WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Deutsch
Español
Français
Nederlands
Русский
Українська
Edit links

Dixon's factorization method

From Wikipedia, the free encyclopedia
(Redirected from Dixon's algorithm)

In number theory, **Dixon's factorization method** (also **Dixon's random squares method**[1] or **Dixon's algorithm**) is a general-purpose integer factorization algorithm; it is the prototypical factor base method, and the only factor base method for which a run-time bound not reliant on conjectures about the smoothness properties of values of a polynomial is known.

The algorithm was designed by John D. Dixon, a mathematician at Carleton University, and was published in 1981.[2]

Contents [hide]

1 Basic idea
2 Method
3 Example
4 Optimizations
5 References

Basic idea [edit]

Dixon's method is based on finding a congruence of squares modulo the integer N which we intend to factor. Fermat's factorization algorithm finds such a congruence by selecting random or pseudo-random *x* values and hoping that the integer x^2 mod N is a perfect square (in the integers):

$$x^2 \equiv y^2 \pmod{N}, \qquad x \not\equiv \pm y \pmod{N}.$$

For example, if N = 84923, we notice (by starting at 292, the first number greater than $\sqrt{N}$ and counting up) that 505^2 mod 84923 is 256, the square of 16. So (505 − 16)(505 + 16) = 0 mod 84923. Computing the greatest common divisor of 505 − 16 and N using Euclid's algorithm gives us 163, which is a factor of N.

In practice, selecting random x values will take an impractically long time to find a congruence of squares, since there are only $\sqrt{N}$ squares less than N.

Dixon's method replaces the condition "is the square of an integer" with the much weaker one "has only small prime factors"; for example, there are 292 squares smaller than 84923; 662 numbers smaller than 84923 whose prime factors are only 2,3,5 or 7; and 4767 whose prime factors are all less than 30. (Such numbers are called *B-smooth* with respect to some bound *B*.)

If we have lots of numbers $a_1 \ldots a_n$ whose squares can be factorized as $a_i^2 \mod N = \prod_{j=1}^{m} b_j^{e_{ij}}$ for a fixed set $b_1 \ldots b_m$ of small primes, linear algebra modulo 2 on the matrix e_{ij} will give us a subset of the a_i whose squares combine to a product of small primes to an even power — that is, a subset of the a_i whose squares multiply to the square of a (hopefully different) number mod N.

Method [edit]

Suppose we are trying to factor the composite number N. We choose a bound B, and identify the *factor base* (which we will call P), the set of all primes less than or equal to B. Next, we search for positive integers z such that z^2 mod N is B-smooth. We can therefore write, for suitable exponents a_k,

$$z^2 \equiv \prod_{p_i \in P} p_i^{a_i} \pmod{N}$$

When we have generated enough of these relations (it's generally sufficient that the number of relations be a few more than the size of P), we can use the methods of linear algebra (for example, Gaussian elimination) to multiply together these various relations in such a way that the exponents of the primes on the right-hand side are all even:

$$z_1^2 z_2^2 \cdots z_k^2 \equiv \prod_{p_i \in P} p_i^{a_{i,1}+a_{i,2}+\cdots+a_{i,k}} \pmod{N} \quad (\text{where } a_{i,1}+a_{i,2}+\cdots+a_{i,k} \equiv 0 \pmod{2})$$

This gives us a congruence of squares of the form $a^2 \equiv b^2 \pmod{N}$, which can be turned into a factorization of N, $N = \gcd(a + b, N) \times (N/\gcd(a + b, N))$. This factorization might turn out to be trivial (i.e. $N = N \times 1$), which can only happen if $a \equiv \pm b \pmod{N}$, in which case we have to try again with a different combination of relations; but with luck we will get a nontrivial pair of factors of N, and the algorithm will terminate.

Example [edit]

We will try to factor $N = 84923$ using bound $B = 7$. Our factor base is then $P = \{2, 3, 5, 7\}$. We then search randomly for integers between $\lceil\sqrt{84923}\rceil = 292$ and N whose squares are B-smooth. Suppose that two of the numbers we find are 513 and 537:

$$\begin{aligned} 513^2 &\mod 84923 = 8400 = 2^4 \cdot 3 \cdot 5^2 \cdot 7 \\ 537^2 &\mod 84923 = 33600 = 2^6 \cdot 3 \cdot 5^2 \cdot 7 \end{aligned}$$

So

$$(513 \cdot 537)^2 \mod 84923 = 2^{10} \cdot 3^2 \cdot 5^4 \cdot 7^2$$

Then

$$\begin{aligned} &(513 \cdot 537)^2 \mod 84923 = (275481)^2 \mod 84923 \\ &= (84923 \cdot 3 + 20712)^2 \mod 84923 \\ &= (84923 \cdot 3)^2 + 2 \cdot (84923 \cdot 3 \cdot 20712) + 20712^2 \mod 84923 \\ &= 0 + 0 + 20712^2 \mod 84923 \end{aligned}$$

That is, $20712^2 \mod 84923 = (2^5 \cdot 3 \cdot 5^2 \cdot 7)^2 \mod 84923 = 16800^2 \mod 84923.$

The resulting factorization is 84923 = gcd(20712 − 16800, 84923) × gcd(20712 + 16800, 84923) = 163 × 521.

Optimizations [edit]

The quadratic sieve is an optimization of Dixon's method. It selects values of x close to the square root of N such that x^2 modulo N is small, thereby largely increasing the chance of obtaining a smooth number.

Other ways to optimize Dixon's method include using a better algorithm to solve the matrix equation, taking advantage of the sparsity of the matrix: a number z cannot have more than $\log_2 z$ factors, so each row of the matrix is almost all zeros. In practice, the block Lanczos algorithm is often used. Also, the size of the factor base must be chosen carefully: if it is too small, it will be difficult to find numbers that factorize completely over it, and if it is too large, more relations will have to be collected.

A more sophisticated analysis, using the approximation that a number has all its prime factors less than $N^{1/a}$ with probability about a^{-a} (an approximation to the Dickman–de Bruijn function), indicates that choosing too small a factor base is much worse than too large, and that the ideal factor base size is some power of $\exp\left(\sqrt{\log N \log\log N}\right)$.

The optimal complexity of Dixon's method is

$$O\left(\exp\left(2\sqrt{2}\sqrt{\log n \log\log n}\right)\right)$$

in big-O notation, or

$$L_n[1/2, 2\sqrt{2}]$$

in L-notation.

References [edit]

1. ^ Kleinjung, Thorsten et al. (2010). "Factorization of a 768-bit RSA modulus". *Advances in Cryptology – CRYPTO 2010*. Lecture Notes in Computer Science **6223**. pp. 333–350. doi:10.1007/978-3-642-14623-7_18.
2. ^ Dixon, J. D. (1981). "Asymptotically fast factorization of integers". *Math. Comp.* **36** (153): 255–260. doi:10.1090/S0025-5718-1981-0595059-1. JSTOR 2007743.

v · t · e	Number-theoretic algorithms [hide]
Primality tests	AKS test · APR test · Baillie–PSW · ECPP test · Elliptic curve · Pocklington · Fermat · Lucas · *Lucas–Lehmer* · *Lucas–Lehmer–Riesel* · *Proth's theorem* · *Pépin's* · Quadratic Frobenius test · Solovay–Strassen · Miller–Rabin
Prime-generating	Sieve of Atkin · Sieve of Eratosthenes · Sieve of Sundaram · Wheel factorization
Integer factorization	Continued fraction (CFRAC) · **Dixon's** · Lenstra elliptic curve (ECM) · Euler's · Pollard's rho · *p* − 1 · *p* + 1 · Quadratic sieve (QS) · General number field sieve (GNFS) · *Special number field sieve (SNFS)* · Rational sieve · Fermat's · Shanks' square forms · Trial division · Shor's
Multiplication	Ancient Egyptian · Long · Karatsuba · Toom–Cook · Schönhage–Strassen · Fürer's
Discrete logarithm	Baby-step giant-step · Pollard rho · Pollard kangaroo · Pohlig–Hellman · Index calculus · Function field sieve
Greatest common divisor	Binary · Euclidean · Extended Euclidean · Lehmer's
Modular square root	Cipolla · Pocklington's · Tonelli–Shanks
Other algorithms	Chakravala · Cornacchia · Integer relation · Integer square root · Modular exponentiation · Schoof's
Italics indicate that algorithm is for numbers of special forms · Smallcaps indicate a deterministic algorithm	

Categories: Integer factorization algorithms