| 25 | 773.50 | 780.75 | −0.03 | 1257.65 | 939.20 | −0.29 |
| 30 | 1804.15 | 1520.60 | −0.16 | – | – | – |

Another interesting result was also obtained. It regards the time consumed as function of the length of the polynomial coefficients. Taking polynomials with fixed degree (10) and varying the length of the coefficients (3, 6, 9, 12, and 15 decimal digits), it was possible to verify that for polynomials with coefficients smaller than 10 decimal digits, the proposed algorithm is faster. On the other hand, the Collins-Krandick algorithm is better for polynomials with bigger coefficients. (See Table 4.)

Table 4. Time consumed (in sec) according to the coefficient length.

| Coefficient Length | t.Mod.Wilf | t.Collins |
|---|---|---|
| 3 | 12.95 | 39.33 |
| 6 | 28.03 | 40.25 |
| 9 | 38.82 | 43.72 |
| 12 | 58.78 | 49.22 |
| 15 | 106.75 | 58.60 |

The average time observed for the implemented algorithms shows that the algorithm devised is more sensitive to the length of the coefficients. Here we note a discordance with the theoretical results found. Perhaps it can be justified because the complexity was established for the worst case, and as Collins and Krandick observed in [4], the time seemed to depend linearly on the coefficient size. In the case of the algorithm proposed, when the polynomial degree increases, the Sturm sequence has more terms, and the coefficients becomes larger. This justifies the low performance for polynomials of high degree. In order to reduce this limitation, we should carry on the research, exploring other alternatives to reduce the coefficient growth in the generation of Sturm sequences.

## 6. CONCLUSIONS

The algorithm presented provides an improved way to isolate complex polynomial zeros. The algebraic approach given to the algorithm, and the treatment given to the zeros on the boundary of a rectangle searched, made it possible to obtain results free of roundoff errors. Some optimization procedures were added in order to deal with special polynomials in a more suitable way. The known properties about its roots were taken into account. According to the case, subalgorithms may be used, speeding up the enumeration process. The program implementation responds quite effective even with ill-conditioned polynomials. The empirical tests applied provided a performance evaluation, showing that the algorithm here developed is competitive to the recent work of Collins-Krandick [4]. According to the experiments, it was verified that for polynomials of degree less than 20, the proposed algorithm is faster. Also, its performance is better for polynomials with small coefficients. Future research should be made in order to improve the performance of the algorithm proposed, mainly with respect to the coefficient growth of the Sturm sequences.

## REFERENCES


1. D.M. Claudio and S.M. Rump, Inclusion methods for real and complex functions in one variable, In *Berichte des Forschungsschwerpunktes Informations und Kommunikationstechnik*, Bericht 92.5, TUHH, Hamburg, (1992).
2. J.R. Pinkert, An exact method for finding the roots of a complex polynomial, *ACM Transactions on Mathematical Software* **2** (4) (December 1976).
3. H.S. Wilf, A global bisection algorithm for computing the zeros of polynomials in the complex plane, *Journal of the ACM* **25** (3) (1978).
4. G.E. Collins and W. Krandick, An efficient algorithm for infallible polynomial complex root isolation, In *Proceedings of ISSAC '92*, pp. 189–194, ACM, (1992).
5. M.A.O. Camargo-Brunetto, Algoritmos algébricos para enumerar e isolar zeros polinomiais complexos. Thesis submitted as requested to the doctor degree in Computer Science. U.F.R.G.S. (1994).
6. M. Marden, The geometry of the zeros of a polynomial in a complex variable, *AMS Mathematical Surveys III* (1949).
7. P. Henrici, *Applied and Computational Complex Analysis*, Volume 1A, Wiley and Sons, (1974).
8. D. Knuth, *The Art of Computer Programming, Volume 2—Seminumerical Algorithms*, Addison-Wesley, (1981).
9. E.J. Barbeau, *Polynomials*, Springer-Verlag, (1989).
10. V. Trevisan, Recognition of Hurwitz polynomials., *SIGSAM Bulletin* **24** (4) (October 1990).
11. G.E. Collins and G.K. Loos, Real zeros of polynomials, In *Computing, Suppl.*, Volume 4, pp. 83–94, Springer-Verlag, (1982).
12. W. Krandick, Isolierung reeler Nullstellen von Polynomen, In *Wissenschaftliches Rechnen*, Akademie-Verlag, Berlin, (1995).