Journal of Mathematics and Statistics 6 (3): 226-232, 2010
ISSN 1549-3644


# The Exact Root Algorithm for Computing the Real Roots of an Nth Degree Polynomial

E.A. Adebile and V.I. Idoko
Department of Mathematical Sciences, Federal University of Technology Akure, Nigeria

**Abstract: Problem statement:** The need to find an efficient and reliable algorithm for computing the exact real roots of the steady-state polynomial encountered in the investigation of temperature profiles in biological tissues during Microwave heating and other similar cases as found in the literature gave rise to this study. **Approach:** The algorithm (simply called ERA-Exact Root Algorithm) adopted polynomial deflation technique and uses Newton-Raphson iterative procedure though with a modified termination rule. A general formula was specified for finding the initial approximation so as to overcome the limitation of local convergence which is inherent in Newton's method. **Results:** A new algorithm for finding the real roots of an nth degree polynomial at a practically low computational cost was obtained. **Conclusion/Recommendations:** ERA is simple, flexible, easy to use and has clear benefits and preferences to a number of existing methods.

**Key words:** Algorithm, computational cost, nth degree polynomial, real roots

## INTRODUCTION

The classical problem of solving an nth degree polynomial equation has substantially influenced the development of mathematics throughout the centuries and still has several important applications to the theory and practice of present-day computing as reported by Pan (1997).

Jenkins-Traub algorithm, a three-stage method for computing the zeros of a polynomial in roughly increasing order of magnitude was presented by Jenkins and Traub (1970). Bairstow's method attempts to find the zeros of real polynomials by searching for pairs of zeros which generate real quadratic factors as reported by Brodlie (1975). Edelman and Murakami (1995) presented a good method for computing the zeros of a polynomial P(x). This method first finds the companion matrix C of P(x) and then computes its eigenvalues knowing that if $\lambda$ is an eigenvalue of C then $\lambda$ is a root of P(x). Thus, finding the eigenvalues of C is equivalent to finding the zeros of P(x).

Bernoulli's method exploits the connection between a linear difference equation and the zeros of its characteristic polynomial in order to find the zeros of a polynomial without knowing crude first approximations. Graeffe's root-squaring method basically replaces the equation:

$$P_n(x) = a_n x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + ...a_3x^3 + a_2x^2 + a_1x + a_0$$

by an equation still of degree n, whose roots are the squares of the roots of $P_n(x)$. By iterating this procedure, the roots of unequal magnitude become widely separated in magnitude. By separating the roots sufficiently, it is possible to calculate the roots directly from the coefficients. Newton's method is a well-known iterative method for approximating the zeros of a polynomial equation. Starting with a given initial approximation $x_0$, a sequence $x_1$, $x_2$, $x_3$, … is computed where $x_{n+1}$ is given by:

$$x_{n+1} = x_n + h_n$$

where, $h_n = P(x_n) \Big/ P'(x_n)$.

The iterative procedure is terminated when $|h_n|$ has become less than the largest error permissible in the root. These were reported in literature The Nth root algorithm though a consequence of Newton's method is a fast converging method for finding the principal nth root $\sqrt[n]{A}$ of a positive real number, A (Wikipedia, 2007).

A comprehensive bibliography on roots of polynomials covering (hopefully) most published works between the "Dawn of history" and 1994 was presented by McNamee (1993). His paper surveyed the twenty-nine existing categories of polynomial root-finding algorithms namely: Bracketting method; Newton's method; Simultaneous root-finding method; Graeffe's method; Integral methods esp. Legendre's; Bernoulli's

**Corresponding Author:** E.A. Adebile, Department of Mathematical Sciences, Federal University of Technology Akure, Nigeria

and QD method; Interpolation methods such as Secant, Muller's; Minimization method; Jenkins-Traub method; Sturm sequences, greatest common divisors, resultants, stability questions, Interval methods; Miscellaneous; Lin and Bairstow's methods; Methods involving derivatives higher than 1st; Complexity, convergence and efficiency questions; Evaluation of polynomials and derivatives; A priori bounds; Low-order polynomials (special methods); Integer and Rational arithmetic; Special cases such as Bessel polynomials; Vincent's methods; Mechanical devices; Acceleration techniques; Existence questions; Error estimates, deflation, sensitivity, continuity; Roots of random polynomials; Relation between roots of a polynomial and those of its derivatives; and Nth roots.

For every one of the standard existing methods listed above, there are some exceptional cases in which the particular method applied fails to work. For example, Newton's method requires a good initial approximation for convergence, Bracketing method requires a previous knowledge of an initial interval guaranteed to contain a root i.e. if a and b are the endpoints of the interval (a, b) , the Graeffe's method is not suitable for polynomials some of whose roots are of equal magnitude and so on .The research on approach for finding and studying the behavior of roots for polynomial equation is still going on vigorously as is evident in literature (San Joe Math Circle, 2009; Wikipedia, 2010; Gatton *et al*., 2007; Goedecker, 1994) (a, b).

In this study, we present a new algorithm for finding the exact real roots of an nth degree polynomial at a practically low computational cost. The need to find an efficient and reliable algorithm for computing the exact real roots of the steady-state polynomial encounter in our previous works and other similar cases as found in the literature gave rise to this study. Our algorithm (simply called ERA-Exact Root Algorithm) adopts polynomial deflation technique and uses Newton-Raphson iterative procedure though with a modified termination rule. We specify a general formula for finding the initial approximation so as to overcome the limitation of local convergence which is inherent in Newton's method. Above all, ERA is insensitive to "clusters" and computes the exact real roots of polynomials (cases of multiple roots inclusive).

**Mathematical formulation:** Given an nth degree polynomial:

$$P_n(x) = c_n x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + ....c_3x^3 + c_2x^2 + c_1x + c_0 \quad (1)$$

where, $n \in Z^+$, $c_i \in R$; i = 1(1)n.

We set:

$$n_i = n\text{-}i+1 \quad (2)$$

Thus, we have:

$$\begin{aligned} n_1 &= n-1+1 = n \\ n_2 &= n-2+1 = n-1 \\ n_3 &= n-3+1 = n-2 \\ &\dots \quad \dots \quad \dots \\ n_n &= n-n+1 = 1 \end{aligned} \quad (3)$$

$$P_{n_1}(x) = P_n(x), P_{n_2}(x) = P_{n-1}(x), P_{n_{31}}(x) = P_{n-2}(x), P_{n_n}(x) = P_1(x) \quad (4)$$

Let $a_1$ be a real root of $P_{n_1}(x)$.

Then from factor theorem, the following holds:

$$P_{n_1}(x) = (x-a_1)P_{n_1-1}(x) \quad (5)$$

where $P_{n_1-1}(x)$ is a polynomial of degree $n_1 - 1$.

Recall from (3) that $n_1 - 1 = n_2$.

Thus, (5) becomes:

$$P_{n_1}(x) = (x-a_1)P_{n_2}(x) \quad (6)$$

Given $a_2$, a real root of $P_{n_2}(x)$ and from (6) we have:

$$P_{n_2}(x) = (x-a_2)P_{n_3}(x) \quad (7)$$

Substituting (7) into (6) yields:

$$P_{n_2}(x) = (x-a_1)\cdot(x-a_2)P_{n_3}(x) \quad (8)$$

It is easy to verify that (5) by extension becomes:

$$P_{n_2}(x) = (x-a_1)\cdot(x-a_2)\cdot(x-a_3)...(x-a_{n_1-1})P_1(x) \quad (9)$$

Recall that $P_1(x)$ is of the form $(x\text{-}a_n)$.

Thus, (9) becomes:

$$P_{n_2}(x) = (x-a_1)\cdot(x-a_2)\cdot(x-a_3)...(x-a_{n_1-1})(x-a_{n_1}) \quad (10)$$

holds if $P_{n_1}(x)$ has $n_1$ real roots; $a_1, a_2, a_3, ..., a_{n_1-1}$ and $a_{n_1}$.

Suppose $P_n(x)$ has k real roots, then we have:

$$P_n(x) = (x-a_1)\cdot(x-a_2)\cdot(x-a_3)...(x-a_{k-1})\cdot (x-a_k)\cdot P_{n-k}(x) \quad (11)$$

where $P_{n-k}(x)$ a factor of $P_n(x)$ has $(n-k)/2$ pairs of complex conjugate roots.

## MATERIALS AND METHODS

**Method of solution:** By substituting $n = n_1$ in (1) we obtain the following:

$$P_{n_1}(x) = c_{n_1}x^{n_1} + c_{n_1-1}x^{n_1-1} + c_{n-2}x^{n-2} + ...c_3x^3 + c_2x^2 + c_1x + c_0 \quad (12)$$

We find a real root $a_1$ of $P_{n_1}(x)$ using Newton-Raphson iterative procedure with a modified termination rule and a predetermined initial approximation as follows:

$$x_{s+1} = x_s - U_{n_1}(x_s) \quad \text{with } x_0 = \sqrt[n_1]{|c_0|} \quad (13)$$

Where:

$$U_{n_1}(x_s) = P_{n_1}(x_s) \Big/ P'_{n_1}(x_s)$$

and

$$P'_{n_1}(x_s) = \frac{dP_{n_1}(x)}{dx} | x = x_s$$

We terminate the iterative procedure if $U_{n_1}(x_{s+1}) = 0$ and then set $a_1 = x_{s+1}$.

From (6) we have:

$$P_{n_1}(x) = (x-a_1)P_{n_2}(x) \quad (14)$$

We deflate the polynomial $P_{n_1}(x)$ using synthetic division to obtain:

$$P_{n_2}(x) = P_{n_1-1}(x) = c_{n_1}x^{n_1-1} + \left(c_{n_1-1} + a_1c_{n_1}\right)x^{n_1-2} + \left(c_{n_1-2} + a_1c_{n_1-1} + a_1^2c_{n_1}\right)x^{n_1-3} + .... \quad (15)$$

We select $r_1 \in Z^+$ and satisfying the inequality $0 \le r_1 \le n_1-1$ and find an expression for the coefficient $c_{r_1}$ of $x^{r_1}$ in $P_{n_1-1}(x)$.

By setting $r_1 = n_1-k_1$ and observing the terms of $P_{n_1-1}(x)$ in (15) it is easy to verify that:

$$c_{r_1} = \sum_{l_1=0}^{n_1-r_1-1} a^{l_1}\cdot c_{l_1+r_1-l_1} \quad (16)$$

Substituting (16) into (15) yields:

$$P_{n_2}(x) = \sum_{r_1=0}^{n_1-1}\left(\sum_{l_1=0}^{n_1-r_1-1} a_1^{l_1}c_{l_1+r_1-l_1}\right)x^{r_1} \quad (17a)$$

and

$$P'_{n_2}(x) = \sum_{r_1=0}^{n_1-1} r_1\left(\sum_{l_1=0}^{n_1-r_1-1} a_1^{l_1}c_{l_1+r_1+1}\right)x^{r_1-1} \quad (17b)$$

where, $c_{l_1+r_1+1}$ is the coefficient of $x^{l_1+r_1+1}$ in $P_{n_1}(x)$.

We again obtain a real root $a_2$ of $P_{n_2}(x)$ following the same procedure as in (13):

$$x_{s+1} = x_s - U_{n_2}(x_s) \quad (18)$$

Where:

$$U_{n_2}(x_s) = P_{n_2}(x_s) \Big/ P'_{n_2}(x_s)$$

and

$$P'_{n_2}(x_s) = \frac{dP_{n_{21}}(x)}{dx} | x = x_s$$

To obtain $x_0$ we set $r_1 = 0$ in (16) and substitute the result in (16):

$$\Rightarrow x_0 = \sqrt[n_2]{\left|\sum_{l_2=0}^{n_1-1} a_1^{l_1}c_{l_1+1}\right|} \quad (19)$$

The iterative procedure is terminated if $U_{n_2}(x_{s+1}) = 0$ and we set $a_2 = x_{s+1}$.

We deflate $P_{n_2}(x)$ following the same steps as above to obtain:

$$P_{n_3}(x) = P_{n_2-1}(x) = c_{n_2}x^{n_2-1} + \left(c_{n_2-1} + a_2c_{n_2}\right)x^{n_2-2} + \left(c_{n_1-2} + a_2c_{n_2-1} + a_2^2c_{n_2}\right)x^{n_2-3} + .... \quad (20)$$

which can be written as:

$$P_{n_3}(x) = \sum_{r_2=0}^{n_2-1}\left(\sum_{l_2=0}^{n_2-r_2-1} a_2^{l_1}c_{l_2+r_2-1}\right)x^{r_2} \quad (21a)$$

where, $c_{l_2+r_2+1}$ is the coefficient of $x^{l_2+r_2+1}$ in $P_{n_2}(x)$ and

$$P'_{n_3}(x) = \sum_{r_2=0}^{n_2-1} r_2\left(\sum_{l_2=0}^{n_2-r_2-1} a_1^{l_2}c_{l_2+r_2+1}\right)x^{r_2} \quad (21b)$$

where, $r_2$ satisfies the inequality $0 \le r_2 \le n_2-1$ and $r_2 = n_2-k_2$.

We perform the iteration to obtain:

$$x_{s+1} = x_s - U_{n_3}(x_s) \text{ with } x_0 = \sqrt[n_3]{\left|\sum_{l_2=0}^{n_2-1} a_2^{l_2}c_{l_2+1}\right|} \quad (22)$$

Where:

$$U_{n_3}(x_s) = P_{n_3}(x_s) \Big/ P'_{n_3}(x_s)$$

and

$$P'_{n_3}(x_s) = \frac{dP_{n_{31}}(x)}{dx} \mid x = x_s$$

The iteration is terminated if $U_{n_3}(x_{s+1}) = 0$ and we set $a_3 = x_{s+1}$.

Generalizing, we have:

$$P_{n_j}(x) = \sum_{r_{hh}=0}^{n_h-1}\left(\sum_{l_h=0}^{n_h-r_h-1} a_h^{l_h}c_{l_h+r_h-1}\right)x^{r_h} \quad (23a)$$

and

$$P'_{n_j}(x) = \sum_{r_h=1}^{n_h-1} r_2\left(\sum_{l_h=0}^{n_h-r_h-1} a_h^{l_h}c_{l_h+r_h+1}\right)x^{r_2} \quad (23b)$$

where, $c_{l_h+r_h+1}$ is the coefficient of $x^{l_h+r_h+1}$ in $P_{n_h}(x)$ and $h = j-1$.

The iterative process becomes:

$$x_{s+1} = x_s - U_{n_j}(x_s) \text{ with } x_0 = \sqrt[n_j]{\left|\sum_{l_h=0}^{n_h-1} a_h^{l_h}c_{l_h+1}\right|} \quad (24)$$

Where:

$$U_{n_j}(x_s) = P_{n_j}(x_s) \Big/ P'_{n_j}(x_s)$$

and

$$P'_{n_j}(x_s) = \frac{dP_{n_{3j}}(x)}{dx} \mid x = x_s$$

and terminates if $U_{n_j}(x_{s+1}) = 0$ and we set $a_j = x_{s+1}$.

Suppose $P_n(x)$ has k real roots; $a_1$, $a_2$, $a_3$,…$a_{k-1}$ and $a_k$ then we have:

$$P_n(x) = (x-a_1)\cdot(x-a_2)\cdot(x-a_3)...(x-a_{k-1})\cdot(x-a_k)\cdot P_{n-k}(x) = 0 \quad (25)$$

where $P_{n-k}(x)$ a factor of $P_n(x)$ has (n-k)/2 complex conjugate roots.

If k = n then (25) becomes:

$$P_n(x) = (x-a_1)\cdot(x-a_2)\cdot(x-a_3)...(x-a_{n-1})\cdot(x-a_n) = 0 \quad (26)$$

and the real roots of $P_n(x)$ are $a_1$, $a_2$, $a_3$,…$a_{n-1}$ and $a_n$.

**The exact root algorithm:**

(I) Input
Degree of the polynomial, N
Coefficients of the polynomial;
$c_0$, $c_1$, $c_2$, $c_3$,…$c_{n-1}$ and $c_n$
Number of iterations, M

(II) Define

$$c(0,n) = c_0, c(0,n) = c_0, c(1,n) = c_1, ..., c(n-1,n) = c_{n-1} \text{ and } c(n,n) = c_n$$

$$P_n(x) = c(n,n)x^n + c(n-1,n)x^{n-1} + ... + c(2,n)x^2 + c(1,n)x + c(0,n)$$

$$P'_n(x) = nc(n,n)x^{n-1} + (n-1)c(n-1,n)x^{n-2} + ... + 2c(2,n)x + c(1,n)$$

$$U_n(x) = P_n(x) \Big/ P'_n(x)$$

(III) Compute Initial approximation

$$x_n = \sqrt[n]{|c(0,n)|}$$

Repeat

$X_{s+1} = x_s - U_n(x_s)$

Until

$U_n(x_{s+1}) = 0$ then set $a_1 = x_{s+1}$

Or

$s = m$ then set $a_1 = 0$

(IV) Deflate $P_n(x)$ using the factor $(x-a_1)$ to obtain the expression

$$P_n(x) = (x - a_1) \cdot P_{n-1}(x)$$

(V) Set The coefficient $c_r$ of $x^r$ in $P_n(x)$ as

$$c_r = \sum_{l=0}^{n-r-1} a^1 \cdot c_{l+r-l_1}$$

The polynomial $P_n(x)$ as

$$P_{n-1}(x) = \sum_{l=0}^{n-1} \left( \sum_{l_2=0}^{n-r-1} a_1^{l_2} c_{l+r-1} \right) x^{r_r}$$

It's first derivative $P'_n(x)$ as:

$$P'_{n-1}(x) = \sum_{l=0}^{n-1} r \left( \sum_{l_2=0}^{n-r-1} a_1^{l_2} c_{l+r+1} \right) x^{r-1}$$

Their ratio $U_n(x)$ as:

$$U_{n-1}(x) = P_{n-1}(x) \Big/ P_{n-1}'(x)$$

(VI) Compute Initial approximation

$$x_0 = \sqrt[n-1]{|c(0,n-1)|}$$

Repeat

$X_{s+1} = x_s - U_{n-1}(x_s)$

Until

$U_{n-1}(x_{s+1}) = 0$ then set $a_2 = x_{s+1}$

Or

$s = m$ then set $a_2 = 0$

(VII) Continue Steps IV, V and VI for n-2, n-3, …, 3, 2 and 1 to compute $a_1$, $a_2$, $a_3$,…$a_{n-1}$ and $a_n$

(VIII) Output Roots of the polynomial; $a_1$, $a_2$, $a_3$,…$a_{n-1}$ and $a_n$

**ERA IN C++ CODE:** We present a model program (in C++ computing language) for executing the Exact Root Algorithm as shown below:

```
#include<iostream.h>
#include<math.h>
int main() {
    int m,n,r,j,i,m1,k1;
    int m2,l1,l2,l,h,k,s1;
    double u;
    double b,b1;
    double *c,*a,*x;
    cout<<"enter value for m ";
  cin>>m;
  cout<<"enter value for n ";
    cin>>n;
c=new double [n,n];
    if(c==0)
      cout<<"eroooor";
    a=new double [n];
    if(a==0)
      cout<<"erooor";
    x=new double [n];
    if(x==0)
      cout<<"erooor";
    for (i=0;i<=n;i++){
  cout<<"enter value for co-efficient of x raised to
power "<<i<<" ";
      cin>>c[n,i];
    }
    int s;
    double d,m0,p1,p,g,g1;
    d=c[n,0]*c[n,0];
    m0=0.5/n;
    x[0]=pow(d,m0);
    s=0; g=0;
  p=c[n,0];
    do {
      s1=s; s=s+1; r=0;
      do {
          r=r+1;
          p1=c[n,r]*pow(x[s1],r);
    g1=r*c[n,r]*pow(x[s1],(r-1));
          p=p+p1; g=g+g1;
      }while(r!=n);
      u=p/g;
      if(u=0)
          break;
      x[s]=x[s1]-u;
    }while(s!=m);
    if(u=0)
      a[1]=x[s1];
    else
      a[1]=0;
    j=1;
    do {
      h=j; j=j+1;
      m1=(n+1)-h;
      m2=(n+1)-j;
```

```
        k=0;
        do {
    k=k1; k1=k1+1;
            r=m1-k1;
            l1=r+1;
            b=c[m1,l1];
            l=0;
            do {
               l=l+1;
               l2=l+r;
    b1=pow(a[h],l)*c[m1,l2];
               b=b+b1;
            }while(l!=k);
            c[m2,r]=b;
        }while(r!=0);
    d=c[m2,0]*c[m2,0];
    m0=0.5/m2;
    x[0]=pow(d,m0);
    s=0; g=0;
  p=c[m2,0];
    do {
       s1=s; s=s+1; r=0;
       do {
            r=r+1;
    p1=c[m2,r]*pow(x[s1],r);
    g1=r*c[m2,r]*pow(x[s1],(r-1));
            p=p+p1; g=g+g1;
       }while(r!=m2);
       u=p/g;
       if(u=0)
            break;
       x[s]=x[s1]-u;
    }while(s!=m);
     if(u=0)
          a[j]=x[s1];
            else
          a[j]=0;
    }while(m2!=0);
    for(i=0;i<=n;++i){
       cout<<a[i];
    }
return 0;
}
```

## RESULTS AND DISCUSSION

This study presents a very efficient and reliable algorithm (simply called ERA-Exact Root Algorithm) for computing the exact real roots of an nth degree polynomial which has the following benefits over the standard existing methods earlier mentioned:

- ERA prescribes a unique formula for deflating an nth degree polynomial to another polynomial of degree (n-1) (Eq. 17a)
- ERA specifies a general formula for computing the initial approximation, $x_0$, which guarantees convergence for each stage of polynomial deflation (Eq. 23a)
- ERA presents a modified termination rule, $U_{n_j}(x_{s+1}) = 0$, which guarantees accuracy in the computation of polynomial roots and also accommodates cases of multiple roots
- Most importantly, ERA is simple, flexible and easy to use

We pick polynomial (of degree 10, say) arbitrarily, use ERA to compute the real roots and display the results as follows:

The roots of the polynomial:

$$X^{10} - 7X^9 - 3X^8 - 64X^7 - 10X^6 - 110X^5 - 85X^4 + 9X^3 - 95X^2 + 10X + 24 = 0$$

are
a[1] = 1.302775638
a[2] = 0.6180339887
a[3] = -0.5615528128
a[4] = -0.7015621187
a[5] = -1.6180339887
a[6] = -2.302775638
a[7] = 5.7015621187
a[8] = 3.561552813
a[9] = 0
a[10] = 0

Note that a[j] = 0, j ε [1, n] implies that a[j] is a complex root of $P_n(x)$.

## CONCLUSION

In conclusion, ERA is good for teaching and research and can be used to solve real life polynomial models.

## REFERENCES

Brodlie, K.W., 1975. On Bairstow's method for solution of polynomial equations. Math. Comput., 29: 816-826. http://www.jstor.org/stable/2005292

Edelman, A. and H. Murakami, 1995. Polynomial roots from companion matrix eigenvalues. Math. Comput., 64: 763-776. http://www.jstor.org/stable/2153450

Gatton, T., A. Datta, P. Dey, J.J. Martinez and C. Ting, 2007. A web-based intelligent tutorial system. http://nucri.nu.edu/WebBasedTutorialSystem.pdf

Goedecker, S., 1994. Remarks on algorithms to find roots of polynomials. SIAM J. Sci. Comp., 15: 1059-1063. DOI: 10.1137/0915064

Jenkins, M.A. and J.F. Traub, 1970. A three-stage algorithm for real polynomials using quadratic iteration. SIAM J. Numer. Anal., 7: 545-566. http://www.jstor.org/pss/2949376

McNamee, J.M., 1993. A bibliography on roots of polynomials. J. Comput. Applied Math., 47: 391-394. DOI: 10.1016/0377-0427(93)90064-I

Pan, V.Y., 1997. Solving a polynomial equation: Some history and recent progress. Soc. Ind. Applied Math., 39: 187-220. http://www.jstor.org/stable/2133107

San Joe Math Circle, 2009. Polynomials II. http://www.sanjosemathcircle.org/handouts/2009-2010/20090912.pdf

Wikipedia, 2007. Root-finding algorithms. http://www.briefphone.com/.../Root-finding+algorithm+-+Wikipedia

Wikipedia, 2010. Root-finding algorithms. http://www.en.wikipedia.org/wiki/Root-finding_algorithm