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 test polynomials of very high degree, such as \( p(x) = (x + 1)^{100} \).

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 more root multiplicities the polynomial has, the more 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 simple elementary arithmetic operations such as subtractions converted trickily from “Longhand polynomial division, Euclidean GCD Algorithm.

II. FORMULATION

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

\[
p(x) = \sum_{i=0}^{N} b_i x^{i-1}, \quad b_0 = 1
\]

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

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

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

\[
p(x) = \prod_{m=1}^{M} (w_m(x))^n, \quad N = \sum_{m=1}^{M} m \cdot \deg(w_m(x))
\]

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

\[
g(x) = \gcd(p(x), p'(x)) = \prod_{k=1}^{K} (x - z_k) = \prod_{m=1}^{M} (w_m(x))^{m-1}
\]

And then

\[
u(x) = p(x)/g(x) = \prod_{k=1}^{K} (x - z_k) = \prod_{m=1}^{M} w_m(x), \quad K = \deg(u(x)) = \sum_{m=1}^{M} \deg(w_m(x))
\]

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:

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

by setting \( g_1(x) = p(x) \) at the start, and reaching \( g_{M-1}(x) = g_M(x) = \cdots = 1 \) at the end. Then execute successively the two steps of simple polynomial divisions:
Before finding our desired \( g(x) = \gcd(b(x),a(x)) \), we consider \( g(x) = \gcd(b(x),a(x)) \) first by setting

\[
p_k(x) = b(x)/b_0, \quad p_k(x) = a(x)x^{-m}/a_0
\]

Both polynomials \( p_k(x) \) and \( p_{k+1}(x) \) 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

\[
p_k(x) = \gcd(p_k(x),p_{k+1}(x)) = \gcd(b(x),a(x)x^{\min(m_k,m_{k+1})})
\]

and finally our expected result is obtained,

\[
g(x) = \gcd(b(x),a(x)) = p_K(x)x^{\min(m_1,m_2)}
\]

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 \cite{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 \( 2m_a^2 \) 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.

```matlab
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;
```
For a test polynomial
\[
p(x) = x^{32} - 5x^{31} + 2x^{30} - 6x^{29} + 76x^{28} + 140x^{27} - 802x^{26} + 954x^{25} - 4251x^{24} + 13663x^{23} - 18740x^{22} + 28472x^{21} - 53504x^{20} + 45776x^{19} + 5212x^{18} - 77580x^{17} + 185243x^{16} - 220631x^{15} + 104794x^{14} + 52458x^{13} - 193356x^{12} + 248612x^{11} - 146266x^{10} + 9202x^{9} - 65791x^{8} + 87555x^{7} + 55800x^{6} - 13500x^{5} + 929x^{4} - 605x^{3} + 90x^{2} - 30x + 0
\]
we shall get
\[
p(x) = (x + 3)(x^2 - 5x + 6)^2 \cdot (x^4 + 3x^3 + 8x^2 + 7x + 5)(1)(x - 0)(1)(x + 1)(x - 1)
\]
All roots with multiplicities of the given polynomial can thus be easily determined.

For a test polynomial
\[
p(x) = x^{32} - 5x^{31} + 2x^{30} - 6x^{29} + 76x^{28} + 140x^{27} - 802x^{26} + 954x^{25} - 4251x^{24} + 13663x^{23} - 18740x^{22} + 28472x^{21} - 53504x^{20} + 45776x^{19} + 5212x^{18} - 77580x^{17} + 185243x^{16} - 220631x^{15} + 104794x^{14} + 52458x^{13} - 193356x^{12} + 248612x^{11} - 146266x^{10} + 9202x^{9} - 65791x^{8} + 87555x^{7} + 55800x^{6} - 13500x^{5} + 929x^{4} - 605x^{3} + 90x^{2} - 30x + 0
\]
we shall get
\[
p(x) = (x + 3)(x^2 - 5x + 6)^2 \cdot (x^4 + 3x^3 + 8x^2 + 7x + 5)(1)(x - 0)(1)(x + 1)(x - 1)
\]
All roots with multiplicities of the given polynomial can thus be easily determined.

V. TYPICAL EXAMPLE

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 \( W = \text{fctpoly}(p) \) and the Zeng’s \( Z = \text{multroot}(p) \) give amazingly the expected results for very high degree polynomials, such as, \( p(x) = (x + 1)^{30} \), \( p(x) = (1234x - 56789)^{30} \).
\( p(x) = (x - 123456789)^{10} \). And the presented routine achieves even further up to \( p(x) = (x + 1)^{1000} \).

**REFERENCES**


