Research Article

Polynomial GCD Derived through Monic Polynomial Subtractions

Feng Cheng Chang

Allwave Corporation, 3860 Del Amo Boulevard, No. 404, Torrance, CA 90503, USA

Correspondence should be addressed to Feng Cheng Chang, fcchang007@yahoo.com

Received 10 March 2011; Accepted 19 April 2011

Academic Editors: E. Kita and F. Lebon

Copyright © 2011 Feng Cheng Chang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A simple and efficient algorithm for finding the GCD of a pair of univariate polynomials is derived. The process requires only successive subtraction of two monic polynomials. Amazingly, this approach gives the desired GCD for a pair of test polynomials of very high degree, such as $(x + 1)^{1000}$ and its derivative.

1. Introduction

The computation of the greatest common divisor (GCD) of a pair of polynomials is an important issue in computational mathematics and also plays a crucial role in many science and engineering fields. Several numerical methods have been proposed for the derivation of polynomial GCD [1–5]. Some are based on the classical Euclidean algorithm and the others are based on the procedures involving Sylvester resultant matrix. Most of them require algorithm of some advanced mathematics, such as the singular value decomposition and the least square iteration process.

In this paper a simple and effective method to find the GCD of a pair of given polynomials is presented. The process requires only the successive subtraction of two monic polynomials. It is modified purposely from the longhand polynomial divisions employed in the classical Euclidean GCD computation. The entire computations require only simple elementary arithmetic operations, such as division and subtraction.

A MATLAB code is provided, along with a typical numerical example. Amazingly, this routine gives the GCD for a pair of test polynomials of very high degree.

2. Mathematical Formulation

From the classical Euclidean algorithm for computing polynomial GCD, the longhand polynomial division is consecutively employed:

$$p_k(x) = p_{k-2}(x) - p_{k-1}(x)q_k(x), \quad (2.1)$$

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":

$$p_k(x) = p_{k-2}(x) - p_{k-1}(x). \quad (2.2)$$

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 ,

$$\begin{aligned} b(x) &= \sum_{j=0}^n b_j x^{n-j} = b_c(x) x^{n_z}, \\ a(x) &= \sum_{j=0}^m a_j x^{m-j} = a_c(x) x^{m_z}, \end{aligned} \quad (2.3)$$

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 that $n_c \geq m_c$, 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 first consider $g_c(x) = \gcd(b_c(x), a_c(x))$ by setting

$$\begin{aligned} p_0(x) &= \frac{b_c(x)}{b_0}, \\ p_1(x) &= \frac{a_c(x) x^{n_c - m_c}}{a_0}. \end{aligned} \quad (2.4)$$

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

$$p_{K+1}(x) = p_{K-1}(x) - p_K(x) = 0, \quad (2.5)$$

we get

$$\begin{aligned} p_K(x) &= \gcd(p_0(x), p_1(x)) = \gcd(b_c(x), a_c(x) x^{n_c - m_c}) \\ &= \gcd(b_c(x), a_c(x)). \end{aligned} \quad (2.6)$$

```

function g = polygcd(p,q)
% *** Find GCD of two given polynomials ***
% through 'Monic polynomial subtraction'
% by F C Chang 03/03/2011
%
    mz = min(length(p)-max(find(p)),length(q)-max(find(q)));
    p0 = p(min(find(p)):max(find(p)));
    q0 = q(min(find(q)):max(find(q)));
    if min(length(p0),length(q0)) < 2,
        g = [1,zeros(1,mz)]; return, end;
    g1 = p0/p0(1);
    g2 = q0/q0(1);
    for k = 3:length(p0) + length(q0),
        l12 = length(g1)-length(g2); l21 = -l12;
        g3 = [g2,zeros(1,l12)]-[g1, zeros(1,l21)];
        g3 = g3(min(find(abs(g3)>1.e-8)):max(find(abs(g3)>1.e-8)));
        if norm(g3,inf)/norm(g2,inf) < 1.e-3, break; end;
        if length(g1) >= length(g2),
            g1 = g2;
            g2 = g3/g3(1);
        else
            g2 = g1;
            g1 = g3/g3(1);
        end;
        P{k} = g2;
    end;
    g = [g1,zeros(1,mz)];
    % celldisp(P);

```

Algorithm 1

Finally our desired polynomial GCD is found

$$g(x) = \gcd(b(x), a(x)) = p_K(x)x^{\min(n_z, m_z)}. \quad (2.7)$$

It is noted that the complete set of $p_k(x)$, referred to as “polynomial remainder sequence” [3, 4], may also be computed during the recurrent process. The entire computations involve only elementary arithmetic operations without any advanced mathematics. The total numbers of recurrent process for computing GCD of the two given polynomials $b(x)$ and $a(x)$ are amazingly fewer than $n_c + m_c$.

3. Computer Routine in MATLAB

The computer code `polygcd.m` in MATLAB is presented. The inputs **p** and **q** are coefficient vectors of two given polynomials $b(x)$ and $a(x)$, and the output **g** is the desired greatest common divisor $g(x)$. The input data can be either real or complex numbers. See Algorithm 1.

4. Typical Example

If one test polynomial $p(x)$ is derived from

$$p(x) = (x+3)(x^2-5x+6)^2(x^5+3x^4+8x^3+8x^2+7x+5)^3x^5(x-1)^7, \quad (4.1)$$

that is,

$$\begin{aligned}
 p(x) = & x^{32} - 5x^{31} + 2x^{30} - 6x^{29} + 76x^{28} + 140x^{27} - 802x^{26} + 954x^{25} \\
 & - 4251x^{24} + 13663x^{23} - 18740x^{22} + 28472x^{21} - 53504x^{20} \\
 & + 45776x^{19} + 5212x^{18} - 77580x^{17} + 185243x^{16} - 220631x^{15} + 104794x^{14} \\
 & + 52458x^{13} - 193356x^{12} + 248612x^{11} - 146266x^{10} + 9202x^9 \\
 & + 65791x^8 - 87555x^7 + 55800x^6 - 13500x^5 + 0x^4 + 0x^3 + 0x^2 + 0x + 0,
 \end{aligned} \tag{4.2}$$

and the another polynomial $q(x)$ is the derivative of $p(x)$,

$$q(x) = p'(x), \tag{4.3}$$

then, applying the routine `polygcd.m`, the desired GCD of $p(x)$ and $q(x)$ is readily computed:

$$\begin{aligned}
 g(x) = & \gcd(p(x), p'(x)) \\
 = & x^{22} - 5x^{21} + 10x^{20} - 36x^{19} + 116x^{18} - 188x^{17} + 308x^{16} - 620x^{15} + 694x^{14} \\
 & - 214x^{13} - 496x^{12} + 1348x^{11} - 1740x^{10} + 1012x^9 + 28x^8 - 692x^7 \\
 & + 929x^6 - 605x^5 + 150x^4 + 0x^3 \\
 & + 0x^2 + 0x + 0.
 \end{aligned} \tag{4.4}$$

Finally it is fairly easy to check that

$$g(x) = (x^2 - 5x + 6)(x^5 + 3x^4 + 8x^3 + 8x^2 + 7x + 5)^2 x^4 (x - 1)^6. \tag{4.5}$$

See Algorithm 2.

5. Applications

One of the most important applications of polynomial GCD is to find the roots with multiplicities of high-degree multiple-root polynomials [3].

Let a univariate polynomial $p(x)$ of degree N be given, and then the K roots z_k with multiplicities m_k are to be sought, such that

$$p(x) = \sum_{i=0}^N b_i x^{n-i} = \prod_{k=1}^K (x - z_k)^{m_k}. \tag{5.1}$$

```

>>
>> 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];
>> q = polyder(p);
>> g = polygcd(p,q);
>> p, q, g,
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
q =
    32    -155      60     -174     2128     3780    -20852    23850    -102024
  314249 -412280  597912 -1070080  869744  93816 -1318860  2963888 -3309465
  1467116  681954 -2320272  2734732 -1462660  82818  526328 -612885  334800
 -67500      0      0      0      0
g =
    1      -5      10     -36      116    -188      308     -620      694
  -214     -496     1348    -1740     1012      28     -692      929     -605
   150      0      0      0      0
>>

```

Algorithm 2

We are now to find the rational function $r(x)$ or its related polynomials $u(x)$ and $v(x)$, such that

$$r(x) = \frac{u(x)}{v(x)} = \sum_{k=1}^K \frac{m_k}{x - z_k}, \quad (5.2)$$

where

$$u(x) = \frac{p(x)}{g(x)}, \quad v(x) = \frac{p'(x)}{g(x)}, \quad (5.3)$$

and $g(x)$ is the polynomial GCD of $p(x)$ and $p'(x)$:

$$g(x) = \gcd(p(x), p'(x)) = \prod_{k=1}^K (x - z_k)^{m_k - 1}. \quad (5.4)$$

Then, z_k and m_k are simply the poles and residues of the rational function $r(x)$. Therefore, instead of directly solving for z_k and m_k from the original N -degree multiple-root polynomial, we may now easily find the roots z_k from the K -degree simple-root polynomial,

$$u(x) = \prod_{k=1}^K (x - z_k), \quad (5.5)$$

```

function Z = poly_roots(p)
% *** Solve multiple-root polynomials ***
%
    mz = length(p)-max(find(p));
    p0 = p(min(find(p)):max(find(p)));
    q0 = polyder(p0);
    if length(p0) < 2, Z = [0,mz]; return, end;
    g = polygcd(p,q);
    u0 = deconv(p0,g0);
    v0 = deconv(q0,g0);
    w0 = polyder(u0);
    z0 = roots(u0);
    m0 = polyval(v0,z0)./polyval(w0,z0);
    Z = [z0,round(abs(m0))];
    if mz > 0, Z = [Z; 0,mz]; end;

```

Algorithm 3:

and the corresponding multiplicities m_k by the partial fraction expansion coefficients of the rational function $r(x) = v(x)/u(x)$,

$$m_k = (x - z_k) \frac{v(x)}{u(x)} \Big|_{x=z_k} = \frac{v(z_k)}{u'(z_k)}, \quad (5.6)$$

A short program code `poly_roots.m` for solving multiple-root polynomials is presented. It uses only basic built-in MATLAB routines except the function `polygcd.m` for computing polynomial GCD. See Algorithm 3.

6. Conclusion

The very simple and efficient algorithm for the calculation of GCD of a pair of given polynomials is developed. The entire computations involve only simple elementary arithmetic operations. Amazingly, the derived routine gives the desired results for the test polynomials of very high degrees, such as $(x+1)^{1000}$, $(x-123456789)^{30}$, and $(1234x+56789)^{50}$, and their respective derivatives.

One of the most important applications for polynomial GCD is to find the roots with multiplicities of polynomials with high degrees [1–3]. It reveals that the higher the root multiplicities of the given polynomial are, 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 [1].

References

- [1] Z. Zeng, "Computing multiple roots of inexact polynomials," *Mathematics of Computation*, vol. 74, no. 250, pp. 869–903, 2005.
- [2] C. D. Yan and W. H. Chieng, "Method for finding multiple roots of polynomials," *Computers & Mathematics with Applications*, vol. 51, no. 3-4, pp. 605–620, 2006.

- [3] F. C. Chang, "Solving multiple-root polynomials," *IEEE Antennas and Propagation Magazine*, vol. 51, no. 6, pp. 151–155, 2009.
- [4] W. S. Brown and J. F. Traub, "On Euclid's algorithm and the theory of subresultants," *Journal of the ACM*, vol. 14, no. 1, pp. 128–142, 1967.
- [5] A. Terui, "Recursive polynomial remainder sequence and its subresultants," *Journal of Algebra*, vol. 320, no. 2, pp. 633–659, 2008.