Simple algorithm for GCD of polynomials

NARDONE PASQUALE,SONNINO GIORGIO
Physics Department
Université Libre de Bruxelles
50 av F. D. Roosevelt, Bruxelles 1050
BELGIUM

Abstract: Based on the Bezout approach we propose a simple algorithm to determine the gcd of two polynomials which doesn'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 nor determinant calculation.

Key-Words: Bezout's identity, polynomial remainder sequence, resultant, discriminant
Received: April 27, 2022. Revised: October 28, 2022. Accepted: December 2, 2022. Published: December 31, 2022.

1 Introduction

There exist different approach to determine the greatest common divisor (gcd) for two polynomials, most of them are based on Euclid algorithm [1] or matrix manipulation [4] [5] or subresultant techniques [2]. All these methods require are long manipulations and calculations around $O(n^2)$ for polynomials of degree n. Bezout identity could be another approach. If $P_n(x)$ is a polynomial of degree n and $Q_n(x)$ is a polynomial of degree at least n, the Bezout identity says that $\texttt{gcd}(P_n(x), Q_n(x)) = s(x)P_n(x) + t(x)Q_n(x)$ 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 which use only a linear combination of $P_n(x)$ and $Q_n(x)$ and division by x to decrease the degree of both polynomials by 1.

2 Problem Formulation

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

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

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

$$\begin{aligned} \mathtt{p}_n &= \{p_0^{(n)}, p_1^{(n)}, \cdots, p_{n-1}^{(n)}, p_n^{(n)}\} \\ \mathtt{q}_n &= \{q_0^{(n)}, q_1^{(n)}, \cdots, q_{n-1}^{(n)}, q_n^{(n)}\} \end{aligned}$$

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 cancelling the lowest degree term and the highest degree term:

$$\begin{aligned} P_{n-1}(x) &= \tfrac{1}{x}(q_0^{(n)} P_n(x) - p_0^{(n)} Q_n(x)) \\ Q_{n-1}(x) &= q_n^{(n)} P_n(x) - p_n^{(n)} Q_n(x) \end{aligned} \qquad (1)$$

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

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

This correspond to the manipulation on the list of coefficients:

$$\begin{aligned} p_k^{(n-1)} &= q_0^{(n)} p_{k+1}^{(n)} - p_0^{(n)} q_{k+1}^{(n)} \\ q_k^{(n-1)} &= q_n^{(n)} p_k^{(n)} - p_n^{(n)} q_k^{(n)} \end{aligned}$$

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

If $\Delta_n = 0$:

$$\begin{aligned} \tilde{q}_0^{(n)} &= 0 \\ \tilde{q}_k^{(n)} &= p_0^{(n)} q_{k-1}^{(n)} - q_0^{(n)} p_{k-1}^{(n)} \end{aligned} \qquad k \in [1, n]$$

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

In term of list manipulation we have (if $\Delta_n \neq 0$) :

$$\begin{aligned} \mathtt{p}_{n-1} &= \mathtt{Drop}[\mathtt{First}[\mathtt{q}_n]\ \mathtt{p}_n - \mathtt{First}[\mathtt{p}_n]\ \mathtt{q}_n, 1] \\ \mathtt{q}_{n-1} &= \mathtt{Drop}[\mathtt{Last}[\mathtt{q}_n]\ \mathtt{p}_n - \mathtt{Last}[\mathtt{p}_n]\ \mathtt{q}_n, -1] \end{aligned}$$

where `First[list]` and `Last[list]` takes 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 $p_0^{(n)}\mathtt{q}_n - q_0^{(n)}\mathtt{p}_n$ ends with 0 so the list manipulation is :

$$\tilde{q}_n = \mathtt{RotateRight}[\mathtt{First}[\mathtt{p}_n]\mathtt{q}_n - \mathtt{First}[\mathtt{q}_n]\mathtt{p}_n]$$

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

So we have the same Bezout argument, the $\mathtt{gcd}(P_n(x), Q_n(x))$ 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 $P_{n-k}(x)$ and $Q_{n-k}(x)$.

If we reach a constant : $P_0(x) = p_0^{(0)}$ and $Q_0(x) = q_0^{(0)} = -p_0^{(0)}$ then $\mathtt{gcd}(P_n(x), Q_n(x)) = 1$. If we reach, at 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 decreases the degree of polynomials. Reversing the process enables us to find a combinations 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 non trivial gcd.

2.1 Result

When dealing with numbers the recurrence could gives large numbers so we can normalise the polynomials by some constant

$$\begin{aligned} P_{n-1}(x) &= \tfrac{\alpha_{n-1}}{x}(q_0^{(n)}P_n(x) - p_0^{(n)}Q_n(x)) \\ Q_n(x) &= \beta_{n-1}(q_n^{(n)}P_n(x) - p_n^{(n)}Q_n(x)) \end{aligned} \tag{3}$$

choosing for example α and β such that the sum of 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: $\alpha_{n-1}^{-1} = \mathtt{max}(p_k^{(n-1)})$, $\beta_{n-1}^{-1} = \mathtt{max}(q_k^{(n-1)})$.

For example $P_8(x) = x^8 - 4x^6 + 4x^5 - 29x^4 + 20x^3 + 24x^2 + 16x + 48$ and $Q_8(x) = x^8 + 3x^7 - 7x^4 - 21x^3 - 6x^2 - 18x$, and let's use the "max" normalisation. The first iteration says that gcd must divide $P_7(x)$ and $Q_7(x)$:

$$P_7(x) = -\frac{1}{21x}Q_8(x) \text{ and } Q_7(x) = \frac{1}{48}(P_8(x) - Q_8(x))$$

$P_7(x) = -\frac{x^7}{21} - \frac{x^6}{7} + \frac{x^3}{3} + x^2 + \frac{2x}{7} + \frac{6}{7}$ and $Q_7(x) = -\frac{x^7}{16} - \frac{x^6}{12} + \frac{x^5}{12} - \frac{11x^4}{24} + \frac{41x^3}{48} + \frac{5x^2}{8} + \frac{17x}{24} + 1$, 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: $P_2(x) = P_8(x)\left(\frac{88}{63x^4} + \frac{50}{63x^3} + \frac{229}{378x^2} + \frac{143}{378x}\right) - Q_8(x)\left(-\frac{704}{189x^5} - \frac{400}{189x^4} + \frac{164}{189x^3} - \frac{100}{189x^2} + \frac{143}{378x}\right) = x^3 + 3x^2 + x + 3$ and

$Q_2(x) = P_8(x)\left(-\frac{6}{x^3} + x - \frac{1}{x}\right) - Q_8(x)\left(\frac{16}{x^4} - \frac{8}{x^2} + x + \frac{4}{x} - 3\right) = 0$

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

2.2 On formal polynomials

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: $p(9qx + 2p^2)P_3(x) - (3pqx^2 + (2p^3 + 9q^2)x + 2p^2q)Q_3(x) = -(4p^3 + 27q^2)x^3$ and $3p(2px - 3q)P_3(x) - (px - 3q)(2px + 3q)Q_3(x) = (4p^3 + 27q^2)x^2$

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 [3]

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

A more formal case [3] is: $P_m(x) = x^m + a\,x + b$ and $Q_m(x) = P_m(x)' = m\,x^{m-1} + a$ applying the procedure gives then the discriminant [3]

$$m^m b^{m-1} + (m-1)^{m-1} a^m \tag{4}$$

3 Conclusion

The algorithm developed here could be use 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.

References:

[1] Knuth, D.E. The Art of Computer Programming, Vol. 2. Addison-Wesley, Reading, Mass., 1969

[2] W. S. Brown and J. F. Traub. 1971. On Euclid's Algorithm and the Theory of Subresultants. J. ACM 18, 4 (Oct. 1971), 505-514. DOI:https://doi.org/10.1145/321662.321665

[3] http://www2.math.uu.se/ svante/papers/sjN5.pdf

[4] Dario A. Bini and Paola Boito. 2007. 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. DOI:https://doi.org/10.1145/1277548.1277551

[5] Fazzi, A., Guglielmi, N., & Markovsky, I. (2021). 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, 393, [113499]. https://doi.org/10.1016/j.cam.2021.113499

Contribution of individual authors to the creation of a scientific article (ghostwriting policy)

Pasquale Nardone and Giorgio Sonnino contributed equally to the development of the algorithm.