Peertechz

Annals of Mathematics and Physics
mathematics@peertechz.us

MATHEMATICS AND PHYSICS GROUP

ANNALS OF Mathematics and Physics OPEN ACCESS JOURNAL

ISSN: 2689-7636 DOI: https://dx.doi.org/10.17352/amp

Research Article

A simple algorithm for GCD of polynomials

Pasquale Nardone[1]* and Giorgio Sonnino[2]

[1]Physics Department, Free University of Brussels, 50 av F. D. Roosevelt, Brussels, 1050, Belgium

[2]Physics Department, International Solvay Institutes for Physics and Chemistry, Belgium

Received: 10 December, 2022
Accepted: 22 December, 2022
Published: 23 December, 2022

***Corresponding author:** Pasquale Nardone, Physics Department, Free University of Brussels, 50 av F. D. Roosevelt, Brussels, 1050, Belgium, Tel: +32475840393; E-mail: Pasquale.nardone@ulb.be

Keywords: Bezout's identity; Polynomial remainder sequence; Resultant; Discriminant

https://www.peertechzpublications.com

Check for updates

Abstract

Based on the Bezout approach we propose a simple algorithm to determine the gcd of two polynomials that don't need division, like the Euclidean algorithm, or determinant calculations, like the Sylvester matrix algorithm. The algorithm needs only n steps for polynomials of degree n. Formal manipulations give the discriminant or the resultant for any degree without needing division or determinant calculation.

Introduction

There exist different approaches to determining the greatest common divisor (gcd) for two polynomials, most of them are based on the Euclid algorithm [1] or matrix manipulation [2,3], or subresultant technics [4]. All these methods require long manipulations and calculations around $O(n^2)$ for polynomials of degree n. Bezout identity could be another approach. It $P_n(x)$ is a polynomial of degree n and $Q_n(x)$ is a polynomial of degree at least n, the Bezout identity says that $gcd(P_n(x),Q_n(x)) = s(x)P_n(x)+t(x)Q_n(x)$ where $t(x)$ and $s(x)$ are polynomials of degree less then n. Finding $s(x)$ and $t(x)$ requires also $O(n^2)$ manipulations. If we know that $P_n(0)\neq 0$ we propose here another approach that uses only a linear combination of $p_n(x)$ and $Q_n(x)$ division by x to decrease the degree of both polynomials by 1.

Method

Let's take two polynomials $P_n(x)$ $Q_n(x)$:

$$P_n(x)=\sum_{k=0}^{n} p_k^{(n)} x^k \quad ; \quad Q_n(x)=\sum_{k=0}^{n} q_k^{(n)} x^k$$

with $p_0^{(n)}\neq 0$ and $p_n^{(n)}\neq 0$. The corresponding list of coefficients is:

$$p_n=\{p_0^{(n)},p_1^{(n)},\cdots,p_{n-1}^{(n)},p_n^{(n)}\} \quad ; \quad q_n=\{q_0^{(n)},q_1^{(n)},\cdots,q_{n-1}^{(n)},q_n^{(n)}\}$$

Let's define $\Delta_n = q_n^{(n)}p_0^{(n)} - p_n^{(n)}q_0^{(n)}$. If $\Delta_n\neq 0$, we can build two new polynomials of degree n-1 by canceling the lowest degree term and the highest degree term:

$$\begin{cases} P_{n-1}(x) = 1/x\left(q_0^{(n)}P_n(x) - p_0^{(n)}Q_n(x)\right) \\ Q_{n-1}(x) = q_n^{(n)}P_n(x) - p_n^{(n)}Q_n(x) \end{cases} \tag{1}$$

If $\Delta_n = 0$ then we replace $Q_n(x)$ it by $\tilde{Q}_n(x)$:

$$\begin{cases} P_n(x) = P_n(x) \\ \tilde{Q}_n(x) = x(p_0^{(n)}Q_n(x) - q_0^{(n)}P_n(x)) \end{cases} \tag{2}$$

This corresponds to the manipulation on the list of coefficients:

$$\text{if } \Delta_n\neq 0 \quad \begin{cases} p_k^{(n-1)} = q_0^{(n)}p_{k+1}^{(n)} - p_0^{(n)}q_{k+1}^{(n)} \\ q_k^{(n-1)} = q_n^{(n)}p_k^{(n)} - p_n^{(n)}q_k^{(n)} \end{cases} \quad k\in[0,n-1] \tag{3}$$

Note also that $p_{n-1}^{(n-1)} = -q_0^{(n-1)} = -\Delta_n$ this will remains true at all iterations ending with $p_0^{(0)} = -q_0^{(0)} = -\Delta_1$.

$$\text{if } \Delta_n = 0 \quad \begin{cases} \tilde{q}_0^{(n)} = 0 \\ \tilde{q}_k^{(n)} = p_0^{(n)}q_{k-1}^{(n)} - q_0^{(n)}p_{k-1}^{(n)} \end{cases} \quad k\in[1,n] \tag{4}$$

Note that the new $\tilde{q}_1^{(n)} = 0$.

Citation: Nardone P, Sonnino G (2022) A simple algorithm for GCD of polynomials. Ann Math Phys 5(2): 193-195 DOI: https://dx.doi.org/10.17352/amp.000065

In terms of list manipulation we have:

$$\text{if } \Delta_n \neq 0 \quad \begin{array}{l} p_{n-1} = Drop[First[\,q_n]\;\,p_n - First[\,p_n]\;\,q_n, 1] \\ q_{n-1} = Drop[Last[\,q_n]\;\,p_n - Last[\,p_n]\;\,q_n, -1] \end{array}$$

where First[list] and Last[list] take the first and the last element of the list respectively, while Drop[list,1] and Drop[list,-1] drop the first and the last element of the list respectively. If $\Delta_n = 0$ then we know that $p_0^{(n)} q_n^{(n)} - q_0^{(n)} p_n^{(n)} = 0$ so the list $p_0^{(n)}\; q_n - q_0^{(n)}\; p_n$ ends with 0 so the list manipulation is :

$$\tilde{q}_n = RotateRight[First[\,p_n]\, q_n - First[\,q_n]\, p_n]$$

where RotateRight[list] rotate the list to the right (RotateRight[{a,b,c}]={c,a,b}).

So we have the same Bezout argument, they $gcd(P_n(x), Q_n(x))$. must divide $P_{n-1}(x)$ and $Q_{n-1}(x)$ or $P_n(x)$ and $\tilde{Q}_n(x)$. Repeating k times the iteration, it must divide $P_{n-1}(x)$ and $Q_{n-1}(x)$.

If we reach a constant: $P_0(x) = p_0^{(0)}$ and $Q_0(x) = q_0^{(0)} = -p_0^{(0)}$ then $gcd(P_n(x), Q_n(x)) = 1$. If we reach, some stage j of iteration, $P_{n-j}(x) = 0$ or $Q_{n-j}(x) = 0$ then the previous stage j-1 contains the gcd.

Repeating these steps decrease the degree of polynomials. Reversing the process enables us to find a combination of $P_n(x)$ and $Q_n(x)$ which gives a monomial x^k and the polynomials are co-prime, or we reach a 0-polynomial before reaching the constant and $P_n(x)$, $Q_n(x)$ have a nontrivial gcd.

Results

When dealing with numbers the recurrence could give large numbers so we can normalize the polynomials by some constant

$$\begin{array}{l} P_{n-1}(x) = \alpha_{n-1} x\left(q_0^{(n)} P_n(x) - p_0^{(n)} Q_n(x)\right) \\ Q_n(x) = \beta_{n-1}\left(q_n^{(n)} P_n(x) - p_n^{(n)} Q_n(x)\right) \end{array} \tag{5}$$

Choosing for example α and β such that the sum of the absolute value of the coefficients of $P_{n-1}(x)$ and $Q_{n-1}(x)$ are 1 : $\alpha_{n-1}^{-1} = \sum_{k=0}^{n-1} \| p_k^{(n-1)} \|$, $\beta_{n-1}^{-1} = \sum_{k=0}^{n-1} \| q_k^{(n-1)} \|$, or that the maximum of the coefficients is always 1 : $\alpha_{n-1}^{-1} = max(p_k^{(n-1)})$, $\beta_{n-1}^{-1} = max(q_k^{(n-1)})$.

For example

$$\begin{cases} P_8(x) = x^8 - 4x^6 + 4x^5 - 29x^4 + 20x^3 + 24x^2 + 16x + 48 \\ Q_8(x) = x^8 + 3x^7 - 7x^4 - 21x^3 - 6x^2 - 18x \end{cases} \tag{6}$$
$$\begin{cases} p_8 = \{48, 16, 24, 20, -29, 4, -4, 0, 1\} \\ q_8 = \{0, -18, -6, -21, -7, 0, 0, 3, 1\} \end{cases}$$

and let's use the ``max'' normalization. The first iteration says that gcd must divide $P_7(x)$ and $Q_7(x)$:

$$P_7(x) = -121 x Q_8(x) \text{ and } Q_7(x) = 148(P_8(x) - Q_8(x))$$

$$\begin{cases} P_7(x) = -\frac{x^7}{21} - \frac{x^6}{7} + \frac{x^3}{3} + x^2 + \frac{2x}{7} + \frac{6}{7} \\ Q_7(x) = -\frac{x^7}{16} - \frac{x^6}{12} + \frac{x^5}{12} - \frac{11x^4}{24} + \frac{41x^3}{48} + \frac{5x^2}{8} + \frac{17x}{24} + 1 \end{cases}$$

then gcd divide

$$\begin{cases} P_6(x) = \frac{x^6}{78} - \frac{2x^5}{13} - \frac{2x^4}{13} + \frac{11x^3}{13} - \frac{67x^2}{78} + x - \frac{9}{13} \\ Q_6(x) = \frac{x^6}{4} + \frac{x^5}{5} - \frac{11x^4}{10} + x^3 - \frac{33x^2}{20} + \frac{4x}{5} - \frac{3}{10} \end{cases}$$

then gcd divide

$$\begin{cases} P_5(x) = \frac{22x^5}{57} + \frac{8x^4}{19} - \frac{31x^3}{19} + x^2 - \frac{115x}{57} + \frac{11}{19} \\ Q_5(x) = -\frac{32x^5}{187} - \frac{19x^4}{187} + \frac{155x^3}{187} - \frac{151x^2}{187} + x - \frac{12}{17} \end{cases}$$

etc.. finally gcd divide

$$\begin{cases} P_3(x) = x^3 + 3x^2 + x + 3 \\ Q_3(x) = \frac{x^3}{3} + x^2 + \frac{x}{3} + 1 \end{cases}$$

the next step will give $Q_2(x) = 0$ ($3Q_3(x) - P_3(x) = 0$), with the last step:

$$\begin{cases} P_2(x) = P_8(x)\left(\frac{88}{63x^4} + \frac{50}{63x^3} + \frac{229}{378x^2} + \frac{143}{378x}\right) + \\ -Q_8(x)\left(-\frac{704}{189x^5} - \frac{400}{189x^4} + \frac{164}{189x^3} - \frac{100}{189x^2} + \frac{143}{378x}\right) = x^3 + 3x^2 + x + 3 \\ Q_2(x) = P_8(x)\left(-\frac{6}{x^3} + x - \frac{1}{x}\right) - Q_8(x)\left(\frac{16}{x^4} - \frac{8}{x^2} + x + \frac{4}{x} - 3\right) = 0 \end{cases}$$

so we have $gcd(P_8(x), Q_8(x)) = x^3 + 3x^2 + x + 3$

Doing the algorithm on formal polynomials gives automatically the resultant or the discriminant of $P_n(x)$ and $Q_n(x)$.

For example for the gcd of $P_n(x)$ and $P_n(x)'$ for formal polynomials (we always cancel the term x^{m-1} by translation) we have:

$$P_3(x) = x^3 + p\,x + q \quad Q_3(x) = P_3(x)' = 3x^2 + p$$

gives after 3 iterations the well-known discriminant $(4p^3 + 27q^2)$, and the Bezout expression is:

$$\begin{array}{l} p(9qx + 2p^2) P_3(x) - (3pqx^2 + (2p^3 + 9q^2)x + 2p^2 q) Q_3(x) = -(4p^3 + 27q^2)x^3 \\ 3p(2px - 3q) P_3(x) - (px - 3q)(2px + 3q) Q_3(x) = (4p^3 + 27q^2)x^2 \end{array}$$

For the general polynomial of degree 4 :

$$P_4(x) = x^4 + p\,x^2 + q\,x + r \quad Q_4(x) = 4x^3 + 2p\,x + q$$

in 5 iterations we have the discriminant is [5]

$$disc = 256r^3 - 128p^2r^2 + 144pq^2r - 27q^4 + 16p^4r - 4p^3q^2$$

A more formal case [5] is:

$$P_m(x) = x^m + a\,x + b; \quad Q_m(x) = P_m(x)' = m\,x^{m-1} + a$$

so we have successively:

$$k \in [0, m]: p_k^{(m)} = b\,\delta_0^k + a\,\delta_1^k + \delta_m^k \text{ and } q_k^{(m)} = a\,\delta_0^k + m\,\delta_{m-1}^k \tag{7}$$

so

Citation: Nardone P, Sonnino G (2022) A simple algorithm for GCD of polynomials. Ann Math Phys 5(2): 193-195 DOI: https://dx.doi.org/10.17352/amp.000065

$$p_0^{(m)} = b\,;\, p_m^{(m)} = 1\,;\, q_0^{(m)} = a\,;\, q_m^{(m)} = 0\,;\, \Delta_m = -a$$
$$k \in [0, m-1] \begin{cases} p_k^{(m-1)} = a^2 \delta_0^k - mb\, \delta_{m-2}^k + a\, \delta_{m-1}^k \\ q_k^{(m-1)} = -a\delta_0^k - m\delta_{m-1}^k \end{cases} \quad (8)$$

then

$$p_0^{(m-1)} = a^2\,;\, p_{m-1}^{(m-1)} = a\,;\, q_0^{(m-1)} = -a\,;\, q_{m-1}^{(m-1)} = -m\,;\, \Delta_{m-1} = -(m-1)a^2$$
$$k \in [0, m-2] \begin{cases} p_k^{(m-2)} = mab\, \delta_{m-3}^k + (m-1)a^2\, \delta_{m-2}^k \\ q_k^{(m-2)} = -(m-1)a^2\delta_0^k + m^2 b\, \delta_{m-2}^k \end{cases} \quad (9)$$

this structure will repeat, indeed, if

$$k \in [0, m-j] \begin{cases} p_k^{(m-j)} = A_j\, \delta_{m-j-1}^k + B_j\, \delta_{m-j}^k \\ q_k^{(m-j)} = -B_j \delta_0^k + C_j\, \delta_{m-j}^k \end{cases} \quad (10)$$

then $p_0^{(m-j)} = 0$, $p_{m-j}^{(m-j)} = B_j$, $q_0^{(m-j)} = -B_j$, $q_{m-j}^{(m-j)} = C_j$, then $\Delta_{m-j} = B_j^2$ and the next coefficients are:

$$k \in [0, m-j-1] \begin{cases} p_k^{(m-j-1)} = -B_j A_j\, \delta_{m-j-2}^k - B_j^2\, \delta_{m-j-1}^k \\ q_k^{(m-j-1)} = B_j^2 \delta_0^k + C_j A_j\, \delta_{m-j-1}^k \end{cases} \quad (11)$$

so we have the recurrence $A_{j+1} = -B_j A_j$, $B_{j+1} = -B_j^2$ and $C_{j+1} = C_j A_j$ from $j = 2$ (with $A_2 = mab$, $B_2 = (m-1)a^2$, $C_2 = m^2 b$) up to $j = m-2$. At $j = m-1$ we arrive then to:

$$k \in [0,1] \begin{cases} p_k^{(1)} = A_{m-1}\, \delta_0^k + B_{m-1}\, \delta_1^k \\ q_k^{(1)} = -B_{m-1} \delta_0^k + C_{m-1}\, \delta_1^k \end{cases} \quad (12)$$

with $p_0^{(1)} = A_{m-1}$, $p_1^{(1)} = B_{m-1}$, $q_1^{(1)} = C_{m-1}$ and $q_0^{(1)} = -B_{m-1}$ so $\Delta_1 = C_{m-1}A_{m-1} + B_{m-1}^2$ and the last iteration gives the constant:

$$\begin{cases} p_0^{(0)} = -B_{m-1}^2 - A_{m-1}C_{m-1} \\ q_0^{(0)} = C_{m-1}A_{m-1} + B_{m-1}^2 \end{cases} \quad (13)$$

the recurrence on B_j, A_j and C_j gives ($j \geq 2$)

$$B_j = -\left((m-1)a^2\right)^{2^{j-2}}\,;\, A_j = m\,a\,b\left((m-1)a^2\right)^{-1+2^{j-2}}$$
$$C_j = (m-1)m^j a^j b^{j-1}\left((m-1)a^2\right)^{2^{j-2}-j}$$

so the final constant term is

$$(m-1)^{-m+2^{m-2}+1} a^{2^{m-1}-m}\left(m^m b^{m-1} + (m-1)^{m-1} p^m\right)$$

we can factorize the constant and the discriminant is then [5].

$$m^m b^{m-1} + (m-1)^{m-1} a^m \quad (14)$$

Discussion

The algorithm developed here could be used for formal or numerical calculation of the gcd of two polynomials, or the discriminant and the resultant. It doesn't use matrix manipulation nor determinant calculations and for polynomials of order n, it takes n steps to achieve the goal. It provide also the two polynomials needed for Bezout identity.

Supplementary information

Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

(Appendix)

References

1. Knuth DE. The Art of Computer Programming. Addison-Wesley, Reading, Mass. 1969; 2.
2. Bini DA, Boito P. Structured matrix-based methods for polynomial ϵ-gcd: analysis and comparisons. In Proceedings of the 2007 international symposium on Symbolic and algebraic computation (ISSAC '07). Association for Computing Machinery, New York, NY. USA. 9-16. 2007. DOI: https://doi.org/10.1145/1277548.1277551
3. Fazzi A, Guglielmi N, Markovsky I. Generalized algorithms for the approximate matrix polynomial GCD of reducing data uncertainties with application to MIMO system and control. Journal of Computational and Applied Mathematics. 2021; 393: 113499. https://doi.org/10.1016/j.cam.2021.113499
4. Brown WS, Traub JF. On Euclid's Algorithm and the Theory of Subresultants. J ACM. 1971; 18: 505-514. DOI:https://doi.org/10.1145/321662.321665
5. http://www2.math.uu.se/ svante/papers/sjN5.pdf

Citation: Nardone P, Sonnino G (2022) A simple algorithm for GCD of polynomials. Ann Math Phys 5(2): 193-195 DOI: https://dx.doi.org/10.17352/amp.000065