Factorization of polynomials

From Wikipedia, the free encyclopedia

In mathematics and computer algebra, **factorization of polynomials** or **polynomial factorization** refers to factoring a polynomial with coefficients in a given field or in the integers into irreducible factors with coefficients in same domain. Polynomial factorization is one of the fundamental tools of the computer algebra systems.

The history of polynomial factorization starts with Hermann Schubert who in 1793 described the first polynomial factorization algorithm, and Leopold Kronecker, who rediscovered Schubert's algorithm in 1882 and extended it to multivariate polynomials and coefficients in an algebraic extension. But most of the knowledge on this topic is not older than circa 1965 and the first computer algebra systems. In a survey of the subject, Erich Kaltofen wrote in 1982 (see the bibliography, below):

> When the long-known finite step algorithms were first put on computers, they turned out to be highly inefficient. The fact that almost any uni- or multivariate polynomial of degree up to 100 and with coefficients of a moderate size (up to 100 bits) can be factored by modern algorithms in a few minutes of computer time indicates how successfully this problem has been attacked during the past fifteen years.

Nowadays[1] one can quickly factor any univariate polynomial of degree 1000, and coefficients with thousands of digits.

Contents

Formulation of the question

Polynomial rings over the integers or over a field are unique factorization domains. This means that every element of these rings is a product of a constant and a product of irreducible polynomials (those that are not the product of two non-constant polynomials). Moreover, this decomposition is unique up to multiplication of the factors by invertible constants.

Factorization depends on the base field. For example, the fundamental theorem of algebra, which states that every polynomial with complex coefficients has complex roots, implies that a polynomial with integer coefficients can be factored (with root-finding algorithms) into linear factors over the complex field **C**.

Similarly, over the field of reals, the irreducible factors have degree at most two, while there are polynomials of any degree that are irreducible over the field of rationals **Q**.

The question of polynomial factorization makes sense only for coefficients in a *computable field* whose every element may be represented in a computer and for which there are algorithms for the arithmetic operations. Fröhlich and Shepherson have provided examples of such fields for which no factorization algorithm can exist.

The fields of coefficients for which factorization algorithms are known include prime fields (i.e. the field of rationals and prime modular arithmetic) and their finitely generated field extensions. Integer coefficients are also tractable: Kronecker's method is interesting only from a historical point of view, modern algorithms proceed by a succession of:

- Square-free factorization
- Factorization over finite fields

and reductions:

- From the multivariate case to the univariate one
- From coefficients in a purely transcendental extension to the multivariate case over the ground field (see below)
- From coefficients in an algebraic extension to coefficients in the ground field
- From rational coefficients to integer coefficients (see below)
- From integer coefficients to coefficients in a prime field with p elements, for a well chosen p.

Primitive part–content factorization

See also: Content (algebra) and Gauss's lemma (polynomial)

In this section, we show that factoring over **Q** (the rational numbers) and over **Z** (the integers) is essentially the same problem.

The *content* of a polynomial $p \in \mathbf{Z}[X]$, denoted "cont(p)", is, up to its sign, the greatest common divisor of its coefficients. The *primitive part* of p is primpart(p)=p/cont(p), which is a primitive polynomial with integer coefficients. This defines a factorization of p into the product of an integer and a primitive polynomial. This factorization is unique up to the sign of the content. It is a usual convention to choose the sign of the content such that the leading coefficient of the primitive part is positive.

For example,

$$-10x^2 + 5x + 5 = (-5) \cdot (2x^2 - x - 1)$$

is a factorization into content and primitive part.

Every polynomial q with rational coefficients may be written

$$q = \frac{p}{c},$$

where $p \in \mathbf{Z}[X]$ and $c \in \mathbf{Z}$: it suffices to take for c a multiple of all denominators of the coefficients of q (for example their product) and $p = cq$. The *content* of q is defined as:

$$\operatorname{cont}(q) = \frac{\operatorname{cont}(p)}{c},$$

and the *primitive part* of q is that of p. As for the polynomials with integer coefficients, this defines a factorization into a rational number and a primitive polynomial with integer coefficients. This factorization is also unique up to the choice of a sign.

For example,

$$\frac{1}{3}x^5 + \frac{7}{2}x^2 + 2x + 1 = \frac{1}{6}(2x^5 + 21x^2 + 12x + 6)$$

is a factorization into content and primitive part.

Gauss has first proved that the product of two primitive polynomials is also primitive (Gauss's lemma). This implies that a primitive polynomial is irreducible over the rationals if and only if it is irreducible over the integers. This implies also that the factorization over the rationals of a polynomial with rational coefficients is the same as the factorization over the integers of its primitive part. On the other hand, the factorization over the integers of a polynomial with integer coefficients is the product of the factorization of its primitive part by the factorization of its content.

In other words, integer GCD computation allows to reduce the factorization of a polynomial over the rationals to the factorization of a primitive polynomial with integer coefficients, and to reduce the factorization over the integers to the factorization of an integer and a primitive polynomial.

Everything that precedes remains true if **Z** is replaced by a polynomial ring over a field F and **Q** is replaced by a field of rational functions over F in the same variables, with the only difference that "up to a sign" must be replaced by "up to the multiplication by an invertible constant in F". This allows to reduce the factorization over a purely transcendental field extension of F to the factorization of multivariate polynomials over F.

Square-free factorization

Main article: square-free polynomial

If two or more factors of a polynomial are identical to each other, then the polynomial is a multiple of the square of this factor. In the case of univariate polynomials, this results in multiple roots. In this case, then the multiple factor is also a factor of the polynomial's derivative (with respect to any of the variables, if several). In the case of univariate polynomials over the rationals (or more generally over a field of characteristic zero), Yun's algorithm exploits this to factorize efficiently the polynomial into factors that are not multiple of a square and are therefore called **square-free**. To factorize the initial polynomial, it suffices to factorize each square-free factor. Square-free factorization is therefore the first step in most polynomial factorization algorithms.

Yun's algorithm extends to the multivariate case by considering a multivariate polynomial as an univariate polynomial over a polynomial ring.

In the case of a polynomial over a finite field, Yun's algorithm applies only if the degree is smaller than the characteristic, because, otherwise, the derivative of a non zero polynomial may be zero (over the field with p elements, the derivative of a polynomial in x^p is always zero). Nevertheless a succession of GCD computations, starting from the polynomial and its derivative, allows to compute the square-free decomposition; see Polynomial factorization over finite fields#Square-free factorization.

Classical methods

This section describes textbook methods that can be convenient when computing by hand. These methods are not used for computer computations because they use integer factorization, which at the moment has a much higher complexity than polynomial factorization.

Obtaining linear factors

All linear factors with rational coefficients can be found using the rational root test. If the polynomial to be factored is $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$, then all possible linear factors are of the form $b_1 x - b_0$, where b_1 is an integer factor of a_n and b_0 is an integer factor of a_0. All possible combinations of integer factors can be tested for validity, and each valid one can be factored out using polynomial long division. If the original polynomial is the product of factors, at least two of which are of degree 2 or higher, this technique only provides a partial factorization; otherwise the factorization is complete. Note that in the case of a cubic polynomial, if the cubic is factorisable at all, the rational root test gives a complete factorization, either into a linear factor and an irreducible quadratic factor, or into three linear factors.

Kronecker's method

Since integer polynomials must factor into integer polynomial factors, and evaluating integer polynomials at integer values must produce integers, the integer values of a polynomial can be factored in only a finite number of ways, and produce only a finite number of possible polynomial factors.

For example, consider

$$f(x) = x^5 + x^4 + x^2 + x + 2.$$

If this polynomial factors over **Z**, then at least one of its factors must be of degree two or less. We need three values to uniquely fit a second degree polynomial. We'll use $f(0) = 2, f(1) = 6$ and $f(-1) = 2$. Note that if one of those values were 0 then you already found a root (and so a factor). If none is 0, then each one has a finite amount of divisors. Now, 2 can only factor as

1×2, 2×1, (−1)×(−2), or (−2)×(−1).

Therefore, if a second degree integer polynomial factor exists, it must take one of the values

1, 2, −1, or −2

at $x = 0$, and likewise at $x = -1$. There are eight different ways to factor 6 (one for each divisor of 6), so there are

4×4×8 = 128

possible combinations, of which half can be discarded as the negatives of the other half, corresponding to 64 possible second degree integer polynomials that must be checked. These are the only possible integer polynomial factors of $f(x)$. Testing them exhaustively reveals that

$$p(x) = x^2 + x + 1$$

constructed from $p(0) = 1, p(1) = 3$ and $p(-1) = 1$, factors $f(x)$.

Dividing f by p gives the other factor $q(x) = x^3 - x + 2$, so that $f = pq$. Now one can test recursively to find factors of p and q. It turns out they both are irreducible over the integers, so that the irreducible factorization of f is

$$f(x) = p(x)q(x) = (x^2 + x + 1)(x^3 - x + 2)$$

(*Van der Waerden*, Sections 5.4 and 5.6)

Modern methods

Factoring over finite fields

Main articles: Factorization of polynomials over finite fields, Berlekamp's algorithm, and Cantor–Zassenhaus algorithm

Factoring univariate polynomials over the integers

If $f(x)$ is a univariate polynomial over the integers, assumed to be content-free and square-free, one starts by computing a bound B such that any factor $g(x)$ will have coefficients of absolute value bounded by B. This way, if m is an integer larger than $2B$, and if $g(x)$ is known modulo m, then $g(x)$ can be reconstructed from its image mod m.

The Zassenhaus algorithm proceeds as follows. First, choose a prime number p such that the image of $f(x)$ mod p remains square-free, and of the same degree as $f(x)$. Then factor $f(x)$ mod p. This produces integer polynomials $f_1(x), ..., f_r(x)$ whose product matches $f(x)$ mod p. Next, apply Hensel lifting, this updates the $f_i(x)$ in such a way that now their product matches $f(x)$ mod p^a, where a is chosen in such a way that p^a is larger than $2B$. Modulo p^a, the polynomial $f(x)$ has (up to units) 2^r factors: for each subset of $f_1(x), ..., f_r(x)$, the product is a factor of $f(x)$ mod p^a. However, a factor modulo p^a need not correspond to a so-called "true factor": a factor of $f(x)$ in $Z[x]$. For each factor mod p^a, we can test if it corresponds to a "true" factor, and if so, find that "true" factor, provided that p^a exceeds $2B$. This way, all irreducible "true" factors can be found by checking at most 2^r cases. This is reduced to 2^{r-1} cases by skipping complements. If $f(x)$ is reducible, the number of cases is reduced further by removing those $f_i(x)$ that appear in an already found "true" factor. Zassenhaus algorithm processes each case (each subset) quickly, however, in the worst case, it considers an exponential number of cases.

The first polynomial time algorithm for factoring rational polynomials has been discovered by Lenstra, Lenstra and Lovász and is an application of Lenstra–Lenstra–Lovász lattice basis reduction algorithm, usually called "LLL algorithm". (Lenstra, Lenstra & Lovász 1982) A simplified version of the LLL factorization algorithm is as follows: calculate a complex (or *p*-adic) root α of the polynomial $f(x)$ to high precision, then use the Lenstra–Lenstra–Lovász lattice basis reduction algorithm to find an approximate linear relation between 1, α, α^2, α^3, ... with integer coefficients, which might be an exact linear relation and a polynomial factor of $f(x)$. One can determine a bound for the precision that guarantees that this method produces either a factor, or an irreducibility proof. Although this method is polynomial time, it was not used in practice because the lattice has high dimension and huge entries, which makes the computation slow.

The exponential complexity in the algorithm of Zassenhaus comes from a combinatorial problem: how to select the right subsets of $f_1(x), ..., f_r(x)$. State of the art factoring implementations work in a manner similar to Zassenhaus, except that the combinatorial problem is translated to a lattice problem that is then solved by LLL.[2] In this approach, LLL is not used to compute coefficients of factors, instead, it is used to compute vectors with r entries in {0,1} that encode the subsets of $f_1(x), ..., f_r(x)$ that correspond to the irreducible "true" factors.

Factoring over algebraic extensions (Trager's method)

We can factor a polynomial $p(x) \in K[x]$, where K is a finite field extension of $\mathbb{Q}$. First, using square-free factorization, we may suppose that the polynomial is square-free. Next we write $L = K[x]/p(x)$ explicitly as an algebra over $\mathbb{Q}$. We next pick a random element $\alpha \in L$. By the primitive element theorem, α generates L over $\mathbb{Q}$ with high probability. If this is the case, we can compute the minimal polynomial, $q(y) \in \mathbb{Q}[y]$ of α over $\mathbb{Q}$. Factoring

$$q(y) = \prod_{i=1}^{n} q_i(y)$$

over $\mathbb{Q}[y]$, we determine that

$$L = \mathbb{Q}[\alpha] = \mathbb{Q}[y]/q(y) = \prod_{i=1}^{n} \mathbb{Q}[y]/q_i(y)$$

(notice that L is a reduced ring since $p(x)$ is square-free), where α corresponds to the element $(y, y, \ldots, y)$. Note that this is the unique decomposition of L as a product fields. Hence this decomposition is the same as

$$\prod_{i=1}^{m} K[x]/p_i(x)$$

where

$$p(x) = \prod_{i=1}^{m} p_i(x)$$

is the factorization of $p(x)$ over $K[x]$. By writing $x \in L$ and generators of K as a polynomials in α, we can determine the embeddings of x and K into the components $\mathbb{Q}[y]/q_i(y) = K[x]/p_i(x)$. By finding the minimal polynomial of x in this ring, we have computed $p_i(x)$, and thus factored $p(x)$ over K.

Bibliography

1. ^ An example of degree 2401, taking 7.35 seconds, is found in Section 4 in: Hart, van Hoeij, Novocin: *Practical Polynomial Factoring in Polynomial Time* ISSAC'2011 Proceedings, p. 163-170 (2011).
2. ^ M. van Hoeij: *Factoring polynomials and the knapsack problem*. J. of Number Theory, 95, 167-189, (2002).

- Fröhlich, A.; Shepherson, J. C. (1955), "On the factorisation of polynomials in a finite number of

steps", *Mathematische Zeitschrift* **62** (1), doi:10.1007/BF01180640 (http://dx.doi.org/10.1007%2FBF01180640), ISSN 0025-5874 (//www.worldcat.org/issn/0025-5874)
- Trager, B.M., "Algebraic Factoring and Rational Function Integration", *Proc. SYMSAC 76 http://dl.acm.org/citation.cfm?id=806338*
- Bernard Beauzamy, Per Enflo, Paul Wang (October 1994). "Quantitative Estimates for Polynomials in One or Several Variables: From Analysis and Number Theory to Symbolic and Massively Parallel Computation". *Mathematics Magazine* **67** (4): 243–257. doi:10.2307/2690843 (http://dx.doi.org/10.2307%2F2690843). JSTOR 2690843 (//www.jstor.org/stable/2690843). (accessible to readers with undergraduate mathematics)
- Cohen, Henri (1993). *A course in computational algebraic number theory*. Graduate Texts in Mathematics **138**. Berlin, New York: Springer-Verlag. ISBN 978-3-540-55640-4. MR 1228206 (//www.ams.org/mathscinet-getitem?mr=1228206).
- Kaltofen, Erich (1982), "Factorization of polynomials" (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.7916&rep=rep1&type=pdf), in B. Buchberger; R. Loos; G. Collins, *Computer Algebra*, Springer Verlag, retrieved September 20, 2012
- Knuth, Donald E (1997). "4.6.2 Factorization of Polynomials". *Seminumerical Algorithms*. The Art of Computer Programming **2** (Third ed.). Reading, Massachusetts: Addison-Wesley. pp. 439–461, 678–691. ISBN 0-201-89684-2.
- Lenstra, A. K.; Lenstra, H. W.; Lovász, László (1982). "Factoring polynomials with rational coefficients". *Mathematische Annalen* **261** (4): 515–534. doi:10.1007/BF01457454 (http://dx.doi.org/10.1007%2FBF01457454). ISSN 0025-5831 (//www.worldcat.org/issn/0025-5831). MR 682664 (//www.ams.org/mathscinet-getitem?mr=682664).
- Van der Waerden, *Algebra* (1970), trans. Blum and Schulenberger, Frederick Ungar.

Further reading

- Kaltofen, Erich (1990), "Polynomial Factorization 1982-1986" (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.7461&rep=rep1&type=pdf), in D. V. Chudnovsky; R. D. Jenks, *Computers in Mathematics*, Lecture Notes in Pure and Applied Mathematics **125**, Marcel Dekker, Inc., retrieved October 14, 2012
- Kaltofen, Erich (1992), "Polynomial Factorization 1987–1991" (http://www4.ncsu.edu/~kaltofen/bibliography/92/Ka92_latin.pdf), *Proceedings of Latin '92*, Springer Lect. Notes Comput. Sci. **583**, Springer, retrieved October 14, 2012

Categories: Polynomials | Computer algebra
