Solving multiple-root polynomials

Feng Cheng Chang, *IEEE Life Member*

Abstract — **A given polynomial is transformed herein into a rational function. All the roots and multiplicities of the polynomial are then easily obtained from the poles and residues of this rational function, instead of solving for them directly using the original, high-degree multiple-root polynomial. The derived program, using only basic MATLAB built-in routines and existing double precision arithmetic, amazingly gives the expected results for test polynomials of very high degree and multiplicity, even as high as $p(x) = (x + 98.765)^{12345}$.**

Keywords—**Numerical analysis; mathematical programming; polynomial solutions; roots and multiplicities; residues and poles; rational function; partial fraction expansion; greatest common divisor; Euclidean GCD algorithm.**

I. INTRODUCTION

A polynomial, possibly with several multiple roots, is transformed into a rational function. All the roots and multiplicities of the polynomial are then obtained simply by finding the poles and residues of this rational function, instead of directly solving on the original, high-degree multiple-root polynomial. The approach requires computation of the greatest common divisor (GCD). The simple and efficient algorithm by Chang [1] is employed. It requires only simple arithmetic operations without the use of any advanced mathematics.

A code in MATLAB is provided, along with a typical numerical example. This routine, using only existing double precision, gives the expected results for test polynomials of very high degree and multiplicities.

II. FORMULATION

Let a polynomial $p(x)$ of degree N, with $N+1$ coefficients b_i, $i = 0,1,\cdots,N$, be given. The K roots z_k with multiplicities m_k, $k = 1,2,\cdots,K$, are *to be sought*, such that

$$p(x) = \sum_{i=0}^{N} b_i x^{N-i} = \prod_{k=1}^{K} (x - z_k)^{m_k}$$

We are now to find the rational function $r(x)$ or a pair of related polynomials $u(x)$ and $v(x)$ derived from the original polynomial $p(x)$, such that

$$r(x) = \frac{v(x)}{u(x)} = \sum_{k=1}^{K} \frac{m_k}{x - z_k}$$

Then, z_k and m_k are simply the poles and residues of the rational function $r(x)$. In other words, the K desired roots z_k of $p(x)$ are obtained by solving the K-degree simple-root polynomial $u(x) = 0$, instead of directly solving the original N-degree multiple-root polynomial $p(x) = 0$, where

$$u(x) = \prod_{k=1}^{K} (x - z_k)$$

The corresponding multiplicities m_k are simply the partial fraction expansion coefficients of the rational function $r(x) = v(x)/u(x)$:

$$m_k = (x - z_k) \frac{v(x)}{u(x)}\bigg|_{x=z_k} = \frac{v(z_k)}{u'(z_k)}$$

It is noted that the transformation between the original polynomial $p(x)$ and the derived rational function $r(x)$ exists:

$$\frac{d}{dx} \ln p(x) = \frac{d}{dx} \ln \prod_{k=1}^{K} (x - z_k)^{m_k} = \sum_{k=1}^{K} \frac{m_k}{x - z_k} = r(x)$$

and inversely,

$$\exp \int_{-\infty}^{x} dx\, r(x) = \exp \int_{-\infty}^{x} dx \sum_{k=1}^{K} \frac{m_k}{x - z_k} = \prod_{k=1}^{K} (x - z_k)^{m_k} = p(x)$$

It follows from the first relation,

$$r(x) = \frac{v(x)}{u(x)} = \frac{p'(x)}{p(x)}$$

Then

$$u(x) = \frac{p(x)}{g(x)}, \qquad v(x) = \frac{p'(x)}{g(x)}$$

where

$$g(x) = \gcd(p(x), p'(x)) = \prod_{k=1}^{K} (x - z_k)^{m_k - 1}$$

is the greatest common divisor of $p(x)$ and $p'(x)$. A pair of polynomials $u(x)$ and $v(x)$ for the rational function $r(x)$ are thus obtained once the polynomial greatest common divisor $g(x)$ is generated.

Of crucial concern in carrying out the process so far is the polynomial GCD computation. The algorithm "monic polynomial subtraction" developed by Chang [1] is applied here. It is adapted and modified from the longhand division in the classical Euclidean algorithm. All required computations involve only elementary arithmetic operations without the use of any advanced mathematics.

Original manuscript drafted May 4, 2009. Revised April 2, 2010.
The author is with Allwave Corporation, 3860 Del Amo Blvd, # 404, Torrance, California 90503, USA. E-mail: fcchang007@yahoo.com.

III. COMPUTER ROUTINES

The complete program source code listing using only the basic MATLAB built-in routines is herein presented. The input **p** to the program is a coefficient vector of the given polynomial $p(x)$, and the output **Z** is a list of computed root-multiplicity pair's z_k - m_k. The built-in function *roots.m* is used here merely for finding the simple-roots polynomial $u(x)$. The polynomial coefficients can be either real or complex numbers.

A short program is also provided for creating a test polynomial coefficient vector from several powered factors.

To solve a polynomial with multiple roots:

```
function Z = polyroots(p)
% *** Solve multiple-root polymonials ***
%
    ntol = 1.e-3;  ztol = 1.e-8;
    mz = length(p)-max(find(p));
    p0 = p(min(find(p)):max(find(p)));
 if length(p0) < 2, Z = [0,mz]; return, end;
    s = abs(p0(end)/p0(1));
 if s < 1,  p0 = p0(end:-1:1);  end;
    q0 = polyder(p0);
    g1 = p0/p0(1);                    G{1} = g1;
    g2 = q0(1:max(find(q0)))/q0(1); G{2} = g2;
for k = 3:2*length(p0),
    l12 = length(g1)-length(g2);  l21 = -l12;
    g12 = [g2,zeros(1,l12)]-[g1,zeros(1,l21)];
    g12 = g12(min(find(abs(g12)>ztol)) ...
              :max(find(abs(g12)>ztol)));
    ren = norm(g12,inf)/norm(g2,inf);
 if ren < ntol,  break;   end;
 if l12 >= 0,  g1 = g2;   end;
    g2 = g12/g12(1);                  G{k} = g2;
end;                               % celldisp(G);
    g0 = g1;
    u0 = deconv(p0,g0);
    v0 = deconv(q0,g0);
    w0 = polyder(u0);
    z0 = roots(u0);
    m0 = polyval(v0,z0)./polyval(w0,z0);
 if s < 1,  z0 = z0.^-1;   end;
    Z = [z0,round(abs(m0))];
 if mz > 0,  Z = [Z; 0,mz];  end;
```

To create a test polynomial from several factors:

```
function p = polyget(A)
% *** Create poly coef vector from factors ***
%
     p = 1;
 for i = 1:length(A(:,1)),
     q = 1;
 for j = 1:A(i,1),
     q = conv(q,A(i,max(find(A(i,:))):-1:2));
 end;
     p = conv(p,q);
 end;
```

IV. TYPICAL EXAMPLE

Solve the given test polynomial of degree 32 expanded from several powered factors:

$$p(x) = (x^4 - 1)^3 (x^3 - x^2)^2 (x^3 + x - 10)^2 \cdot (x^3 + 5x^2 + 11x + 15)(x^2 - 4x + 3)^2 x$$

i. e.,

$$p(x) = x^{32} - 5x^{31} + 2x^{30} - 6x^{29} + 76x^{28} + 140x^{27} + \cdots\cdots + 65791x^8 - 87555x^7 + 55800x^6 - 13500x^5$$

To run in MATLAB:

```
>> A = [ 3  -1   0   0   0   1;  ...
         2   0   0  -1   1   0;  ...
         2 -10   1   0   1   0;  ...
         1  15  11   5   1   0;  ...
         2   3  -4   1   0   0;  ...
         1   0   1   0   0   0  ];

>> p = polyget(A),
  p =
           1        -5         2        -6
          76       140      -802       954
       -4251     13663    -18740     28472
      -53504     45776      5212    -77580
      185243   -220631    104794     52458
     -193356    248612   -146266      9202
       65791    -87555     55800    -13500
           0         0         0         0
           0

>> Z = polyroots(p),
  Z =
         3.0000                    2
        -3.0000                    1
        -1.0000 + 2.0000i          3
        -1.0000 - 2.0000i          3
         2.0000                    2
        -1.0000                    3
        -0.0000 + 1.0000i          3
        -0.0000 - 1.0000i          3
         1.0000                    7
         0.0000                    5
>>
```

V. CONCLUSION

A short and compact program routine, using only existing double precision, has been presented for the solution of a polynomial with multiple roots. It reveals that the higher are the root multiplicities of the given polynomial, the more efficient this approach becomes. This is contrary to the usual experience that the most difficult part of solving for the roots of a polynomial is calculating those that have high multiplicities [2]. For general root-finding routines, the MATLAB software package *MULTROOT* introduced by Zeng [2] is highly recommended. However, his routine embodies an algorithm that makes use of advanced mathematics.

In comparison against test polynomials, both the routines give spectacularly concordant results for test polynomials of very high degree, such as, $p(x) = (x+1)^{500}$. And the routine now presented retains its root-finding power much, much further, all the way up to $p(x) = (x+98.765)^{12345}$!

However, we are almost sure to find failure with any numerical algorithm, especially if the multiple roots are closely clustered together. For example, the presented routine as well as Zeng's will fail to achieve the expected results even when the test polynomial is as simple as

$p(x) = (x - 1. - .99i)(x - 1. - 1.01i)^7 (x - .99 - 1.i)^2 (x - 1.01 - 1.i)^4$

where all the four multiple roots are clustered around $(1+i)$.

VI. ACKNOWLEDGEMENT

The author wishes to thank Dr. Jan Grzesik of Allwave Corporation, Torrance, California, for his useful discussions and comments on the preparation of the manuscript.

REFERENCES

[1] F. C. Chang, "Factoring a polynomial with multiple-roots," *International Journal of Computational and Mathematical Sciences*, vol. 2, no. 4, Fall 2008, pp. 173-176.

[2] Z. Zeng, "Computing multiple roots of inexact polynomials," *Mathematics of Computation*, vol. 74, 2005, pp.869-903.

ADDITIONAL EXAMPLES

Find the roots and multiplicities from the following test polynomials *expanded*:

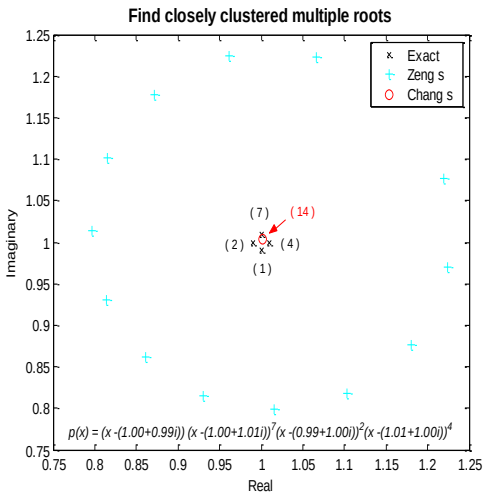
1. $p(z) = (z-1)^{30}(z+2)^{25}(z-3)^{20}(z+4)^{15}(z-5)^{10}(z+6)^5$
2. $p(z) = (z^8 + 2z^7 + 3z^6 + 4z^5 + 5z^4 + 6z^3 + 7z^2 + 8z + 9)^{10}$
3. $p(z) = (z-1.1234)^9 (z+2.5678)^8 (z-3.9123)^7 \cdot (z+4.4567)^6 (z-5.8912)^5 (z+6.3456)^4 \cdot (z-7.7891)^3 (z+8.2345)^2 (z-9.6789)$
4. $p(z) = (z-1+2i)^9 (z+3-4i)^8 (z-5-6i)^7$
5. $p(z) = (z-2.0123-3.4321i)^{30} (z-4.4567+5.8765i)^{25} \cdot (z+6.8901-7.2109i)^{17} (z+8.2345+1.6543i)^9 \cdot (z-9.6789-0.0987i)^4$
6. $p(z) = (506z+1)^{13}(z-987)^{24}$
7. $p(z) = (z^5+1)^{750}$
8. $p(z) = (z^8-1)^{1000}$
9. $p(z) = (z^{100}-1)^{100}$
10. $p(z) = (z^6-1)(z^5-1)^2(z^4-1)^3(z^3-1)^4(z^2-1)^5(z-1)^6$
11. $p(z) = (12345z - 9876)^{70}$
12. $p(z) = (123456789z+1)^{1234}$
13. $p(z) = (z - 123456789)^{987}$
14. $p(z) = (z-(1.00+0.99i))\ (z-(1.00+1.01i)) \cdot (z-(0.99+1.00i))\ (z-(1.01+1.00i))$
15. $p(z) = (z-(1.00+0.99i))\ (z-(1.00+1.01i))^7 \cdot (z-(0.99+1.00i))^2 (z-(1.01+1.00i))^4$

That is

```
>> % Ex. 1
>> r = [ +1*ones(1,30), -2*ones(1,25), ...
>>       +3*ones(1,20), -4*ones(1,15), ...
>>       +5*ones(1,10), -6*ones(1, 5) ];
>> p = poly(r);
>> Z = polyroots(p),
  Z   =
      -6.0000                          5
       5.0000                         10
      -3.9999                         15
      -2.0000                         25
       2.9999                         20
       1.0000                         30

>> % Ex. 2
>> p = polyget([ 10  9 8 7 6 5 4 3 2 1 ]);
>> Z = polyroots(p),
  Z   =
      -1.2887 + 0.4476i               10
      -1.2887 - 0.4476i               10
      -0.7243 + 1.1369i               10
      -0.7243 - 1.1369i               10
       0.1363 + 1.3049i               10
       0.1363 - 1.3049i               10
       0.8767 + 0.8813i               10
       0.8767 - 0.8813i               10

>> % Ex. 3
>> A = [ 9  -1.1234 1; 8 +2.5678  1; ...
>>       7  -3.9123 1; 6 +4.4567  1; ...
>>       5  -5.8912 1; 4 +6.3456  1; ...
>>       3  -7.7891 1; 2 +8.2345  1; ...
>>       1  -9.6789 1 ];
>> p = polyget(A);
>> Z = polyroots(p),
  Z   =
        9.6789                         1
       -8.2345                         2
        7.7891                         3
       -6.3456                         4
        5.8912                         5
       -4.4567                         6
        3.9123                         7
       -2.5678                         8
        1.1234                         9

>> % Ex. 4
>> A = [ 9  -1+2i 1; 8  3-4i 1; 7  -5-6i 1 ];
>> p = polyget(A);
>> Z = polyroots(p),
  Z   =
      5.0000 + 6.0000i                 7
     -3.0000 + 4.0000i                 8
      1.0000 - 2.0000i                 9

>> % Ex. 5
>> r = [ (+2.0123+3.4321i)*ones(1,30), ...
>>       (+4.4567-5.8765i)*ones(1,25), ...
>>       (-6.8901+7.2109i)*ones(1,17), ...
>>       (-8.2345-1.6543i)*ones(1,9), ...
>>       (+9.6789+0.0987i)*ones(1,4) ];
>> p = poly(r);
>> Z = polyroots(p),
  Z   =
      -6.8901 + 7.2109i               17
      -8.2345 - 1.6543i                9
       9.6789 + 0.0987i                4
       4.4567 - 5.8765i               25
       2.0123 + 3.4321i               30

>> % Ex. 6
>> p = polyget([ 24  -987 +1; 13  +1 +506 ]);
>> Z = polyroots(p),
  Z   =
     987.00000                        24
      -0.00197                        13
```

```
>> % Ex. 7
>> p = polyget([ 750  +1 0 0 0 0 +1 ]);
>> Z = polyroots(p),
  Z   =
      -1.0000 + 0.0000i              750
      -0.3090 + 0.9510i              750
      -0.3090 - 0.9510i              750
       0.8090 + 0.5877i              750
       0.8090 - 0.5877i              750

>> % Ex. 8
>> p = polyget([1000 -1 0 0 0 0 0 0 0 +1]);
>> Z = polyroots(p),
  Z   =
      -1.0000 + 0.0000i             1000
      -0.7071 + 0.7071i             1000
      -0.7071 - 0.7071i             1000
       0.0000 + 1.0000i             1000
       0.0000 - 1.0000i             1000
       1.0000 + 0.0000i             2000
       0.7071 + 0.7071i             2000
       0.7071 - 0.7071i             2000

>> % Ex. 9
>> p = polyget([ 100  -1 zeros(1,99) +1 ]);
>> Z = polyroots(p),
  Z   =
       1.0000 + 0.0000i              100
       0.9980 + 0.0627i              100
       0.9980 - 0.0627i              100
       0.9921 + 0.1253i              100

        ..... (92 lines skipped) .....

      -0.0627 + 0.9980i              100
      -0.0627 - 0.9980i              100
       0.0000 + 1.0000i              100
       0.0000 - 1.0000i              100

>> % Ex. 10
>> A = [ 1  -1  0  0  0  0  0  1;  ...
>>       2  -1  0  0  0  0  1  0;  ...
>>       3  -1  0  0  0  1  0  0;  ...
>>       4  -1  0  0  1  0  0  0;  ...
>>       5  -1  0  1  0  0  0  0;  ...
>>       6  -1  1  0  0  0  0  0 ];
>> p = polyget(A),
  p   =
       1    -6    10     6   -29    10    13
      22   -17   -46    38   -34    40    50
     -34   -34   -49    46   -25    50    71
      -2  -116   -62    97   -68    96    36
       0   -36   -96    68   -97    62   116
       2   -71   -50    25   -46    49    34
      34   -50   -40    34   -38    46    17
     -22   -13   -10    29    -6   -10     6
      -1
>> Z = polyroots(p),
  Z   =
        1.0000 + 0.0000i             21
       -1.0000 + 0.0000i              9
       -0.8090 + 0.5878i              2
       -0.8090 - 0.5878i              2
       -0.5000 + 0.8660i              5
       -0.5000 - 0.8660i              5
        0.5000 + 0.8660i              1
        0.5000 - 0.8660i              1
        0.3090 + 0.9511i              2
        0.3090 - 0.9511i              2
        0.0000 + 1.0000i              3
        0.0000 - 1.0000i              3

>> % Ex. 11
>> p = polyget([ 70  -9876  +12345 ]);
>> Z = polyroots(p),
  Z   =
           0.8                        70
```

```
>> % Ex. 12
>> p = polyget([ 1234  +1  +123456789 ]);
>> Z = polyroots(p),
  Z   =
       -8.10000007371e-009          1234

>> % Ex. 13
>> p = poly([ +123456789*ones(1,987) ]);
>> Z = polyroots(p),
  Z   =
      123456789                      987

>> % Ex. 14
>> A = [ 1  -1.00-0.99i  1; ...
>>       1  -1.00-1.01i  1; ...
>>       1  -0.99-1.00i  1; ...
>>       1  -1.01-1.00i  1 ];
>> p = polyget(A);
>> Z = polyroots(p),  % Chang's   (Fail)
  Z =
       1.0000 + 1.0000i                4
>> Zz = multroot(p),  % Zeng's  (Success)
  Zz =
       1.0000 + 1.0100i                1
       1.0100 + 1.0000i                1
       0.9900 + 1.0000i                1
       1.0000 + 0.9900i                1

>> % Ex. 15
>> A = [ 1  -1.00-0.99i  1; ...
>>       7  -1.00-1.01i  1; ...
>>       2  -0.99-1.00i  1; ...
>>       4  -1.01-1.00i  1 ];
>> p = polyget(A);
>> Z = polyroots(p),  % Chang's   (Fail)
  Z =
       1.0014 + 1.0043i               14
>> Zz = multroot(p),  % Zeng's    (Fail)
  Zz =
       1.2195 + 1.0771i                1
       1.1616 + 1.1704i                1
       1.2244 + 0.9696i                1
       1.0662 + 1.2232i                1
       1.1800 + 0.8771i                1
       0.9609 + 1.2238i                1
       1.1035 + 0.8184i                1
       0.8716 + 1.1781i                1
       1.0152 + 0.7988i                1
       0.9308 + 0.8152i                1
       0.8150 + 1.1022i                1
       0.8610 + 0.8613i                1
       0.8138 + 0.9304i                1
       0.7965 + 1.0145i                1
```



Extract from F. C. Chang's "Solving multiple-root polynomials," *IEEE Antennas & Propagation Magazine*, vol. 51, no. 6, Dec 2009, pp. 151-155. 04/02/2010