CSE291-14: The Number Field Sieve

https://cseweb.ucsd.edu/classes/wi22/cse291-14

Emmanuel Thomé

February 17, 2022

Part 6c

Sparse linear algebra algorithms

The Lanczos algorithm

The Wiedemann algorithm

Computing the linear generator in Wiedemann

Block algorithms

Plan

The Lanczos algorithm

The Wiedemann algorithm

Computing the linear generator in Wiedemann

Block algorithms

Lanczos

Here we assume $K = \mathbb{F}_\ell$, with ℓ large. "almost characteristic zero".

Lanczos requires a symmetric matrix so we consider $A = M^T M$.

Temporarily inhomogenous

The Lanczos algorithm is easier to state for an inhomogenous linear system, so let $b = Az$ for some random $z \in K^N$. We will solve

$$Av = b$$

from which we will have $A(v - z) = 0$.

A few definitions

Def. Let $y \in K^N$. Krylov subspace $\mathcal{K}_{A,y} = \left\langle y, Ay, \ldots, A^i y, \ldots \right\rangle$.

- $\dim \mathcal{K}_{A,y} \leq N$.
- $\mathcal{K}_{A,y}$ has a known basis.

Def. (pseudo-) scalar product associated to A: $(u, v) \stackrel{\text{def}}{=} u^T A v$.

Note: over a finite field, there are isotropic vectors.

Gram-Schmidt orthogonalization process:

- build an orthogonal basis from an arbitrary one.
- defined in characteristic zero for a real scalar product, but let's see.

GSO in positive characteristic

We take the method for its merits.

- It builds a sequence of vectors with $(e_i, e_j) = 0$ if $i \neq j$.
- We believe for a moment that nothing fails.
- We'll see what might fail and why.

Apply GSO to the basis $(A^i b)_i$ of $\mathcal{K}_{A,b}$. Denote $S_i = \langle b, \ldots, A^i b \rangle$.

$$e_0 \leftarrow b,$$

$$e_j \leftarrow A^j b - \sum_{i<j} \frac{(A^j b, e_i)}{(e_i, e_i)} e_i = A^j b - \sum_{i<j} \frac{b^T A^{j+1} e_i}{e_i^T A e_i} e_i.$$

Prop. $(e_i, e_j) = 0$ if $i \neq j$.
Note that $\langle e_0, \ldots, e_i \rangle = S_i$. Optimization: replace $A^j b$ by $A e_{j-1}$.

Lanczos (cont'd)

$$e_j \leftarrow Ae_{j-1} - \sum_{i<j} \frac{(Ae_{j-1}, e_i)}{(e_i, e_i)} e_i = Ae_{j-1} - \sum_{i<j} \frac{e_{j-1}^T A^2 e_i}{e_i^T A e_i} e_i,$$

Note that
$i < j-2 \Rightarrow Ae_i \in S_{j-2} \subset e_{j-1}^{\perp} \Rightarrow (Ae_{j-1}, e_i) = (e_{j-1}, Ae_i) = 0.$

$$e_j \leftarrow Ae_{j-1} - \frac{(Ae_{j-1}, e_{j-1})}{(e_{j-1}, e_{j-1})} e_{j-1} - \frac{(Ae_{j-1}, e_{j-2})}{(e_{j-2}, e_{j-2})} e_{j-2},$$

$$\leftarrow Ae_{j-1} - \frac{e_{j-1}^T A^2 e_{j-1}}{e_{j-1}^T A e_{j-1}} e_{j-1} - \frac{e_{j-1}^T A^2 e_{j-2}}{e_{j-2}^T A e_{j-2}} e_{j-2}$$

Algorithm. compute this, maintaining $O(1)$ vectors.

What do we have to do ? Examine failure cases.

Lanczos over $\mathbb{F}_\ell$: failure cases

Two possible reasons for stopping:

- We may reach an isotropic (a.k.a. self-orthogonal) vector: $(e_i, e_i) = 0$.
  - We have $(e_i, e_i) = e_i^T A e_i = (Me_i)^T Me_i = 0$.
  - Me_i might be isotropic for the "standard" bilinear form, but heuristically Prob $\approx \frac{1}{\ell}$ only.
- Eventually, we reach $e_i = 0$ at the end. This means success.
  - This implies that $\langle e_0, \ldots, e_{i-1} \rangle = \langle b, Ae_0, \ldots, Ae_{i-1} \rangle$.
  - Let z be a solution to $Az = b$ (z is not known). Let $w = \sum_{j<i} \frac{(e_j, z)}{(e_j, e_j)} e_j = \sum_{j<i} \frac{e_j^T b}{e_j^T A e_j} e_j$.
  - By construction, $\forall j,\ (e_j, w - z) = 0$. Thus $w - z \in \operatorname{Ker} M$ (and $Aw = b$) with proba $\approx 1 - \frac{1}{\ell}$.
  - If we started with $b = Az$ (z known), this gives $w - z \in \operatorname{Ker} M$.

Lanczos: remarks

Note: As is, the Lanczos algorithm does not work over $\mathbb{F}_2$ because for $\ell = 2$, a failure probability of $\frac{1}{\ell}$ at each step is a lot.

Complexity:

- N products $A \times v$,
- hence $2N$ products M (or M^T) times v.

Important (mis-)features:

- Needs fast operations for both M^T and M.
- Must keep track of several vectors.

Plan

The Lanczos algorithm

The Wiedemann algorithm

Computing the linear generator in Wiedemann

Block algorithms

The Wiedemann algorithm

The Wiedemann algorithm for $Mv = 0$ over $\mathbb{F}_p$ is easy.

- Pick $x, y \in \mathbb{F}_\ell^N$ at random.
- Compute $a_i = x^T M^i y$. These are all scalars.
- Compute the generator F of this linear recurring sequence.
- $\hat{F}$ divides the minimal polynomial μ_M. Hope $X^\lambda \hat{F} = \mu_M$.
- We then have $M^\lambda \hat{F}(M)y = 0$. Which means $M^{\lambda-1}\hat{F}(M)y \in \operatorname{Ker} M$.

This is very accessible to proofs of success probabilities.

The Wiedemann algorithm: workflow

Implementation of the Wiedemann algorithm is fairly straightforward.

- Computation of the sequence of a_i.
- Computation of the linear generator F.
- Computation of the kernel vector.

The sequence of a_i

- $i \leftarrow 0$;
- $v \leftarrow y$;
- While $i < 2N$.
  - $a_i \leftarrow x^T v \in \mathbb{F}_\ell$.
  - $v \leftarrow Mv$;
  - $i \leftarrow i + 1$.
- return $(a_i)_i$, sequence of $2N$ elements of $\mathbb{F}_\ell$.

Cost

To compute $2N$ terms, we need:

- Exactly $2N$ matrix-times-vector products.
- If the weight of M is W, this means $\approx 2N \times W$ operations.
  here, operation $=$ addition in $\mathbb{F}_\ell$.

The linear generator

Linear generator

The linear generator of the sequence is such that:

$$\forall i \geq d, F_0 a_i + F_1 a_{i-1} + \cdots + F_d a_{i-d} = 0.$$

Note. The set of polynomials $\sum_{i=0}^{d} F_i X^i$ is an ideal of $\mathbb{F}_\ell[X]$, and $\hat{\mu_M}$ belongs to it. So $d \leq N$.

The linear generator

Another point of view

Let $A(X) = \sum_{i \le 2N} a_i X^i$, then:

$$A(X)F(X) = (\text{terms of deg} < N) + (\text{terms of deg} \ge 2N).$$

By construction, there is an infinite precision solution to $(\sum a_i X^i)F(X) = G(X)$, and looking at precision $2N$ will be sufficient to find it.

Several possible restatements ($\deg F \le N$ and $\deg G < N$):

- $A(X)F(X) - X^{2N}R(X) = G(X)$.
- $A(X) = \dfrac{G(X)}{F(X)} + O(X^{2N})$.
- $A(X)F(X) = G(X) + O(X^{2N})$.

$O(X^i)$ means X^i times any polynomial in $\mathbb{F}_\ell[X]$

Computing the linear generator

Various algorithms can be used to compute F.

- The Berlekamp-Massey algorithm (from coding theory).
- The Euclidean algorithm!

We have several ways to do this in time $O(N^2)$ or even $O(N \log^2 N)$. More on this later.

Probabilistic aspect

We hope that we'll find a generator F which is such that $X^\lambda \hat{F} = \mu_M$. with $\lambda \geq 1$.

Reconstructing the solution

To compute $\hat{F}(M)y$, the process is similar to the first phase:

- $k \leftarrow 0$;
- $v \leftarrow y$;
- $w \leftarrow 0$;
- While $k \leq \deg F$;
  - $w \leftarrow w + v \times$ (coefficient of degree k in $\hat{F}(X)$);
  - $v \leftarrow Mv$;
  - $k \leftarrow k + 1$.
- return w.

Cost

N matrix-times-vector products.

Wiedemann: summary

The Wiedemann algorithm costs about $3N$ matrix-times-vector products.

Probability of failure is $O(1/\ell)$.
(main failure case: $\nu_X(\mu_M) = 1$, $\dim \mathrm{Ker}\, M = 1$, and $y \in \mathrm{Im}\, M$).

Comparison with the Lanczos method

The Wiedemann algorithm:

- costs $3N$ matrix-times-vector products.
- has a three-stage workflow which is a little bit more complicated than the Lanczos algorithm.

The Lanczos algorithm (not described):

- costs only $2N$ matrix-times-vector products.
- is comparatively slightly simpler.

Neither is really usable over $\mathbb{F}_2$.

Plan

The Lanczos algorithm

The Wiedemann algorithm

Computing the linear generator in Wiedemann

Block algorithms

Linear generator

The problem of computing the linear generator is central in the Wiedemann algorithm.

Next few slides: a brief review of how we can do in quasi-linear time, with a view towards a possible generalization.

Problem

Problem statement

Given $A \in \mathbb{F}_\ell[X]$ with $\deg A < 2N$, find $F, G \in \mathbb{F}_\ell[X]$ such that:

- $\deg F \leq N$ and $\deg G < N$. IOW, $\max(\deg F, 1 + \deg G) \leq N$.
- $A(X)F(X) = G(X) + O(X^{2N})$.

We may look at the linear algebra point of view.

- Degrees of freedom: $N + 1$ (coefficients of F).
- Constraints: N (coefficients of degree N to $2N - 1$).

But of course we can do much better than $O(N^3)$ here!

Fixed versus infinite precision

The series $A(X)$ is a truncation (to degree $2N$) of the series $\sum a_i X^i$.

By construction, $(a_i)_i$ is linearly generated with a generator of degree at most N.

The Berlekamp-Massey algorithm finds this generator $F(X)$.
If we ever attempt to compute $A(X)F(X)$ with more terms of the series $A(X)$, we will see that the trailing terms are zero!

Berlekamp-Massey vs Euclid

While we often look at the problem with high degrees first (Euclid), the Berlekamp-Massey presentation (low degrees first) generalizes much better.

Berlekamp-Massey point of view

- Form solutions to $A(X)F(X) = G(X) + O(X^t)$, for increasing values of t (starting with $t = 1$).
- We work with two candidates at a time.
  $F(X)$ and $G(X)$ are extended to matrices.
- The value $t = 2N$ is the target of this process.
- Do so in a way that $\max(\deg F, 1 + \deg G)$ does not grow too fast (not as fast as t).

Example

Let $N = 4, \ell = 17$, and $A = 2 + 5X + 3X^2 + X^3 + \cdots$.

We work with two candidates (two series in $\mathbb{F}_\ell[[X]]$).

$$\begin{pmatrix} 1 \\ X \end{pmatrix} \cdot A = \begin{pmatrix} 2 \\ 0 \end{pmatrix} + \begin{pmatrix} 5 + 3X + X^2 + \cdots \\ 2 + 5X + 3X^2 + \cdots \end{pmatrix} \cdot X$$

$$\begin{pmatrix} 1 \\ X + 3 \end{pmatrix} \cdot A = \begin{pmatrix} 2 \\ 6 \end{pmatrix} + \begin{pmatrix} 5 + 3X + X^2 + \cdots \\ 0 - 3X + 6X^2 \cdots \end{pmatrix} \cdot X$$

$$\begin{pmatrix} X \\ 3 + X \end{pmatrix} \cdot A = \begin{pmatrix} 2X \\ 6 \end{pmatrix} + \begin{pmatrix} 5 + 3X + \cdots \\ -3 + 6X + \cdots \end{pmatrix} \cdot X^2$$

$$\begin{pmatrix} 5 - 3X \\ 3X + X^2 \end{pmatrix} \cdot A = \begin{pmatrix} 2X - 7 \\ 6X \end{pmatrix} + \begin{pmatrix} -4 + \cdots \\ -3 + \cdots \end{pmatrix} \cdot X^3.$$

Berlekamp-Massey

At each step, we decide on the linear combination to use based on the degree t coefficients on the right-hand side.

- Which row we add to the other depends on which is smallest with respect to $\max(\deg F, 1 + \deg G)$.
- This smallest row is eventually multiplied by X, while the degree of the other is unchanged.
- On average $\max(\deg F, 1 + \deg G)$ grows like $t/2$.
- Complexity: N steps, $O(N)$ at each step, so $O(N^2)$.

Berlekamp-Massey

Key aspects

The computation involves matrices of polynomials.
The control flow is directed by the knowledge of:

- the knowledge of $\max(\deg F, 1 + \deg G)$ for each candidate.
- the error matrix $E(X) = (A(X)F(X) - G(X)) \operatorname{div} X^t$

The output is a matrix of polynomials $\pi(X)$ that encodes the necessary transformations to move from the pair of solutions (F, G) at $t = 1$ to the pair of solutions at some larger value of t.

Berkekamp-Massey, recursively

- Compute the initial error matrix $E(X)$.
- Truncate $E(X)$ to degree N (=half of $2N$).
- Recurse and find a matrix such that $\pi(X)E(X) = O(X^N)$. keep track of $\max(\deg F, 1 + \deg G)$ for each candidate.
- Multiply $\pi(X)$ by the full $E(X)$, get coefficients of degrees N to $2N - 1$. (middle product)
- Recurse and find a second matrix $\pi'(X)$.
- Compute $\pi'(X) \cdot \pi(X) \cdot \begin{pmatrix} 1 \\ X \end{pmatrix}$. (polynomial product)

Benefit: complexity is driven by large polynomial multiplications, doable in quasi-linear time.
The complexity of the linear generator step becomes $\widetilde{O}(N)$.

Plan

The Lanczos algorithm

The Wiedemann algorithm

Computing the linear generator in Wiedemann

Block algorithms

Block algorithms

Two popular block algorithms, with block size n:

- Block Lanczos (BL). $\frac{2N}{n-0.76}$ black box applications (for $\ell = 2$);
- Block Wiedemann (BW). In its simplest form: $\frac{3N}{n}$.

There are, however,

- multiple aspects beyond just this computational cost
- and multiple ways to parameterize BW, which end up modifying the picture a lot.

Montgomery's block Lanczos algorithm

BL (Montgomery) is a terrible mess, notationally speaking.

Key idea:

- Try to "orthogonalize" a sequence of subspaces of dim $= n$.
- When ℓ is small, the dimension of our subspaces may decrease in the process. (whenever we hope to find n new vectors, we find only $n - 0.76$ on average when $\ell = 2$.)

Problem with BL

- The procedure we have given does build a nice sequence of spaces, until it collapses.
- rank(W_i) decreases slowly to 0.

$$V_0 \begin{cases} \to \mathcal{W}_0, \text{ dimension } n_0 \leq n \\ \to n - n_0 \text{ vectors dropped} \end{cases}$$

$$V_1 = AW_0 \begin{cases} \to \mathcal{W}_1, \text{ dimension } n_1 \leq n_0 \\ \to n_0 - n_1 \text{ vectors dropped} \end{cases}$$

$$V_2 = AW_1 \begin{cases} \to \mathcal{W}_2, \text{ dimension } n_2 \leq n_1 \\ \to n_1 - n_2 \text{ vectors dropped} \end{cases}$$

Problem with BL

- The procedure we have given does build a nice sequence of spaces, until it collapses.
- rank(W_i) decreases slowly to 0.

$$V_0 \longrightarrow \mathcal{W}_0, \text{ dimension } n_0 \leq n$$
$$\longrightarrow n - n_0 \text{ vectors dropped}$$

$$V_1 = AW_0 \longrightarrow \mathcal{W}_1, \text{ dimension } n_1 \leq n_0$$
$$\longrightarrow n_0 - n_1 \text{ vectors dropped}$$

$$V_2 = AW_1 \longrightarrow \mathcal{W}_2, \text{ dimension } n_2 \leq n_1$$
$$\longrightarrow n_1 - n_2 \text{ vectors dropped}$$

What makes BL work

Solution to the problem: reinject vectors from previous steps to make the thing work.

It is possible to obtain a recurrence equation with small depth, but presenting it is really painful.
$\Rightarrow$ I'm deliberately skipping details here.

Various presentations: Montgomery (1995), Montgomery & Elkenracht-Huizing (1996), T. (2017).

Limitations of the block Lanczos algorithm

The BL algorithm does not offer a huge lot of parameterization opportunities.

- If one wants to involve multiple cores and nodes, all have to participate in the same matrix-times-vector product at each iteration.
- The implementation must keep track of a significant number of vectors, and does dot products at each iteration.
- AFAIK, there is no known mechanism to quickly validate some intermediary checkpoint data.