Block Wiedemann likes Schirokauer maps

E. Thomé

INRIA/CARAMEL, Nancy.

Oct. 2nd, 2015

Plan

General context

The main application target for this talk is

- the computation of discrete logarithms
- ... in large or medium characteristic finite fields
- ... using using the Number Field Sieve or its variants.

More specifically, we present a practical improvement to the linear algebra step of NFS-DL.

Throughout the talk, we consider the DLP problem in a subgroup of prime order ℓ within $\mathbb{F}_p^*$.

Relations in NFS

$$
\begin{tikzcd}
 & \mathbb{Z}[x] \arrow[dl, "\psi^{(1)} : x \mapsto m"'] \arrow[dr, "\psi^{(2)} : x \mapsto \alpha"] & \\
\mathbb{Z}[m] \arrow[dr, "\varphi_g : t \mapsto t \mod p"'] & & \mathbb{Z}[\alpha] \arrow[dl, "\varphi_f : \alpha \mapsto m \mod p"] \\
 & \mathbb{F}_p &
\end{tikzcd}
$$

Relations in NFS

$$\begin{tikzcd}[column sep=small]
& a - bx \in \mathbb{Z}[x] \arrow[dl, "\psi^{(1)} : x \mapsto m"'] \arrow[dr, "\psi^{(2)} : x \mapsto \alpha"] & \\
a - bm \in \mathbb{Z}[m] \arrow[dr, "\varphi_g : t \mapsto t \mod p"'] & & \mathbb{Z}[\alpha] \ni a - b\alpha \arrow[dl, "\varphi_f : \alpha \mapsto m \mod p"] \\
& \mathbb{F}_p &
\end{tikzcd}$$

same element

NFS collects many "good pairs" (a, b) such that:

- the integer $a - bm$ is smooth: product of small primes;
- the ideal $a - b\alpha$ is a product of small prime ideals.

Combining relations

NFS-DL can combine together (multiply) many $(a - bx)$.

- See what happens multiplicatively on both sides;
- Gain knowledge about logarithms in our subgroup of order ℓ.

First task:

- Which kind of objects are we looking at on both sides ?
- Which knowledge do we get ?

Prelude: introduce virtual logs and Schirokauer maps.

Multiplying things

Fact 1: being smooth is a multiplicative property

If $a_0 - b_0 m$ and $a_1 - b_1 m$ are smooth, so is $(a_0 - b_0 m) \cdot (a_1 - b_1 m)$.
Same on the other side.

$$\begin{tikzcd}
 & \prod_i (a_i - b_i x)^{\epsilon_i} \arrow[dl] \arrow[dr] & \\
\prod_i (a_i - b_i m)^{\epsilon_i} \arrow[dr] & & \prod_i (a_i - b_i \alpha)^{\epsilon_i} \arrow[dl] \\
 & \text{subgroup of } \mathbb{F}_p^* \arrow[d, "x \mapsto x^{(p-1)/\ell}"] & \\
 & \text{our subgroup of order } \ell \text{ within } \mathbb{F}_p^* &
\end{tikzcd}$$

Multiplying things

Fact 1: being smooth is a multiplicative property

If $a_0 - b_0 m$ and $a_1 - b_1 m$ are smooth, so is $(a_0 - b_0 m) \cdot (a_1 - b_1 m)$. Same on the other side.

$$
\begin{tikzcd}[column sep=tiny]
 & \text{subgroup of } \mathbb{Q}(x)^* \arrow[dl] \arrow[dr] & \\
\text{subgroup of } \mathbb{Q}^* \arrow[dr] & & \text{subgroup of } \mathbb{Q}(\alpha)^* \arrow[dl] \\
 & \text{subgroup of } \mathbb{F}_p^* \arrow[d, "x \mapsto x^{(p-1)/\ell}"] & \\
 & \text{our subgroup of order } \ell \text{ within } \mathbb{F}_p^* &
\end{tikzcd}
$$

Taking out powers

Our two subgroups of smooth things are important.

Fact 2: ℓ-th powers map to 1 eventually

What are exactly the subgroups of interest on both sides ?

$$\{\text{smooth things}\}/\{\text{smooth things}\}^{\ell}.$$

- vector spaces (fact 1);
- defined over $\mathbb{F}_\ell$ (fact 2);
- finite dimensional:
  - smooth rationals (mod ℓ-th powers) determined by ...
  - smooth algebraic numbers (mod ℓ-th powers) determined by ...

Describing smooth elements

Smooth rationals (mod ℓ-th powers): Σ_r

They are simply determined by valuations at small primes mod ℓ.

- A bound on "small primes" is set beforehand.
- Units in $\mathbb{Z}$ are just ± 1: trivial modulo ℓ-th powers.

Smooth algebraic numbers (mod ℓ-th powers): Σ_a

- Need valuations at small prime ideals mod ℓ.
- Torsion units are harmless;
- But non-torsion units lead to (finite-dimensional) ambiguity.
  - The map $\Sigma_a \to \{\nu_{\mathfrak{p}}(\cdot) \bmod \ell\}$ is not injective.
  - Fix: use $\Sigma_a \to \{\nu_{\mathfrak{p}}(\cdot) \bmod \ell\} + \{\text{Schirokauer maps}\}$.

The log map

NFS-DL lifts the log to a linear form coming from smooth things.

$$
\begin{array}{c}
\Sigma_r \times \Sigma_a \subset \mathbb{F}_\ell^{\#\mathcal{F}_r + \#\mathcal{F}_a + \text{unit rank}} \\
\downarrow \\
\text{subgroup of } \mathbb{F}_p^* \text{ (mod } \ell\text{-th powers)} \\
\downarrow \; x \mapsto x^{(p-1)/\ell} \\
\text{our subgroup of order } \ell \\
\downarrow \; x \mapsto \log x \\
\mathbb{F}_\ell
\end{array}
$$

The log map

NFS-DL lifts the log to a linear form coming from smooth things.

$$\begin{array}{c} \Sigma_r \times \Sigma_a \subset \mathbb{F}_\ell^{\#\mathcal{F}_r + \#\mathcal{F}_a + \text{unit rank}} \\ \Big| \\ \boxed{\text{Linear form } \Phi} \\ \Big\downarrow \\ \mathbb{F}_\ell \end{array}$$

The log map

NFS-DL lifts the log to a linear form coming from smooth things.

$$
\begin{array}{c}
\Sigma_r \times \Sigma_a \subset \mathbb{F}_\ell^{\#\mathcal{F}_r + \#\mathcal{F}_a + \text{unit rank}} \\
\Big\downarrow \ \text{Linear form } \Phi \\
\mathbb{F}_\ell
\end{array}
$$

- Relations help to nail down Φ. (kernel elements!)
- Seek coordinates of Φ; a.k.a. virtual logarithms.
- This reduces to a linear algebra problem $Mw = 0$.

Our concern: the presence of Schirokauer maps in the relations.

Plan

The linear system

We have many relations = rows of a matrix M. We want to solve

$$Mw = 0.$$

The solution vector is the set of virtual logarithms.

- By construction, a non-zero solution exists.
- If we have sufficiently many relations, it is unique.
- In practice, it may happen the set of solutions of $Mw = 0$ has dimension slightly more than 1, generated by:
  - The good Φ;
  - plus some small hamming weight vectors, quite harmless.

Bottom line: any non-zero solution to $Mw = 0$ is good to go.

Relations with SM

Why are Schirokauer maps here ?

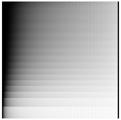
- Because they are a key part of a coordinate system for Σ_a.

Why are Schirokauer maps annoying ?

- extra coordinates in each row of M (number $=$ unit rank);
- full-size integers mod ℓ, much larger than valuations.

```
115 7:2 1:-6 0:-9 -2 4:3 3:6 5:2 10:-2 9:-2 13:-3 14:2 21:-2 17:2 -16 -20 26:-2 22 24:2 -25 30 28 -35 34 -33 39 -45 43 -50 -62 -68 80 67 -77 -79 82 105 88 -107 -92 110 147 129 -162
  158 -223 179 -209 189 -169 -208 220 172 -259 248 257 -339 287 -322 317 494 413 447 468 464 460 -574 -722 1389 -1546 616 757 911 859 -1597 -3002 -1203 3005 1499 -1563 2388 -2283
  1164 3534 -4599 4855 1307 1504 3130 3779 3336 -2688 3337 -1910 4767 4038 4636 -4149 -2240 5300 3156 5421 5426 3478 -4799 5451 2890 -2307 2633 1915 -3732 2197 1314 -2066 1517 -4709
   1918902163910971358342249522003507866580463311636550391 587691104072599510196942808794364879161331978889856821 1962472614102761699352739429726112125956851202601753243
157 -7 1:3 0:-2 2:-2 8:-2 4 5:-6 6:2 11 10:-3 9:2 -13 12 14:2 21:-2 17:2 19:-4 18 20:3 15:5 26 22:-2 24:2 -25 -29 28 32 -35 -31 36:-2 -52 -44 -46 47 -51 55 -57 -56 -63 66 58 72:-2
  78 -76 67 -77 -85 -103 -93 89 -92 -91 -117 115 -127 125:2 137 146 155 -190 -196 202 188:2 203 -326 243 275 -321 219 -363 432 -405 393 387 528 -304 -378 1056 564 -599 480 -759 -486
  -559 -527 440 701 -941 807 608 741 -703 789 2072 921 -493 -734 -2651 -1117 887 1050 -1052 -840 -1195 -1041 -1371 2654 -2031 1559 -2089 2330 1940 2211 -1206 1383 1735 -1770 1328
  -1774 -1814 -5506 -2096 -2045 -2946 -1443 -2769 3032 -1537 6196 -3337 -4146 -4382 -3806 6220 3702 4938 -3598 -3818 -5318 -3374 -5453 5939 2110 2553 -3491 2062 -2807 -1625 -2570
  -1705 -3735 861 2726 1214 -3190 1629 -2912   112741066656439527085314348557406601769538491399985266 1389257887743132560050638908235896167973209639160674338
  2034480659967088911878326251186212823038149619256053339
124 -7 1 0:2 2 8 -4 3:-2 5:-4 6:4 11:-2 -10 13 14:2 -21 17 16:2 -18 -15 27:3 24:-2 -28 32:-2 -35 34 -31 -37 36 40 41:3 46:-2 -57 -66 -69 -80 -78 76 71 -73 86 -87 96 -91 -110 -98
  -150 -125 181 -157 -156 -155 -165 -158 -178 -213 -191 -170 -215 188 -241 263 272 -308 356 219 -357 -270 382 449 -409 -525 -552 -697 -655 -549 1115 -733 -932 876 -1671 1393 618
  -2080 -2030 -926 -4223 4329 -1601 1094 -1074 -4856 -2939 4110 2943 -2402 5013 3557 2948 -5379 4369 -5161 5173 -3149 2000 4528 -4657 6393 2109 -4545 4965 6050 -3166 5211 2556 -3611
  3264 -5220 -2118 2119 -1046 2724 -2574 2575 -2992 -4092  10482311835115361986151353798071031967364125970466455151 1637498222607624361471047203748763659968941246535430330
  1979115440994922488817888530191549492933831869476915978
67 7 1 0:2 2:-2 -8 4:3 5:4 9:-2 -14 21:2 17:-2 -16 26:2 23 30 -35 -40 43 -42 -46 -48 60 56 -80 78 73 87 93 -89 122 123 131 142 -157 166 168 -200 202 -197 265 243 364 -317 -486 1117
  -876 767 1274 860 -4224 3655 2288 1740 2850 -1505 4115 -4640 2963 -4646 4802 -4412 6055 2245 -2362 3749 -4093 2730  1469550753304343111452953100100368813708772237727267738
  1283287581040594781486025452041390384261817830179664397 2064834356908888032261261330546281386296771592938000965
141 7 1:-5 0:-8 8 4:2 3:6 5:-2 6 -11 10:-6 9:4 -12 14:2 -21 17:2 16 19 18:-2 20:-2 26 23:-2 22 30:-2 32 35:2 31 37 36:2 47:-2 49 -55 -65 60 62 -58 61 -69 -70 85 79 105 -103 -95
  -102 96 -106 110 112 113 117 120 -126 139 141:2 -128 135 148:2 144 -140 134 142 164 161 154 166 159 -155 175 167 183 -191 -209 206 -247 -276 405 342 366 -305 288 -373 -528 -362 599
  581 689 -760 634 -654 -762 -608 562 -498 845 777 609 1418 -543 -1421 -737 1550 980 2272 940 -1763 -5364 -1110 -1257 1464 3211 -1991 5850 2940 -2676 -4733 -4611 -4742 -2606 -4620
  2102 6294 -3153 -4792 -3478 4817 4818 -3718 3064 -2251 -2899 1830 -2560 3959 1963 1964 -5956 5480 -2645 2261 -1709 -2452
  1950344097105955355681548645846701554618683633557545639 1003758221202475279031671890106882129030050564204062245 1367801815154152297425303265007561885189049816683159375
49 7 1:-2 0:-11 2 8 4:4 3:2 5 11 10:-2 9 13 17:2 19 -26 23 31 47 54 55 56 -75 84 91 140 -210 221 253 358 639 605 518 -436 -521 2458 -912 1197 1772 -3117 3665 1309 3790 -5752 3475
  -5694 3062 -4567 2648 2823  391647361048377121178237271597689366106770963115064670 183028351918043150986925271814127289629480033189928399 6
  196885450478721390927427269646235342796732758310643336
```

What does the matrix look like ?



The matrix M is large and sparse.

- N rows and columns;
- Want to solve $Mw = 0$ over $\mathbb{F}_\ell$, with roughly 200-bit ℓ;
- d dense columns (Schirokauer maps; same size as ℓ);
- Other coefficients (typically $c \approx 100$ per row) are all < 10.

We use sparse linear algebra techniques.

- Touching the matrix is forbidden (want to avoid densification);
- Rely only on the matrix times vector operation.

$$v \longrightarrow \boxed{M} \longrightarrow M \times v$$

SM overhead for doing $v \to M \times v$

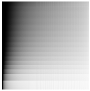
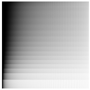
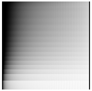
Coefficients of v are integers mod ℓ.

Cost for each coefficient of $M \times v$

- about 100 multiplications (tiny coeff of M $\times$ coeff of v) (more than 90% of the time, tiny means ± 1);
- about d multiplications (SM coeff $\times$ coeff of v);
- about $100 + d$ additions;
- one reduction modulo ℓ.

- The multiplications by SM coeffs are not negligible in practice.
- Because of them, some alternative representation formats are hampered or ineffective (RNS), or we have to take into account conversion costs.

Related linear algebra problems

- Factoring : solve $w \times$  $= 0$ over $K = \mathbb{F}_2$;
- FFS of old : solve  $\times\, w = 0$ over $K = \mathbb{F}_\ell$ (obsolete);
- NFS-DL : solve  $\times\, w = 0$ over $K = \mathbb{F}_\ell$.

Sad thing

The SM columns would be harmless if we were to solve $wM = 0$. For NFS-DL, not.

- no such thing as a "partial solution" which we can complete.
- for the p180 record, significant cost from SM columns.

Homogeneous vs inhomogeneous

We can as well write:

$$M \times w = \left(\boxed{\quad M_0 \quad} \boxed{b} \right) \times w = 0.$$

M_0 sparse of size $N \times (N-d)$. Dense SM block b of size $N \times d$.

- We look for one solution vector w: size $N \times 1$.
  $\Rightarrow$ a priori knowledge that a solution space exists.
- NOTE: if $d = 1$, this amounts to solving $M_0 w_0 = b$.
  $\Rightarrow$ knowledge that $b \in \operatorname{Im}(M_0)$. Can pad M_0 to square.

How does sparse LA do this ? How to expand to $d > 1$?

Plan

Several sparse algorithms

Solving sparse linear systems over finite fields (hence exact) often done with the following black-box algorithms:

- Lanczos algorithm (1950);
- Wiedemann algorithm (1986);
- their block variants: Block Lanczos (Montgomery, 1995), Block Wiedemann (Coppersmith, 1994).

Desired properties: complexity, parallelization, distribution.

Wiedemann algorithm

Let x and y be an arbitrary vectors in K^N.

The Wiedemann algorithm computes $(a_i)_i$, with $a_i = x^T M^i y \in K$.

Rational reconstruction on $A(X) = \sum_{i=0}^{2N} a_i X^i$

Find $F(X)$ and $G(X)$ such that:

$$A(X)F(X) = G(X) + O(X^{2N}),$$
$$\deg F \leq N \quad \deg G < N.$$

The N zero coefficients in the middle of the RHS rewrite as:

$$\forall k \in [0, N-1], \quad x^T M^k \widehat{F}(M) y = 0.$$

Unless disaster occurs, this means $\widehat{F}(M)y = 0$.

Wiedemann algorithm

Wiedemann for inhomogeneous system:

To solve $Mw = b$, Wiedemann sets $y = b$, and x random.
We hope that $\widehat{F}$ has non-zero constant coefficient: $\widehat{F} = 1 - XQ$.
This implies $M \cdot (Q(M)b) = b$. Found solution $w = Q(M)b$.

Wiedemann for homogeneous system:

Simple strategy: set $y = Mz$. Then $M \cdot (\widehat{F}(M)z) = 0$.
Found solution $w = \widehat{F}(M)z$.

(alternatively, compute F from $A(X) \bmod X$)

Bottom line: non-block Wiedemann adapts to both.
Correctness ? In order to defend against degeneracy mishaps, preconditioning *might* be required.

Plan

Block Wiedemann

Invented by Coppersmith (1994), for the factoring context.

Replace black box by matrix times block of vectors.

The black box (BB) becomes a block black box (BBB).

- Replace x and y by vector blocks $\mathbf{x}, \mathbf{y} \in K^{N \times n}$.
- Expect that fewer black box calls are required.
- Very well adapted to the $K = \mathbb{F}_2$ case.
  Good distribution opportunities;
  Can also be used for DLP with $K = \mathbb{F}_\ell$.

Block Wiedemann

Let $\mathbf{x}$ and $\mathbf{y}$ be an arbitrary vector blocks in $K^{N\times n}$.
Compute $(\mathbf{a}_i)_i$, with $\mathbf{a}_i = \mathbf{x}^T M^i \mathbf{y} \in K^{n\times n}$.

Hermite-Padé approximation on $\mathbf{A}(X) = \sum_{i=0}^{2N/n} \mathbf{a}_i X^i$

Find $\mathbf{F}(X)$ and $\mathbf{G}(X)$ such that:

$$\mathbf{A}(X)\mathbf{F}(X) = \mathbf{G}(X) + O(X^{2N/n}),$$
$$\deg \mathbf{F} \leq N/n \quad \deg \mathbf{G} < N/n.$$

Algorithms: Beckermann-Labahn (1994), T. (2001).

The N/n zero coefficients in the middle of the RHS rewrite as:

$$\forall k \in [0, N/n - 1], \quad \mathbf{x}^T M^k \mathbf{THING} = 0.$$

Means **THING** is orthogonal to $n \times N/n$ columns of $\{(M^T)^k \mathbf{x}\}$.
Unless disaster occurs, this means $\mathbf{THING} = 0$.

What is this THING ?

The Hermite-Padé approximation computes $\mathbf{F} \in K[X]^{n \times n}$.

- As in the non-block case, $\widehat{\mathbf{F}}$ is related to the min.poly. of M.
- Actually $\det \widehat{\mathbf{F}}$ is "close to" μ_M.

Let $\mathbf{F} = (F_{i,j})_{1 \leq i,j \leq n}$ and columns of $\mathbf{y}$ be $(y_1 \dots y_n)$.

$$\begin{aligned}\text{column 1 of } \mathbf{THING} &= \widehat{F_{1,1}}(M)y_1 + \cdots + \widehat{F_{n,1}}(M)y_n,\\ \text{column 2 of } \mathbf{THING} &= \widehat{F_{1,2}}(M)y_1 + \cdots + \widehat{F_{n,2}}(M)y_n,\\ &\dots\end{aligned}$$

Conclusion: **THING** is made of n distinct expressions, all evaluating to zero.

Columns of $\mathbf{F}$

column 1 of **THING** $= \widehat{F_{1,1}}(M)y_1 + \cdots + \widehat{F_{n,1}}(M)y_n$.

Solving $Mw = 0$:

- take $\mathbf{y} = M\mathbf{z}$ for a random z.
- each column of **THING** gives a solution.
- needs N/n extra BBB calls.

Correctness: same as non-block, but harder.

In practice, we only want a select number of solutions. Use this many columns of $\mathbf{F}$.

- For RSA-768, maybe fetching 512 solutions was overkill. (not embarrassing, since computational excess is negligible).
- DLP: one will be good enough. Better not do more. Because of SM, we have our d annoying dense columns.

Plan

Homogeneous vs inhomogeneous

Recall that in the non-block case, for $d = 1$, we can solve the inhomogeneous linear system, then the SM column disappears.

Can we do the same in the block case ?

SMs within $\mathbf{y}$

Assumption from now on $n \geq d$.

- First d columns of $\mathbf{y}$ are chosen as $\mathbf{b}$.
- Last $n - d$ chosen as $M\mathbf{z}$ for a random $\mathbf{z} \in K^{N \times (n-d)}$.
- **Erase** the d dense columns in M. Call that M_0
  From now on, M_0 is $N \times N$, but has d zero columns.

Run Block Wiedemann on this.

- expect smaller cost for each matrix times vector operation;
- exact same cost everywhere else, provided that we are able to work with any column of $\widehat{\mathbf{F}}$.

Writing down solutions

How can we use one of the **THING** $= 0$ equations ?

$$\text{column 1 of } \textbf{THING} = \widehat{F_{1,1}}(M_0)y_1 + \cdots + \widehat{F_{n,1}}(M_0)y_n.$$

For $i \leq d$, let $\widehat{F_{i,1}} = c_i + XQ_i$; for $i > d$, let $\widehat{F_{i,1}} = Q_i$

$$0 = c_1 b_1 + \cdots + c_d b_d + M_0 \cdot (\sum_i Q_i(M_0)z_i).$$

Deriving a solution to $Mw = 0$

Set $N - d$ first columns of w to be those of $\sum_i Q_i(M_0)z_i$;
Set d last columns to be $(c_1, \ldots c_d)$.

- Overhead from SM columns is eliminated, provided $n \geq d$.
- Implemented in CADO-NFS since nov. 2014.

Change in cost

Cost impact analyzed by Joux and Pierrot.

- Assume c non-zero coefficients per row in M_0, and d SMs.
- Take SM $\times v_j$ to cost β times more than $m_{ij} v_j$.

With SMs in the matrix: $\boxed{3(c+\beta d)N^2 + \kappa n^2 N \log^2 N}$

- $\mathbf{A}(X)$: $3N/n$ BBB calls: $3(c+\beta d)N^2$, or n-fold distributed.
- $\mathbf{F}(X)$: $\kappa n^2 N \log^2 N$ for some κ.

Now with SMs in $\mathbf{y}$: $\boxed{3cN^2 + \kappa \min(n,d)^2 N \log^2 N}$

- $\mathbf{A}(X)$: $3N/n$ BBB calls: $3cN^2$, or n-fold distributed.
- $\mathbf{F}(X)$: same, but recall we want $n \geq d$.

Plan

Is that new ?

Short answer: NO.

MATHEMATICS OF COMPUTATION
VOLUME 62, NUMBER 205
JANUARY 1994, PAGES 333–350

SOLVING HOMOGENEOUS LINEAR EQUATIONS OVER $GF(2)$ VIA BLOCK WIEDEMANN ALGORITHM

DON COPPERSMITH

Inhomogeneous equations. We developed this algorithm for homogeneous equations, because that is the case of interest for integer factorization. For the inhomogeneous system of equations $B\mathbf{w} = \mathbf{b}$, where $\mathbf{b}$ is a block of at most n vectors, variants that can be tried include the following:

1. Set the first few columns of $\mathbf{y}$ equal to $\mathbf{b}$, and calculate the rest of $\mathbf{y}$ as $B\mathbf{z}$. Then hope that in the equation

$$\mathbf{x}_\mu^{\mathrm{T}} B^{j-d'} \sum_{\nu,k} f_{l,\nu}^{(l,k)} B^{d'-k} \mathbf{y}_\nu = 0$$

the coefficients of $B^0\mathbf{y}_\nu$ form an invertible matrix, allowing one to solve for $\mathbf{y}$ in terms of vectors in the image of B.

Longer answer: let's see why.

What Coppersmith is doing

Coppersmith aims at solving $\mathbf{Mw} = \mathbf{b}$.

In effect this means d independent one-vector systems.

Claim: this solves a harder problem. Ours is an easy by-product.

How does Coppersmith do this ? As we do.

- First d columns of $\mathbf{y}$ are chosen as $\mathbf{b}$.
- Last $n - d$ chosen as $M\mathbf{z}$ for some $\mathbf{z} \in K^{N\times(n-d)}$.

To solve $\mathbf{Mw} = \mathbf{b}$, we need to be able to force:

- one solution with $(c_1, \ldots, c_d) = (1, 0, \ldots, 0)$,
- one solution with $(c_1, \ldots, c_d) = (0, 1, 0, \ldots, 0)$, etc.

Can we force $(c_1, \ldots, c_d)$?

Our proposed approach uses one single column of $\widehat{\mathbf{F}}$.

This won't do for $\mathbf{Mw} = \mathbf{b}$. Have only one $(c_1, \ldots, c_d)$ choice.

BUT we may combine the columns linearly.

- If $[X^0]\mathbf{F}_{\{1\ldots d\}\times\{1\ldots n\}}$ has rank d, then we can force any value for $(c_1, \ldots, c_d)$.
- More generally, the set of possible $(c_1, \ldots, c_d)$ is a vector space, and it can be covered.
- If we don't mind which $(c_1, \ldots, c_d)$ value we get, easy.

Cost: once for $\mathbf{F}$ is computed, cost for each vector in w is same as ours for one.

Conclusion

- Solving inhomogeneous linear systems with block Wiedemann has small overhead.
- Key is to put the right hand side in the starting vectors.
- For NFS-DL, SM columns do NOT have to go in the matrix.
- The same applies to block Lanczos (but less appealing anyway).

Implementation is more or less straightforward.

Must handle $\mathbf{F}_{\{1\ldots d\}\times\{1\ldots n\}}$ and $\mathbf{F}_{\{d+1\ldots n\}\times\{1\ldots n\}}$ properly.