Complexity Theory in the Service of Algorithm Design

Erich Kaltofen

Rensselaer

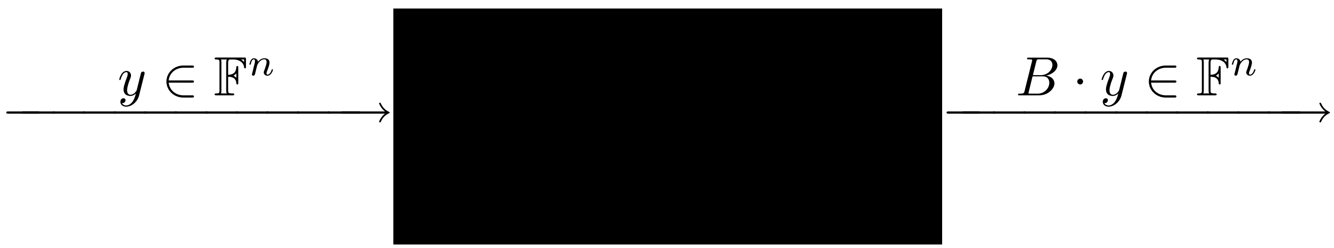
Rensselaer Polytechnic Institute
Department of Computer Science
Troy, New York, USA

Outline

- **Wiedemann's sparse linear system solver**
  - Coordinate recurrences
  - More applications of the transposition principle
- **Reverse mode of automatic differentiation**
  - Transposition principle by derivatives
  - More applications
- **Polynomial factorization**
  - Berlekamp's polynomial factorization algorithm
  - use of the Wiedemann method
  - new baby step/giant step algorithm

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 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 the same computational complexity [Kaminski *et al.*, 1988].

Other Uses:

- Vandermonde$^{\mathrm{tr}} \cdot b$ ("*weighted power sums*") for sparse polynomial interpolation (Canny, Kaltofen, and Lakshman ISSAC '89) and for polynomial factoring (Shoup ISSAC '91).

- Computing the minimum polynomial of an algebraic number in $O(n^2)$ ground field operations (Shoup 1992) by *modular power projection*

- Polynomial factorization and normal bases by *transposed modular polynomial composition* (K and Shoup 1995).

Transposed modular polynomial composition (TCOMP)

Let

$$\mathcal{L}\colon \mathbb{F}[x]/(f) \longrightarrow \mathbb{F}$$
$$c_0 + \cdots + c_{n-1}x^{n-1} \longmapsto c_0u_0 + \cdots + c_{n-1}u_{n-1}, \quad u_i = \mathcal{L}(x^i)$$

be a $\mathbb{F}$-linear map, and let

$$\mathcal{C}[\![h]\!]\colon \mathbb{F}[x]/(f) \longrightarrow \mathbb{F}[x]/(f)$$
$$v_0 + \cdots + v_{n-1}x^{n-1} \longmapsto v(h) \bmod f = \textstyle\sum_l v_l h^l \bmod f$$

Problem: Compute all "power projections"

$$\mathcal{L}(h^i \bmod f) \quad \text{for } 0 \le i < n.$$

Note:

$$\begin{bmatrix} u_0 & \dots & u_{n-1} \end{bmatrix} \cdot C \cdot \begin{bmatrix} v_0 \\ \vdots \\ v_{n-1} \end{bmatrix} = \mathcal{L}(\mathcal{C}[\![h]\!](v))$$

where $C \cdot \begin{bmatrix} v_0 \\ \vdots \\ v_{n-1} \end{bmatrix}$ corresponds to $v(h) \bmod f$.

Application of TCOMP to minimum polynomials

Let $\alpha \in \mathbb{F}[\theta]/(f)$ where θ is algebraic with minimum polynomial f

Problem: Compute the minimum polynomial $g(\alpha) = 0$ where

$$g(x) = x^m - c_{m-1}x^{m-1} - \cdots - c_0 \in \mathbb{F}[x] \qquad \text{with } m \leq n$$

The coefficient vectors $\vec{a}_i = \alpha^i \bmod f(\theta)$ satisfy

$$\forall\, j \geq 0\colon\ \vec{a}_{m+j} = c_{m-1}\vec{a}_{m-1+j} + \cdots + c_0\vec{a}_j$$

Any non-trivial linear projection $\mathcal{L}(\vec{a}_i)$ preserves the linear generator, because g is irreducible

Transposed modular polynomial multiplication (TMULT)

Let $\mathcal{L}\colon \mathbb{F}[x]/(f) \longrightarrow \mathbb{F}$ be a $\mathbb{F}$-linear map, and let

$$\begin{aligned} \mathcal{M}[\![g]\!]\colon \mathbb{F}[x]/(f) &\longrightarrow \mathbb{F}[x]/(f) \\ v(x) &\longmapsto v(x)\cdot g(x) \bmod f(x) \end{aligned}$$

Problem: Compute $\mathcal{L} \circ \mathcal{M}[\![g]\!]$, that is, all

$$\mathcal{L}(\mathcal{M}[\![g]\!](x^i)) = \mathcal{L}(x^i g(x) \bmod f(x)) \quad \text{for } 0 \le i < n.$$

Note:

$$\begin{bmatrix} u_0 & \dots & u_{n-1} \end{bmatrix} \cdot M \cdot \begin{bmatrix} v_0 \\ \vdots \\ v_{n-1} \end{bmatrix} = [\dots \mathcal{L}(\mathcal{M}[\![g]\!](x^i)) \dots] \cdot \begin{bmatrix} v_0 \\ \vdots \\ v_{n-1} \end{bmatrix}$$

where $M \cdot \begin{bmatrix} v_0 \\ \vdots \\ v_{n-1} \end{bmatrix}$ corresponds to $v(x)g(x) \bmod f(x)$.

Baby step/giant step TCOMP (Shoup '94)

$t \leftarrow \lceil \sqrt{n}\, \rceil$

compute $h^2 \bmod f, \ldots, h^{t-1} \bmod f$

$\mathcal{L}^{(0)} \leftarrow \mathcal{L}$

for $j \leftarrow 0$ **to** $\lceil n/t \rceil - 1$ **do**

{**for** $k \leftarrow 0$ **to** $t - 1$ **do**

/* Baby steps */

$\mathcal{L}(h^{jt+k} \bmod f) \leftarrow \mathcal{L}^{(j)}(h^k \bmod f)$

/* Giant steps */

compute $\mathcal{L}^{(j+1)} \leftarrow \mathcal{L}^{(j)} \circ \mathcal{M}[\![h^t \bmod f]\!] = \mathcal{L} \circ \mathcal{M}[\![h^{t(j+1)} \bmod f]\!]$

(by TMULT with h^t from previous $u_i = \mathcal{L}^{(j)}(x^i)$)

} /* end **for** j */

Explicit TMULT algorithm

1. $T_1 \leftarrow \mathrm{FFT}^{-1}(\mathrm{RED}_k(g))$
2. $T_2 \leftarrow T_1 \cdot S_2$
3. $v \leftarrow -\mathrm{CRT}_{0\ldots n-2}(\mathrm{FFT}(T_2))$
4. $T_2 \leftarrow \mathrm{FFT}^{-1}(\mathrm{RED}_{k+1}(x^{n-1} \cdot v))$
5. $T_2 \leftarrow T_2 \cdot S_3$
6. $T_1 \leftarrow T_1 \cdot S_4$
7. Replace T_1 by the 2^{k+1}-point residue table whose j-th column ($0 \le j < 2^{k+1}$) is 0 if j is odd, and is column number $j/2$ of T_1 if j is even.
8. $T_2 \leftarrow T_2 + T_1$
9. $u \leftarrow \mathrm{CRT}_{0\ldots n-1}(\mathrm{FFT}(T_2))$

"we offer no other proof of correctness other than the validity of this transformation technique (and the fact that it does indeed work in practice)" (Shoup)

Analysis of baby step/giant step TCOMP

$\approx 2\sqrt{n}$	modulo f multiplications
$\approx n^2$	additions, multiplications in $\mathbb{F}$

versus

$n - 2$	modulo f multiplications
$\approx n^2$	additions, multiplications in $\mathbb{F}$

Ostrowski, Wolin, and Borisow (1971) circuit transformation

Note that the size of the circuit for partials does not depend on n.

K and Singer 1991: Depth (= parallel time) of circuit for $\partial f / \partial x_i$
= O(depth of circuit for f).

Transformation is numerically stable.

Inverted transposition principle by automatic differentiation

The problems $A^{-1} \cdot b$ and $(A^{\text{tr}})^{-1} \cdot b$, given $A \in \mathbb{F}^{n \times n}$ non-singular and $b \in \mathbb{F}^n$, have the same asymptotic circuit complexity: Let

$$f(x_1, \ldots, x_n) = ([\, x_1 \quad \ldots \quad x_n \,] \cdot (A^{\text{tr}})^{-1}) \cdot b \in \mathbb{F}[x_1, \ldots, x_n].$$

Then

$$\begin{bmatrix} \partial_{x_1} f \\ \vdots \\ \partial_{x_n} f \end{bmatrix} = (A^{\text{tr}})^{-1} b.$$

Note: Transposition principle may not apply due to divisions.

Used for:

- $(\text{Vandermonde}^{\text{tr}})^{-1} \cdot b$ for sparse polynomial interpolation (K and Lakshman '88).

Reduction: Matrix Inverse $\preccurlyeq$ Determinant (Baur, Strassen '83)

Consider a circuit for the determinant,

$$f(a_{1,1}, \ldots, a_{n,n}) = \mathrm{Det}(\begin{bmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \vdots & \vdots & & \vdots \\ a_{n,1} & a_{n,2} & \ldots & a_{n,n} \end{bmatrix}).$$

Then

$$(-1)^{i+j} \frac{\partial f}{\partial a_{j,i}} = \mathrm{Det}(A)(A^{-1})_{i,j}.$$

$\Longrightarrow$ Circuit for partials computes adjoint matrix.

Used for:

- Processor-efficient poly-log parallel computation of A^{-1} (K and Pan '91, '92).
- Division-free computation of adjoint of A in $\widetilde{O}(n^3\sqrt{n}\,)$ arithmetic operations (K '92).

The black box Berlekamp algorithm

Factor squarefree $f(x) = f_1(x) f_2(x) \cdots f_r(x)$, where $n = \deg(f)$, into irreducible polynomials $f_i(x) \in \mathbb{F}_p[x]$, $\mathbb{F}_p$ a finite field with p elements.

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

$\forall i\colon w(x) \bmod f_i(x) = s_i \in \mathbb{F}_p$ (Then $\mathrm{GCD}(f(x), w(x) - s_i) \neq 1$)

$$\Updownarrow$$

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

$$\Updownarrow$$

$$\overrightarrow{w}^{\mathrm{tr}}(Q - I) = 0 \quad \text{where} \quad Q = \begin{bmatrix} \vdots \\ \overrightarrow{x^{ip} \bmod f(x)}^{\mathrm{tr}} \\ \vdots \end{bmatrix}_{i=0,\ldots,n-1}$$

"Black-box matrix" algorithm: 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)

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 $a_i = \overrightarrow{u}^{\mathrm{tr}} \cdot Q^i \cdot \overrightarrow{v} = (\overrightarrow{u}^{\mathrm{tr}} Q^j) \cdot (Q^{tk} \overrightarrow{v})$, 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: $\overrightarrow{u}^{\mathrm{tr}} \cdot Q^j$ 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.

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

Lobo's '94 parallel implementation

Degree n	Prime p	Task	# Computers 8	32	Factor degrees
15001	127	Step W1	$82^h 20'$		1, 1, 2, 2, 4, 12
		Step W2	$12^h 53'$		21, 21, 33, 55
		Step W3	$42^h 42'$		155, 158, 351
		split/refine	$3^h 19'$		809, 1793, 2665
		total time	$141^h 14'$		2813, 2919, 3186
		work	$87577^{\#}$		

Parallel CPU time (hours^{h}minutes$'$) for factoring

$$(x^{7501} + x + 1) \cdot (x^{7500} + x + 1) \pmod{127}$$

on 86.1 MIPS computers; work is measured in MIPS-hours$^{\#}$

Shoup's baby step/giant step 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

Normal bases

Let $\alpha \in \mathbb{F}_p[\theta]/(f)$ where θ is algebraic with minimum polynomial f

α is **normal**

$$\Updownarrow$$

$$\alpha, \alpha^p, \ldots, \alpha^{p^{n-1}} \text{ is an } \mathbb{F}_p\text{-vector space basis for } \mathbb{F}_p[\theta]/(f)$$

$$\Updownarrow$$

$$\vec{\alpha}^{\mathrm{tr}}, \vec{\alpha}^{\mathrm{tr}} Q, \ldots, \vec{\alpha}^{\mathrm{tr}} Q^{n-1} \text{ are linearly independent}$$

$$\Updownarrow$$

$$\exists\, \vec{u}\colon \quad \vec{\alpha}^{\mathrm{tr}} \cdot \vec{u}, \ldots, \vec{\alpha}^{\mathrm{tr}} Q^i \vec{u}, \ldots \text{ is linearly generated by } \lambda^n - 1 \tag{1}$$

Subquadratic algorithms

Basis selection: $\geq \dfrac{1}{12\max\{\log_p n, 1\}}$ pairs $\vec{\alpha}, \vec{u}$ satisfy (1)

(checking for (1) is Step W1 in black box Berlekamp)

Conversion from normal basis: compute $c_0\alpha + \cdots + c_{n-1}\alpha^{p^{n-1}} \bmod f$

(is Step W3 in black box Berlekamp)

Conversion to normal basis: Given γ, find c_i with

$$\gamma = c_0\alpha + \cdots + c_{n-1}\alpha^{p^{n-1}} \bmod f$$

Solve the Hankel system

$$\vec{u}^{\mathrm{tr}} \cdot \overrightarrow{\gamma^{p^j}} = \sum_{i=0}^{n-1} c_i \, \vec{u}^{\mathrm{tr}} \cdot \overrightarrow{\alpha^{p^{i+j}}} \quad (0 \leq j < n)$$