World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:11, 2008

Factoring a Polynomial with Multiple-Roots

Feng Cheng Chang, *Life Member IEEE*

***Abstract*—**A given polynomial, possibly with multiple roots, is factored into several lower-degree distinct-root polynomials with natural-order-integer powers. All the roots, including multiplicities, of the original polynomial may be obtained by solving these lower-degree distinct-root polynomials, instead of the original high-degree multiple-root polynomial directly.

The approach requires polynomial Greatest Common Divisor (GCD) computation. The very simple and effective process, "Monic polynomial subtractions" converted trickily from "Longhand polynomial divisions" of Euclidean algorithm is employed. It requires only simple elementary arithmetic operations without any advanced mathematics.

Amazingly, the derived routine gives the expected results for the test polynomials of very high degree, such as $p(x)=(x+1)^{1000}$.

***Keywords*—**Polynomial roots, greatest common divisor, Longhand polynomial division, Euclidean GCD Algorithm.

I. Introduction

A given polynomial, possibly with several multiple roots, is factored into several lower-degree distinct-root polynomials with power set of natural-order integers. All of the roots with corresponding multiplicities are then found by individually solving these lower-degree distinct-root polynomials, instead of directly solving the original high-degree multiple-root polynomial. It shows that the more root multiplicities the polynomial has, the more efficient this algorithm becomes. This is contrary to the usual issue that the most difficult part of solving a polynomial is calculating the roots with high multiplicities [1].

The approach requires the greatest common divisor (GCD) computation. The simple and efficient process developed by Chang [2] is applied for polynomial GCD. It requires only simple elementary arithmetic operations such as subtractions and divisions. A MATLAB code is provided, along with a typical numerical example. Amazingly, this routine gives the expected results for test polynomials of very high degree.

II. Formulation

A given polynomial $p(x)$ of degree N with $N+1$ coefficients b_i, $i=0,1,\cdots,N$, expressed in a polynomial coefficient form,

$$p(x)=\sum_{i=0}^{N} b_i x^{N-i}, \qquad b_0=1$$

can always be expressed in a factored form having K distinct roots z_k with corresponding multiplicities m_k, $k=1,2,\cdots,K$,

$$p(x)=\prod_{k=1}^{K}(x-z_k)^{m_k}, \qquad N=\deg(p(x))=\sum_{k=1}^{K} m_k$$

Generally the evaluation of the polynomial coefficients from the roots and the multiplicities is very easy and straightforward. On the contrary, the calculation of the roots and multiplicities from a given polynomial coefficients is very much involved and cumbersome, especially for high degree polynomials with large root multiplicities.

When roots with identical multiplicities are collected together, the polynomial $p(x)$ can be factored as

$$p(x)=\prod_{m=1}^{M}\left(w_m(x)\right)^m, \qquad N=\sum_{m=1}^{M} m\cdot\deg(w_m(x))$$

where factors $w_m(x)$ are polynomials having all distinct roots. Therefore the greatest common divisor $g(x)$ of the given polynomial $p(x)$ and its derivative $p'(x)$ is found to be

$$g(x)=\gcd\left(p(x),p'(x)\right)=\prod_{k=1}^{K}\left(x-z_k\right)^{m_k-1}=\prod_{m=2}^{M}\left(w_m(x)\right)^{m-1}$$

And then

$$u(x)=p(x)/g(x)=\prod_{k=1}^{K}(x-z_k)=\prod_{m=1}^{M} w_m(x),$$

$$K=\deg(u(x))=\sum_{m=1}^{M}\deg(w_m(x))$$

The process to find all the desired M factoring polynomials $w_m(x)$ can therefore be summarized into the following recurrent relation. First perform consecutively the GCD computations:

$$g_m(x)=\gcd\left(g_{m-1}(x),g'_{m-1}(x)\right)=\prod_{j=m}^{M}\left(w_j(x)\right)^{j-(m-1)}$$

$$m=2,3,\cdots$$

by setting $g_1(x)=p(x)$ at the start, and reaching $g_{M+1}(x)=g_{M+2}(x)=\cdots=1$ at the end. Then execute successively the two steps of simple polynomial divisions:

Manuscript created August 28, 2007; revised October 31, 2008.

Feng Cheng Chang is with Allwave Corporation, 3860 Del Amo Blvd, Suite 404, Torrance, CA 90503 USA. (e-mail: fcchang007@yahoo.com).

$$u_m(x) = g_m(x)/g_{m+1}(x) = \prod_{j=m}^{M} w_j(x), \qquad m = 1, 2, \cdots, M+1$$

and

$$w_m(x) = u_m(x)/u_{m+1}(x) = g_m(x)\, g_{m+2}(x)/g_{m+1}^{\,2}(x), \qquad m = 1, 2, \cdots, M$$

All the desired factors $w_m(x)$ can therefore be determined directly from $g_m(x)$.

The individual distinct-root polynomial $w_m(x)$ can be more easily solved than the original multiple-root polynomial $p(x)$ and the distinct-root polynomial $u(x)$. Since the polynomials $w_m(x)$ are defined to have distinct roots, it is not necessary to determine the multiplicities of roots.

III. The Greatest Common Divisor

The crucial concern in carrying out the process for factorization of a given polynomial is the GCD computation. Several numerical methods have been proposed for the derivation of GCD [1,3-7]. Some are based on the classical Euclidean algorithm, while the others are based on the procedures involving Sylvester resultant matrix. Most methods require use of advanced mathematics, such as the singular value decomposition and the least square iteration process.

The simple and efficient recurrent process derived by Chang will be applied here. From the Euclidean GCD algorithm, the longhand polynomial division is expressed as,

$$p_k(x) = p_{k-2}(x) - p_{k-1}(x) q_k(x)$$

where quotient $q_k(x)$ and remainder $p_k(x)$ are obtained from dividing dividend $p_{k-2}(x)$ by divisor $p_{k-1}(x)$. If the quotient $q_k(x)$ in every recurrent process can be converted into a numerical constant or even equal to 1, $q_k(x) = 1$, by making both $p_{k-2}(x)$ and $p_{k-1}(x)$ equal degree and monic, then the longhand polynomial division becomes simply a pair of "monic polynomials subtraction",

$$p_k(x) = p_{k-2}(x) - p_{k-1}(x)$$

The recurrent process of this approach may therefore be summarized as follows:

Given a pair of polynomials $b(x)$ and $a(x)$ of degrees n and m,

$$b(x) = \sum_{j=0}^{n} b_j x^{n-j} = b_c(x)\, x^{n_z}, \quad a(x) = \sum_{j=0}^{m} a_j x^{m-j} = a_c(x)\, x^{m_z}$$

where $b(x)$ and $a(x)$ have n_z and m_z zero trailing coefficients, respectively. And $b_c(x)$ and $a_c(x)$ represent the polynomials without zero trailing coefficients, assuming $n_c \geq m_c$, where $n_c = \deg(b_c(x))$ and $m_c = \deg(a_c(x))$.

Before finding our desired $g(x) = \gcd(b(x), a(x))$, we consider $g_c(x) = \gcd(b_c(x), a_c(x))$ first by setting

$$p_0(x) = b_c(x)/b_0, \qquad p_1(x) = a_c(x) x^{n_c - m_c}/a_0$$

Both polynomials $p_0(x)$ and $p_1(x)$ are now in the same degree and monic. Apply the monic polynomial subtraction consecutively starting from $k = 2$ until $k = K+1$, such that

$$p_{K+1}(x) = 0,$$

Then

$$p_K(x) = \gcd(p_0(x), p_1(x)) = \gcd(b_c(x), a_c(x)\, x^{n_c - m_c}) = \gcd(b_c(x), a_c(x))$$

and finally our expected result is obtained,

$$g(x) = \gcd(b(x), a(x)) = p_K(x)\, x^{\min(n_z, m_z)}$$

It is noted that the complete set of $p_k(x), k = 0, \cdots, K$, referred as "polynomial remainder sequence" (PRS), are also obtained during the recurrent process. The computation of PRS by the presented "Monic polynomial subtraction" is very simple, efficient, and accurate, comparing to the approaches by some other authors [3,6,7]. If the coefficients are all real and rational, we may even get the results by hand calculation.

All computations involve only elementary arithmetic operations without any advanced mathematics. Total numbers of operations are fewer than $2n_c^2$ for computing GCD of polynomials $b(x)$ and $a(x)$.

IV. Computer Routine in MATLAB

A MATLAB realization of the algorithm is presented. The input is a coefficient vector for a given $p(x)$, and the outputs are lists of coefficient vectors of computed $w_m(x)$, $g_m(x)$, and $u_m(x)$. The complete PRS may easily be printed . The input coefficients can be either real or complex numbers.

```
function [W,G,U] = fctpoly(p)
%
%  Factorization of multiple-root polynomial
%   G(k) = gcd(g(k-1),der(g(k-1))), k=1:K+2
%   U(k) = G(k)/G(k+1),             k=1:K+1
%   W(k) = U(k)/U(k+1),             k=1:K
%  by F C Chang    10/01/08
%
      g2 = p/p(1);
  for k = 1:length(p);
      g1 = g2;
      g2 = prsgcd(g1,polyder(g1));
      g3 = prsgcd(g2,polyder(g2));
      u1 = deconv(g1,g2);
      u2 = deconv(g2,g3);
      w1 = deconv(u1,u2);
      G{k} = g1;    U{k} = u1;     W{k} = w1;
    if length(u2) == 1;
      G{k+1} = g2;  G{k+2} = 1;  U{k+1} = u2;
    break;   end;
  end;
```

```
   %  celldisp(G);  celldisp(U);  celldisp(W);

function [g,P] = prsgcd(p,q)
%  ** GCD of a pair of polynomials **
%     From "Polynomial remainder sequence"
%     with "Monic polynomial subtraction"
%      P(k) = P(k-2)-P(k-1),       k=1:K
%     by F C Chang    10/01/08
%
  if length(p) < 2,  g = 1; P{1} = 1;  return,
  end;
       n = length(p)-1;
       m = length(q)-1;
       nc = max(find(abs(p)>1.e-6))-1;
       mc = max(find(abs(q)>1.e-6))-1;
      p2 = p(1:nc+1);
      p3 = q(1:mc+1);
  for k = 1:nc+nc+2;
      p1 = p2/p2(1);
       nz = length(p2)-length(p3);
      p2 = [p3/p3(1),zeros(1,nz)];
      p3 = p1-p2;
      p3 = p3(min(find(abs(p3)>1.e-6)):end);
      P{k} = p1(1:max(find(abs(p1)>1.e-6)));
   if norm(p3,inf)/norm(p1,inf) < 1.e-6;
      P{k+1} = P{k};   break;  end;
  end;
      gc = P{k};
      g = [gc,zeros(1,min(n-nc,m-mc))];
      celldisp(P), g,
```

V. Typical Example

For a test polynomial

$$\begin{aligned} p(x) = {} & x^{32} - 5x^{31} + 2x^{30} - 6x^{29} + 76x^{28} + 140x^{27} - 802x^{26} \\ & + 954x^{25} - 4251x^{24} + 13663x^{23} - 18740x^{22} + 28472x^{21} \\ & - 53504x^{20} + 45776x^{19} + 5212x^{18} - 77580x^{17} + 185243x^{16} \\ & - 220631x^{15} + 104794x^{14} + 52458x^{13} - 193356x^{12} \\ & + 248612x^{11} - 146266x^{10} + 9202x^{9} + 65791x^{8} - 87555x^{7} \\ & + 55800x^{6} - 13500x^{5} + 0x^{4} + 0x^{3} + 0x^{2} + 0x + 0 \end{aligned}$$

we shall get

$$\begin{aligned} p(x) = {} & (x+3)^1(x^2-5x+6)^2 \\ & \cdot(x^5+3x^4+8x^3+8x^2+7x+5)^3(1)^4(x-0)^5(1)^6(x-1)^7 \end{aligned}$$

All roots with multiplicities of the given polynomial can thus be easily determined.

```
>> p = poly([ 1 1 1 1 1 1 1 -1 -1 -1 -1+2i -1+2i
        -1+2i -1-2i -1-2i -1-2i 2 2 3 3 +i +i +i
        -i -i -i -3 0 0 0 0 0 ])

 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

>> [W,G,U] = fctpoly(p);
>> celldisp(G); celldisp(U); celldisp(W);

 G{1} =   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

 G{2} =   1      -5      10      -36      116    -188
        308     -620     694     -214     -496    1348
      -1740     1012      28     -692      929    -605
        150        0       0        0        0
 G{3} =   1      -2       3      -12       22     -16
          2       12     -23       18       -5       0
          0        0
 G{4} =   1      -4       6       -4        1       0
          0
 G{5} =   1      -3       3       -1        0
 G{6} =   1      -2       1
 G{7} =   1      -1
 G{8} =   1
 G{9} =   1

 U{1} =   1       -0       -8      -10      -10       90
          8       10         9      -90        0
 U{2} =   1       -3         1      -13       29        3
         -1       13       -30        0
 U{3} =   1        2         5        0       -1       -2
         -5        0
 U{4} =   1       -1         0
 U{5} =   1       -1         0
 U{6} =   1       -1
 U{7} =   1       -1
 U{8} =   1

 W{1} =     1       3
 W{2} =     1      -5       6
 W{3} =     1       3       8        8       7       5
 W{4} =     1
 W{5} =     1       0
 W{6} =     1
 W{7} =     1      -1
```

VI. Conclusion

A very simple, effective algorithm is derived for factorization of a polynomial with multiple roots. The more multiplicities the polynomial roots have, the more efficient this algorithm will be. This is contrary to the statement that the most difficult part of solving a polynomial is computing its roots with high multiplicities.

The Chang's algorithm [2] for computing polynomial GCD is applied here. It requires only simple and efficient recurrent "monic polynomial subtraction" process. This algorithm may also be used for computing the GCD of multivariate polynomials.

The main objective of this algorithm is the factorization of a given polynomial, and is not for root finding. If a polynomial does not possess any multiple roots, then this algorithm will at least reveal that all roots are distinct, and may be solved by any available zero-finding routines.

For general root-finding routines, the MATLAB software package of *MULTROOT* introduced by Zeng [1] is highly recommended. Its routine however requires algorithm of some advanced mathematics.

For comparison by the test polynomials, both the presented `W = fctpoly(p)` and the Zeng's `Z = multroot(p)` give amazingly the expected results for very high degree polynomials, such as, $p(x) = (x+1)^{500}$, $p(x) = (1234x - 56789)^{60}$,

$p(x) = (x - 123456789)^{30}$. And the presented routine achieves even further up to $p(x) = (x+1)^{1000}$!

REFERENCES

[1] Z. Zeng, "Computing multiple roots of inexact polynomials," Math. Comput, 74 (2005), pp. 869-903.
[2] F.C. Chang, "GCD of two univaiate polynomials by monic polynomial subtractions," submitted to Appl. Math. Comput.
[3] W.S. Brown and J.F. Traub, "On Euclid's algorithm and the theory of subresultants," J. ACM 14 (1) (1967), pp. 128-142.
[4] I.S. Pace and S. Barnett, "Comparison of algorithm for calculation of GCD of polynomials," Int. J. System Scien, 4 (1973), pp. 211-226.
[5] M. Mitrouli and N. Karcanias, "Comutation of the GCD of polynomials using Gaussian transformation and shifting," Int. J. Control, 58 (1993), pp. 211-228.
[6] A. Terui, "Recursive polynomial remainder sequence and its subresultants," J. Algebra, (2008).
[7] C.D. Yan and W.H.Chieng, "Method for finding multiple roots of polynomials," Int. J. Computers & Mathematics with Applications, 51 (2006), pp. 605-620.