Subquadratic-Time Factoring of Polynomials over Finite Fields

Erich Kaltofen

Rensselaer

Rensselaer Polytechnic Institute
Department of Computer Science
Troy, New York, USA

Joint work with Victor Shoup

Outline

- **Factorization of integers and polynomials**
- **Statement of result**
- **Matrix-free linear system solvers**
- **Matrix-free Berlekamp algorithm**
- **Baby steps/giant steps speed-up**
- **Implementation observations**

Factorization of an integer N
(quadratic sieves, number field sieves)

Compute a solution to the congruence equation

$$X^2 \equiv Y^2 \pmod{N}$$

via r relations on b basis primes

$$X_1^2 \cdot X_2^2 \cdots X_r^2 \equiv (p_1^{e_1})^2 \cdot (p_2^{e_2})^2 \cdots (p_b^{e_b})^2 \pmod{N}$$

Then N divides $(X+Y)(X-Y)$, hence

$$\mathrm{GCD}(X+Y, N) \text{ divides } N$$

Factorization of polynomial f over finite field $\mathbb{F}_p$ (Berlekamp 1967 algorithm)

Note that since $a^p \equiv a \pmod{p}$ for all $a \in \mathbb{F}_p$ we have

$$x^p - x \equiv x \cdot (x-1) \cdot (x-2) \cdots (x-p+1) \pmod{p}$$

Compute a polynomial solution to the congruence equation

$$w(x)^p \equiv w(x) \pmod{f(x)}$$

Then f divides $w \cdot (w-1) \cdot (w-2) \cdots (w-p+1)$, hence

$$\mathrm{GCD}(w(x)-a, f(x)) \text{ divides } f(x) \text{ for some } a \in \mathbb{F}_p$$

Solving $w^p \equiv w \pmod{f}$ by linear algebra

For $w(x) \in \mathbb{F}_p[x]$, $\deg(w) < n$:

$$w(x)^p = w(x^p) \;\equiv\; w(x) \pmod{f(x)} \quad \text{(Note: } (a+b)^p = a^p + b^p$$

$$\text{because } \binom{p}{i} \equiv 0 \pmod{p}$$

$$\text{for } 0 < i < p)$$

$$\Updownarrow$$

$$\overrightarrow{w(x^p) \bmod f(x)}^{\mathrm{tr}} = \underbrace{[\, w_0 \ldots w_{n-1} \,]}_{\vec{w}^{\mathrm{tr}}} \cdot \underbrace{\begin{bmatrix} \vdots \\ \overrightarrow{x^{ip} \bmod f(x)}^{\mathrm{tr}} \\ \vdots \end{bmatrix}_{0 \le i < n}}_{\substack{Q \\ \text{(Petr's 1937 matrix)}}} = \vec{w}^{\mathrm{tr}}$$

Run-time comparisons (field arithmetic operations)

	$p = O(1)$	$\log p = \Theta(n)$
Berlekamp '70 $O(n^{\omega} + n^{1+o(1)} \log p)$	$O(n^{2.38})$	$O(n^{2.38})$
Cantor & Zassenhaus '81 $O(n^{2+o(1)} \log p)$	$O(n^{2+o(1)})$	$O(n^{3+o(1)})$
von zur Gathen & Shoup '91 $O(n^{2+o(1)} + n^{1+o(1)} \log p)$	$O(n^{2+o(1)})$	$O(n^{2+o(1)})$
Kaltofen & Shoup '94 $O(n^{(\omega+1)/2+(1-\gamma)(\omega-1)/2} + n^{1+\gamma+o(1)} \log p)$ for any $0 \leq \gamma \leq 1$	$O(n^{1.82})$	$O(n^{2.5})$

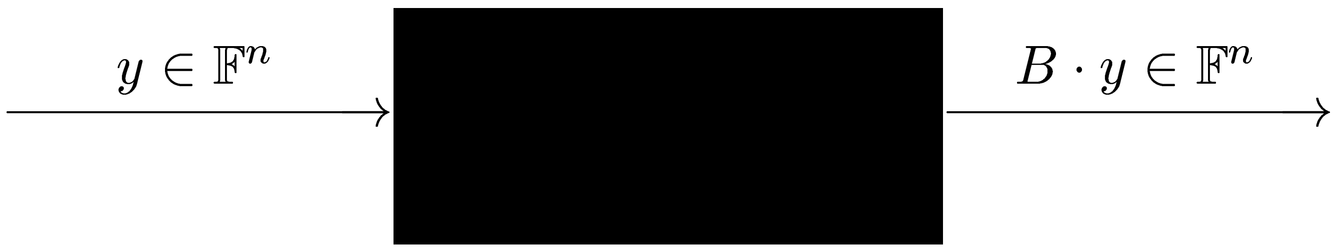
ω = matrix multiplication exponent

Asymptotically fastest methods

degree n versus $\log p = O(n^{\alpha})$

A "black box" matrix

is an efficient **procedure** with the specifications



$B \in \mathbb{F}^{n \times n}$
$\mathbb{F}$ an arbitrary field

i.e., the matrix is not stored explicitly, its structure is unknown.

Main algorithmic problem: How to efficiently solve a linear system with a black box coefficient matrix?

Idea for Wiedemann's algorithm

$B \in \mathbb{F}^{n\times n}$, $\mathbb{F}$ a (possibly finite) field

$\phi^B(\lambda) = c'_0 + c'_1\lambda + \cdots + c'_m\lambda^m \in \mathbb{F}[\lambda]$ **minimum polynomial** of B:

$\forall\, u, v \in \mathbb{F}^n$: $\forall\, j \geq 0$: $u^{\mathrm{tr}} B^j \phi^B(B) v = 0$

$$\Updownarrow$$

$$c'_0 \cdot \underbrace{u^{\mathrm{tr}} B^j v}_{a_j} + c'_1 \cdot \underbrace{u^{\mathrm{tr}} B^{j+1} v}_{a_{j+1}} + \cdots + c'_m \cdot \underbrace{u^{\mathrm{tr}} B^{j+m} v}_{a_{j+m}} = 0$$

$$\Updownarrow$$

$\{a_0, a_1, a_2, \ldots\}$ is generated by a linear recursion

Theorem (Wiedemann 1986): *For random* $u, v \in \mathbb{F}^n$*,* *a linear generator for* $\{a_0, a_1, a_2, \ldots\}$ *is one for* $\{I, B, B^2, \ldots\}$*.*

$$\forall\, j \geq 0\colon c_0 a_j + c_1 a_{j+1} + \cdots + c_d a_{j+d} = 0$$

$$\Big\Downarrow \text{ (with high probability)}$$

$$c_0 B^j v + c_1 B^{j+1} v + \cdots + c_d B^{j+d} v = 0$$

$$\Big\Downarrow \text{ (with high probability)}$$

$$c_0 B^j + c_1 B^{j+1} + \cdots + c_d B^{j+d} = 0$$

that is, $\phi^B(\lambda)$ divides $c_0 + c_1\lambda + \cdots + c_m\lambda^m$

Algorithm *Homogeneous Wiedemann*

Input: $B \in \mathbb{F}^{n\times n}$ singular

Output: $w \neq \mathbf{0}$ such that $Bw = \mathbf{0}$

Step W1: Pick random $u, v \in \mathbb{F}^n$; $\quad b \leftarrow Bv$;
for $i \leftarrow 0$ **to** $2n-1$ **do** $a_i \leftarrow u^{\mathrm{tr}} B^i b$.
(Requires $2n$ black box calls.)

Step W2: Compute a linear recurrence generator for $\{a_i\}$,

$$c_\ell \lambda^\ell + c_{\ell+1}\lambda^{\ell+1} + \cdots + c_d \lambda^d, \quad \ell \geq 0, d \leq n, c_\ell \neq 0,$$

by the Berlekamp/Massey 1967 algorithm.

Step W3: $\widehat{w} \leftarrow c_\ell v + c_{\ell+1} Bv + \cdots + c_d B^{d-\ell} v$;
(With high probability $\widehat{w} \neq 0$ and $B^{\ell+1}\widehat{w} = 0$.)
Compute first k with $B^k \widehat{w} = 0$; **return** $w \leftarrow B^{k-1}\widehat{w}$.

Steps W1 and W3 have the same computational complexity

$$u^{\mathrm{tr}} \cdot [\, v \mid Bv \mid B^2 v \mid \ldots \mid B^{2n} v \,] = [\, a_{-1} \quad a_0 \quad a_1 \quad \ldots \quad a_{2n-1} \,]$$

$$[\, v \mid Bv \mid B^2 v \mid \ldots \mid B^{2n} v \,] \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{2n} \end{bmatrix} = w$$

Fact: $X \cdot y$ and $X^{\mathrm{tr}} \cdot z$ have he same computational complexity (Kaminski, Kirkpatrick, Bshouty 1988).

“Matrix-free” Berlekamp algorithm (K 1991)

Compute $\overrightarrow{v}^{\,\mathrm{tr}} \cdot (Q - I)$ as

$v(x)^p - v(x) \bmod f(x)$ in $n \log p \cdot (\log n)^{O(1)}$ $\mathbb{F}_p$-ops

$$-v(x) + \sum_{i=0}^{n-1} v_i(\underbrace{x^p \bmod f(x)}_{h_1(x)})^i \bmod f(x)$$

$= -v(x) + v(h_1(x)) \bmod f(x)$ in $O(n^{1.7})$ $\mathbb{F}_p$-ops (given h_1)

“modular polynomial composition” (Brent and Kung 1978)

The probabilistic analysis needed when using the Wiedemann algorithm as the solver can be made explict (K & Lobo 1994).

For example, one has:

Fact: If f is squarefree, the minimum polynomial of Q is

$$\phi^Q(\lambda) = \mathrm{LCM}_{1 \le i \le r}(\lambda^{m_i} - 1), \quad \text{where } m_i = \deg(f_i).$$

Note: $\phi^Q(\lambda) = \phi^{Q-I}(\lambda - 1)$.

The baby steps/giant steps polynomial factorizer

Consider computing $a_i = \overrightarrow{u}^{\rm tr} \cdot Q^i \cdot \overrightarrow{v} = (\overrightarrow{u}^{\rm tr} Q^j) \cdot (Q^{tk} \overrightarrow{v})$, where

$$0 \le i \le 2n, 0 \le j < t, 0 \le k \le 2n/t,$$
$$t = \lceil n^\gamma \rceil, 0 \le \gamma \le 1.$$

Baby steps: $\overrightarrow{u}^{\rm tr} \cdot Q^j$ by repeated $u(x)^p \bmod f(x)$.

Giant steps: $Q^{tk} \cdot \overrightarrow{v}$ by repeated **transposed** modular polynomial composition with $h_t(x) = x^{p^t} \bmod f(x)$.

Finally, all a_i by fast rectangular matrix multiplication.

Shoup's baby steps/giant steps implementation

Can factor a 1024 degree pseudo-random polynomial modulo a 1024 bit prime number in about 50 hours on a **single** 20 MIPS computer.

The algorithm requires 11 Mbytes of memory.

Note: Shoup implemented a variant based on the distinct-degree factorization algorithm (see paper).

Email: `kaltofen@cs.rpi.edu`
URL: `http://www.cs.rpi.edu/~kaltofen`