EXPLOITING EUCLID'S ALGORITHM FOR FUN AND PROFIT

Edmund A. Lamagna
Department of Computer Science
University of Rhode Island
Kingston, Rhode Island 02881

E-mail: eal@cs.uri.edu

Euclid's algorithm for finding the greatest common divisor (GCD) of two integers is perhaps the oldest nontrivial algorithm that has survived to the present day. GCD computation is at the heart of every computer algebra system. Find a better way to compute polynomial GCDs and you can build a faster computer algebra system.

We consider how the basic and extended Euclidean algorithms for polynomial GCDs can be efficiently implemented in a CAS. We also present some surprising applications including factorization of polynomials, simplification of expressions involving radicals, and integration of rational functions.

Integer GCDs: The Basic and Extended Euclidean Algorithms

We are probably all familiar with Euclid's method for finding the greatest common divisor of two non-negative integers. The procedure may be described recursively as follows:

$$\gcd(u,v) = \begin{cases} \gcd(v, u \bmod v) & \text{if } v \neq 0, \\ u & \text{if } v = 0. \end{cases}$$

The following example illustrates how the procedure works. It also shows how the algorithm can be extended to determine integers s, t such that the GCD is expressed as a linear combination of the original inputs, $\gcd(u, v) = su + tv$.

$$\begin{aligned} \gcd(700, 440) &= \gcd(440, 260) & & \\ &= \gcd(260, 180) & 260 &= 700 - 440 \\ &= \gcd(180, 80) & 180 &= 440 - 260 = -1\,(700) + 2\,(440) \\ &= \gcd(80, 20) & 80 &= 260 - 180 = 2\,(700) - 3\,(440) \\ &= \gcd(20, 0) & 20 &= 180 - 2\,(80) = -5\,(700) + 8\,(440) \\ &= 20. & & \end{aligned}$$

Note that the integers s, t are not unique. For example,

$$\begin{aligned} \gcd(30, 18) = 6 &= -1\,(30) + 2\,(18) \text{ (found by extended Euclidean algorithm)} \\ &= 2\,(30) - 3\,(18) \\ &= -4\,(30) + 7\,(18). \end{aligned}$$

Polynomial GCDs

The basic and extended versions of Euclid's algorithm work for polynomials with integer and rational coefficients, but some minor adjustments are needed. Here's a first cut:

$$\gcd(u,v) = \begin{cases} \gcd(v, \text{remainder}(u,v) & \text{if degree}(v) \neq 0, \\ u & \text{if degree}(v) = 0 \text{ and } v = 0, \\ 1 & \text{if degree}(v) = 0 \text{ and } v \neq 0. \end{cases}$$

For example,

$$\begin{aligned} \gcd(x^2 - 1, 2x^2 + 4x + 2) &= \gcd(2x^2 + 4x + 2, -2x - 2) \\ &= \gcd(-2x - 2, 0) \\ &= -2x - 2. \end{aligned}$$

What happened? We expected the result to be $x+1$ since

$$\begin{aligned} x^2 - 1 &= (x + 1)(x - 1) \\ 2x^2 + 4x + 2 &= 2(x + 1)^2. \end{aligned}$$

But we can also write

$$\begin{aligned} x^2 - 1 &= 1/2\,(-2x - 2)(-x + 1) \\ 2x^2 + 4x + 2 &= -(-2x - 2)(x + 1). \end{aligned}$$

The difficulty is that we want our factorizations to be *unique*. How can we adjust Euclid's algorithm? We define the *content* of a polynomial with integer coefficients, content(u), to be the GCD of its coefficients, and the *primitive part*, pp(u) = u/content(u). If we work only with the primitive parts of the polynomials at each iteration, dividing the content out of the remainder, the GCD computed is unique up to unit factors (*i.e.*, ± 1).

Euclid's algorithm for polynomial GCDs, like the method for integers, can be extended to produce polynomials s, t such that gcd(u, v) is expressed as $su + tv$. For example,

$$\begin{aligned} \gcd(x^4 + 4, x^4 + 2x^3 + x^2 - 2x - 2) &= x^2 + 2x + 2 \\ &= (2/5\,x + 3/5)(x^4 + 4) + (-2/5\,x + 1/5)(x^4 + 2x^3 + x^2 - 2x - 2). \end{aligned}$$

Note that the quotient and remainder produced by dividing two polynomials with integer coefficients are, in general, polynomials with rational (rather than integer) coefficients.

Remainder Sequences

Polynomial GCD calculation lies at the heart of any computer algebra system and plays a prominent role in many applications besides simple arithmetic operations on rational functions. As a result, the development of more efficient ways to calculate polynomial GCDs can greatly improve the performance of a computer algebra system.

We will explore and compare several strategies for determining polynomial GCDs. In the discussion below, we present the sequence of remainders produced by applying each method to the polynomials

$$u = x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5 \quad \text{and} \quad v = 3x^6 + 5x^4 - 4x^2 - 9x + 21,$$

whose GCD is 1.

Rational remainder sequence. In this strategy, we apply Euclid's algorithm directly and work with remainders that, in general, have rational coefficients.

$-5/9\, x^4 + 1/9\, x^2 - 1/3 \qquad 233150/19773\, x - 102500/6591$

$-117/25\, x^2 - 9\, x + 441/25 \qquad -1288744821/543589225$

The disadvantages are the growth in the size of the rational coefficients, and the integer GCD calculation required to reduce the fractions arising in the course of each division.

Euclidean remainder sequence. Here we keep the computation in the domain of the integers by using "pseudo-division", but the content is not removed from the remainders.

$-15\, x^4 + 3\, x^2 - 9 \qquad 1254542875143750\, x - 1654608338437500$

$15795\, x^2 + 30375\, x - 59535 \qquad 12593338795500743100931141992187500$

The advantage of this approach over the rational sequence is that no integer GCD calculation is required. However, the coefficients grow to enormous (exponential) sizes.

Monic rational remainder sequence. In this approach, we work with rational coefficients but normalize the remainders after each division to have a leading coefficient of one. Therefore, we always divide monic polynomials.

$-5/9\, x^4 + 1/9\, x^2 - 1/3 \qquad -46630/2197\, x + 61500/2197$

$-39/25\, x^2 - 3\, x + 147/25 \qquad 11014913/21743569$

The coefficients are smaller than in the rational remainder sequence, but we still must perform many integer GCD computations as each division is carried out.

Primitive remainder sequence. In this strategy, we work in the domain of the integers but remove the content (*i.e.*, GCD of the coefficients) from the remainders at each step.

$5\, x^4 - x^2 + 3 \qquad 4663\, x - 6150$

$13\, x^2 + 25\, x - 49 \qquad 1$

The growth of the coefficients is kept to the absolute minimum, but integer GCD calculations are required to determine the contents.

Reduced remainder sequence. While the coefficients produced by this sequence are larger than those in the primitive sequence, no integer GCD calculation is used at all.

$-15\, x^4 + 3\, x^2 - 9 \qquad -18885150\, x + 24907500$

$585\, x^2 + 1125\, x - 2205 \qquad 527933700$

The method works best when the degree of the remainder drops by one at each iteration.

Subresultant remainder sequence. Like the reduced sequence, this strategy also eliminates integer GCD calculation.

$15\, x^4 - 3\, x^2 + 9 \qquad 9326\, x - 12300$

$65\, x^2 + 125\, x - 245 \qquad 260708$

The size of the coefficients is *guaranteed* to grow about linearly.

Applications

The most obvious application of Euclid's algorithm in a computer algebra system is to simplify rational functions. For example,

$$\frac{x^6-1}{x^4-1} = \frac{x^4+x^2+1}{x^2+1}$$

where the common factor of x^2-1 has been cancelled out of both the numerator and denominator. However, the Euclidean and extended Euclidean algorithms are used in many other, perhaps surprising, contexts. We indicate a few of these applications here.

Polynomial factorization. The problem of factoring a polynomial is inherently more difficult than that of identifying common factors of two polynomials. Nonetheless, the Euclidean algorithm plays an important role in the implementation of factoring algorithms.

First, Euclid's algorithm can be applied to detect repeated factors. A polynomial p has repeated factors if and only if $\gcd(p, p') = 1$. A proof of this fact embodies a procedure for computing what is called the *square-free decomposition.* This algorithm involves taking only derivatives and polynomial GCDs.

An example of a polynomial in square-free form is

$$p(x) = (x^2+1)\,(x^2-1)^4\,(x^3+3x)^5.$$

Note that

$$p'(x) = 2x(x^2-1)^4(x^3+3x)^5 + 8x(x^2+1)(x^2-1)^3(x^3+3x)^5 + 15(x^2+1)^2(x^2-1)^4(x^3+3x)^4$$

and $\gcd(p, p') = (x^2-1)^3\,(x^3+3x)^4 \neq 1$. $\gcd(p, p')$ identifies the repeated factors! Once the square-free form has been found, each component can be factored separately to produce the complete factorization,

$$p(x) = (x^2+1)\,(x-1)^4\,(x+1)^4\,x^5\,(x^2+3)^5.$$

There is yet another application of the Euclidean algorithm in factorization. Most computer algebra systems factor a polynomial with integer coefficients by finding the factorization modulo a prime number m, and then "lifting" this result to a factorization over the integers. The extended Euclidean algorithm coupled with a technique called Hensel lifting enables us to start with the factorization modulo a very small prime, m, to produce the factorizations modulo m^2, m^3, ... until the factorization over the integers is uncovered.

Simplification of expression with radicals. In elementary algebra, we are taught a simple technique to "rationalize the denominator" of expressions involving radicals. For example,

$$1/(2\sqrt{5}-1) = 1/19\,(2\sqrt{5}+1).$$

The technique we learned in school works well for small examples but is neither sufficiently algorithmic nor sufficiently powerful to deal with transformations such as