Current Advances

in

Open Source

Gröbner Basis Algorithms

My name is

Christian Eder

I am from the

University of Kaiserslautern

3 years ago

Signature-based Gröbner Basis Algorithms in SINGULAR

Christian Eder, Jean-Charles Faugère and Bjarke Hammersholt Roune

May 15, 2014

Christian Eder, Jean-Charles Faugère

A survey on signature-based Gröbner basis computations

Journal of Symbolic Computation 80: 719-784 (May-June 2017 issue)

12 reviewers, +15 pages

Idea of signatures:

Try to detect zero reductions in advance

Let $\mathbf{I} = \langle g_1, g_2 \rangle \in \mathcal{R} := \mathbb{Q}[x, y, z]$ and let $<$ denote DRL where

$$g_1 = xy - z^2$$
$$g_2 = y^2 - z^2.$$

Apply signatures in $\mathcal{R}^2$:

$\mathbf{sig}(g_1) = e_1$,
$\mathbf{sig}(g_2) = e_2$.

Order signatures by POT (e.g. $e_2 > x^{1000} e_1$).

In general:
$\mathbf{sig}(\text{polynomial}) = \mathbf{lt}(\text{module representation})$

Main idea: Try to keep signature minimal.

Generate first S-pair:

$$\begin{aligned} g_3 &:= \mathrm{sp}(g_1, g_2) = y g_1 - x g_2 \\ &= y\left(xy - z^2\right) - x\left(y^2 - z^2\right) \\ &= xz^2 - yz^2. \end{aligned}$$

$$\mathrm{sig}(g_3) = \mathrm{lt}(ye_1 - xe_2) = -xe_2.$$

$\mathbf{sp}(g_3, g_2)$ reduces to zero:

$\mathbf{lt}(g_2) = y^2$ coprime to $\mathbf{lt}(g_3) = xz^2$

(Buchberger's Product Criterion)

$\mathsf{sp}(g_3, g_2)$ reduces to zero:

$$\begin{aligned} \mathsf{sig}(\mathsf{sp}(g_3, g_2)) &= \mathsf{lt}(y^2(ye_1 - xe_2) - xz^2e_2) \\ &= -xy^2e_2. \end{aligned}$$

Use syzygy $g_1e_2 - g_2e_1$ with lead term xye_2

▷ reduce module representation

▷ Lower signature for $\mathsf{sp}(g_3, g_2)$

(Syzygy Criterion)

What's about $\mathbf{sp}(g_3, g_1)$?

Buchberger's Product Criterion? **NO**

Buchberger's Chain Criterion? **NO**

But $\mathsf{sp}(g_3, g_1)$ reduces to zero:

$$\begin{aligned} \mathsf{sig}(\mathsf{sp}(g_3, g_1)) &= \mathsf{lt}(y(ye_1 - xe_2) - z^2 e_1) \\ &= -xye_2. \end{aligned}$$

Again: Syzygy $g_1 e_2 - g_2 e_1$ with lead term xye_2

▷ reduce module representation

▷ Lower signature for $\mathsf{sp}(g_3, g_1)$

(Syzygy Criterion)

Precondition for this talk:
Next chosen S-pair from pair set has minimal possible signature.

Note: Over fields this limitation is not required, but makes life easier.

General rule

For each signature handle exactly **one** element.

Sketch of proof

Take pairs by increasing signature.

α, β module elements.

If $\mathbf{sig}(\alpha) = \mathbf{sig}(\beta)$ reduce in module.
(We are not reducing the polynomials!)

- $\mathbf{sig}(\alpha - \beta) < \mathbf{sig}(\alpha), \mathbf{sig}(\beta)$.
- Algorithm has handled these relations already.

#1

Computing with signatures over the integers

joint work with

Gerhard Pfister

Adrian Popescu

Problem not mentioned until now: **sig-reductions**

When reducing polynomials (**over fields**) we are not allowed to change the signature:

Increasing signature?
Well, we want small signatures.

Decreasing signature?
We can throw away the element, criteria apply.

Over fields this ensures a computation by increasing signature.

Over the integers stuff gets more difficult.

We want **strong Gröbner Bases**:
For all $\mathbf{f} \in \mathbf{I} \setminus \{0\}$ there exists $\mathbf{g} \in \mathbf{G}$ s.t. $\mathbf{lt}(\mathbf{g}) \mid \mathbf{lt}(\mathbf{f})$.

($\mathbf{G} \subset \mathbf{I}$ and $\mathbf{L}(\mathbf{G}) = \mathbf{L}(\mathbf{I})$ is not enough anymore.)

Have to take care of the coefficients, too:

If

- $\mathsf{lm}(g) \mid \mathsf{lm}(f)$ &
- $\exists\, a, b \in \mathbb{Z}$ s.t. $\mathsf{lc}(f) = a\,\mathsf{lc}(g) + b$, $a \neq 0$ and $b < \mathsf{lc}(f)$

then compute $f - a\,\frac{\mathsf{lm}(f)}{\mathsf{lm}(g)}\,g$.

(Either **smaller lm** or **smaller lc**!)

Same process generalizes concept of S-pairs:

We need to consider **GCD-pairs**.

Problem arising from this:

Over Euclidean rings we can no longer guarantee that the computation of signature-based Gröbner Bases is done by increasing signatures.

We need to restrict signature-based criteria to remove useless elements:

We can only remove elements for a given signature S if there exists a syzygy π s.t.

$\mathrm{lt}(\pi) \mid S$

Still, signature drops can appear.

Idea

- ▷ Stop computation at this point.
- ▷ Interreduce intermediate basis without considering signatures.
- ▷ Apply new signatures / module representations and restart.

Optimization 1: Exploit GCD-pairs

Replace $\mathsf{f} \in \mathsf{G}$ by $\mathsf{gp}(\mathsf{f}, \mathsf{g})$ if there exists $\mathsf{g} \in \mathsf{G}$ s.t.

$$\begin{aligned} \mathsf{gp}(\mathsf{f}, \mathsf{g}) &= (\pm 1)\mathsf{f} + \mathsf{ctg} \ \ \& \\ \mathsf{sig}(\mathsf{gp}(\mathsf{f}, \mathsf{g})) &= (\pm 1)\mathsf{sig}(\mathsf{f})\,. \end{aligned}$$

▷ Trying to keep coefficient growth at a low.

Optimization 2: Optimistic **sig**-reductions

- **sig**-reduce an element $\mathbf{f}$ w.r.t. $\mathbf{G}$.
- If there exists $\mathbf{g} \in \mathbf{G}$ s.t. $\mathbf{ct}\,\mathbf{lt}(\mathbf{g}) = \mathbf{lt}(\mathbf{f})$ and $\mathbf{sig}(\mathbf{f} - \mathbf{ctg}) < \mathbf{sig}(\mathbf{f})$ for some $\mathbf{c}$ and $\mathbf{t}$ then start **usual reduction process** (no longer taking care of signatures)
- If $\mathbf{f}$ reduces to zero we can go on.
- Otherwise we have to restart the computation.

Restarting is a huge bottleneck in general.

But often the intermediate computed elements are quite useful for further computations.

Optimization 3: Hybrid algorithm

- Start with signature-based algorithm
- If signature drops, restart for a (small) number of times the signature-based algorithm.
- Take intermediate basis and start non-signature-based Gröbner basis computation.

Examples	STD	HBA	STD/HBA
1	10.43	0.37	28.19
2	24.91	0.10	249.10
3	87.27	0.39	223.77
4	83.51	0.20	417.55
5	23,200.05	5,873.21	3.95
6	134.29	0.61	220.15
7	1,004.56	1,128.07	0.89
8	554.02	337.55	1.641

#2

Some other ideas when computing over the integers

joint work with

Gerhard Pfister

Adrian Popescu

wonderful bug support by

Anne Frühbis-Krüger

Jakob Kröker

Syzygy check

- $I = \langle f_1, \ldots, f_r \rangle$, $J = \langle f_1, \ldots, f_r, 1 \rangle$.
- Compute GB H for I over $\mathbb{Q}$.
- Compute syzygies S for J over $\mathbb{Q}$.
- If $H = \langle 1 \rangle$ extract c from S, $I = I \cup \{c\}$.
  Else try to extract cm from S, $I = I \cup \{cm\}$.
- Compute GB G for I over $\mathbb{Z}$.

Normal form for local/mixed orders

Mora: Termination relies on
elements of large ecart

$$\mathbf{ecart}(f) = \mathbf{deg}(f) - \mathbf{deg}(\mathbf{lt}(f))$$

Over the integers we need to apply
this principle to GCD-computations, too.

Reduce f w.r.t. intermediate basis G.

Initialize set of possible reducers $\mathsf{R} = \mathsf{G}$.

While $\mathsf{f} \neq 0$ and $\exists\, \mathsf{g} \in \mathsf{R}$ s.t. $\mathbf{lt}(\mathsf{g}) \mid \mathbf{lt}(\mathsf{f})$:

- ▷ Choose $\mathsf{g} \in \mathsf{R}$, $\mathbf{lt}(\mathsf{g}) \mid \mathbf{lt}(\mathsf{f})$ (minimal ecart).
- ▷ If $\mathbf{ecart}(\mathsf{f}) < \mathbf{ecart}(\mathsf{g})$ then
  $\mathsf{R} = \mathsf{R} \cup \{\mathsf{f}\} \cup \{\mathbf{gp}(\mathsf{f}, \mathsf{h}) \mid \mathsf{h} \in \mathsf{R}\}$.
- ▷ Reduce f with g.

#3
Non-commutative stuff

joint work with

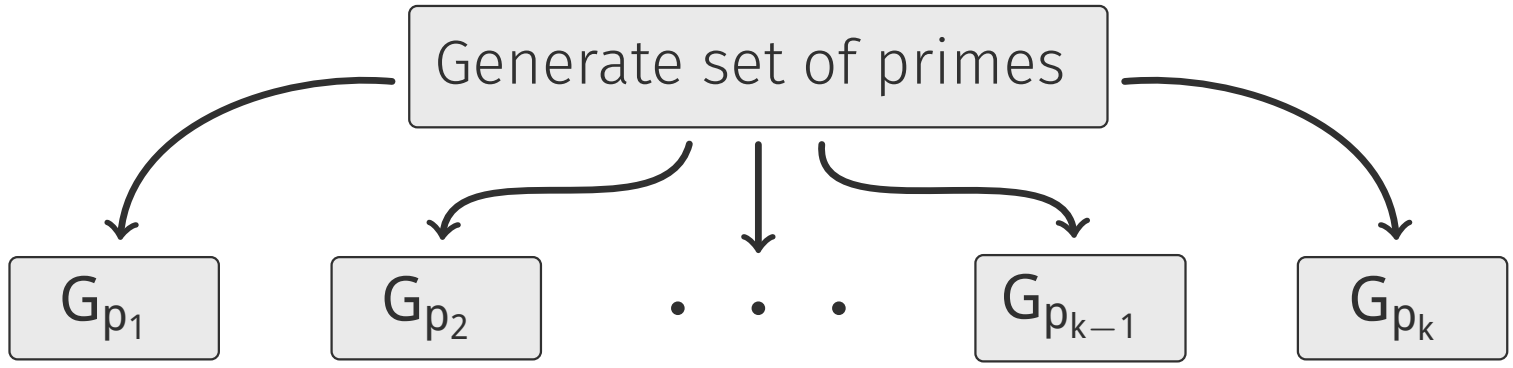
Wolfram Decker
Viktor Levandovskyy
Sharwan K. Tiwari

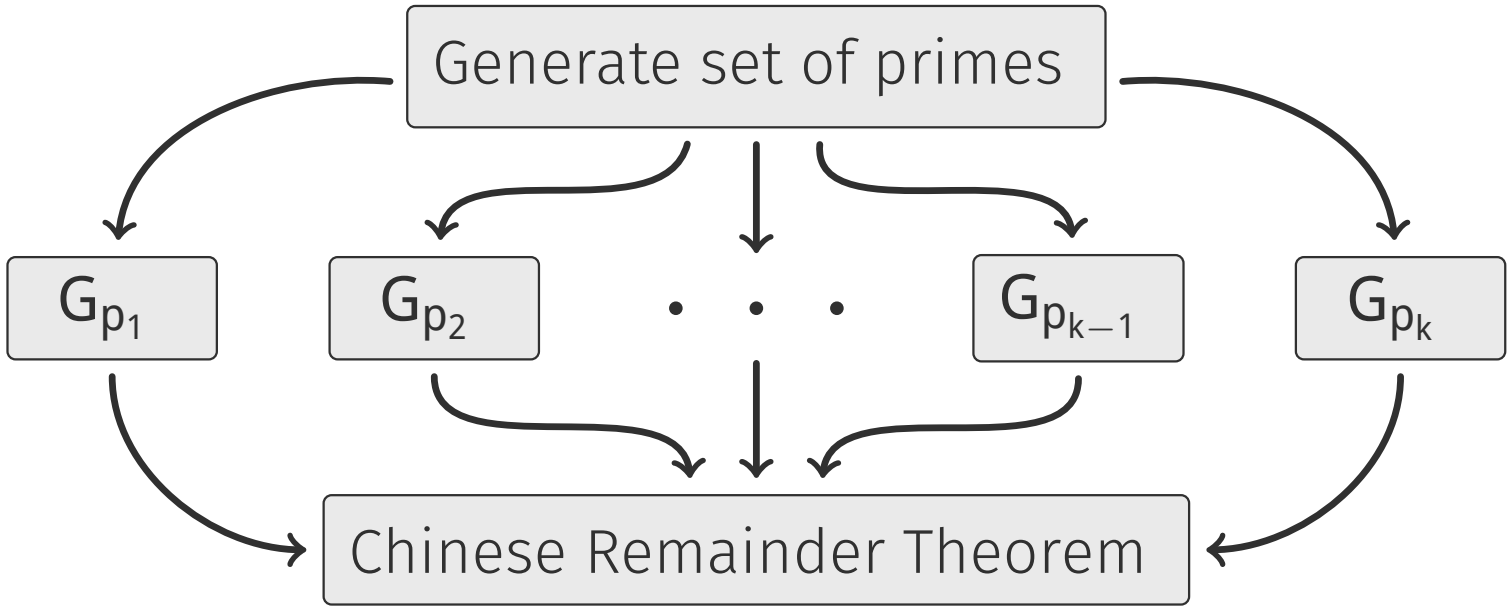
Modular GB Computation

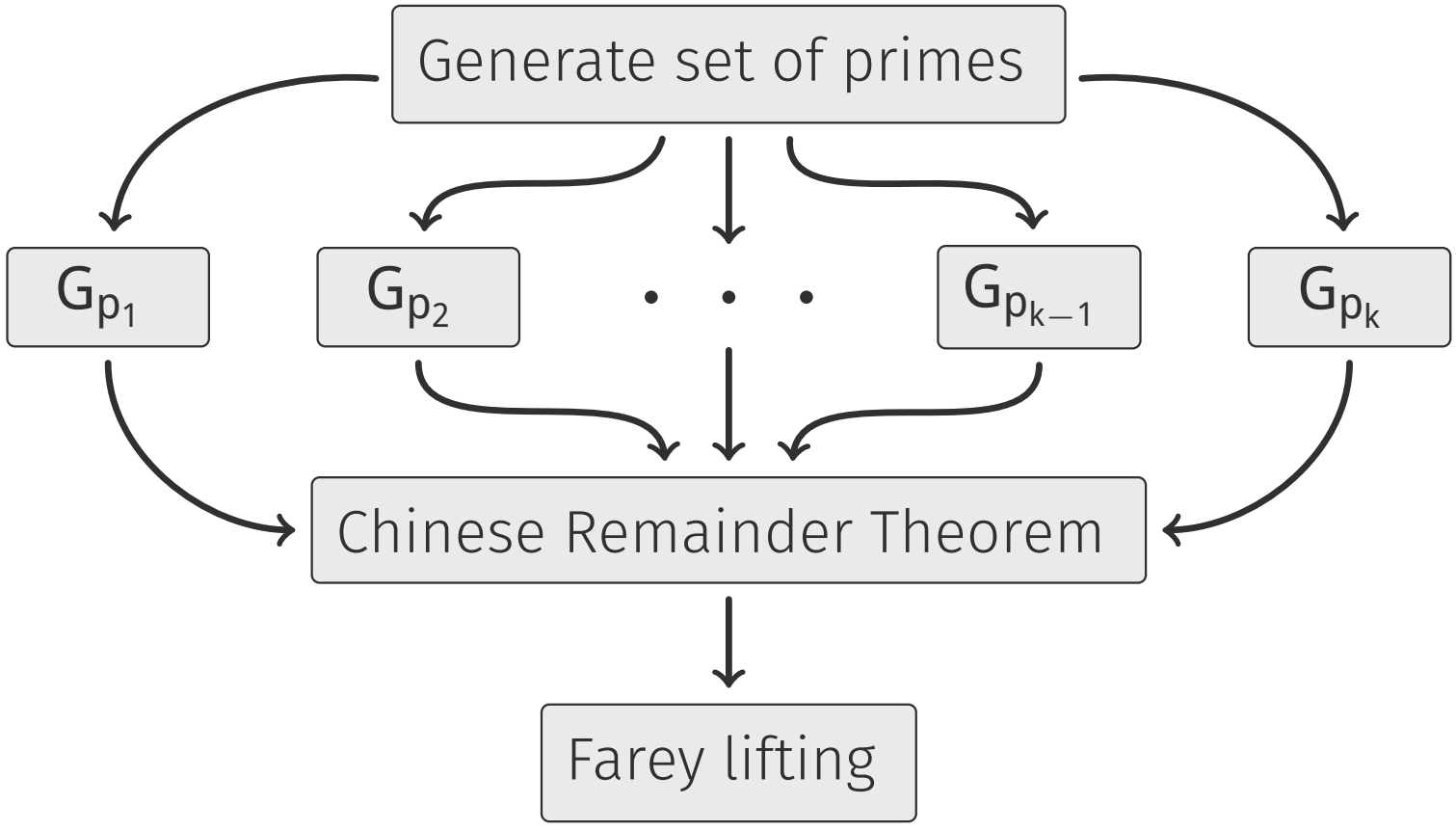
We extended modular techniques for computing Gröbner bases to the class of non-commutative **G**-algebras.

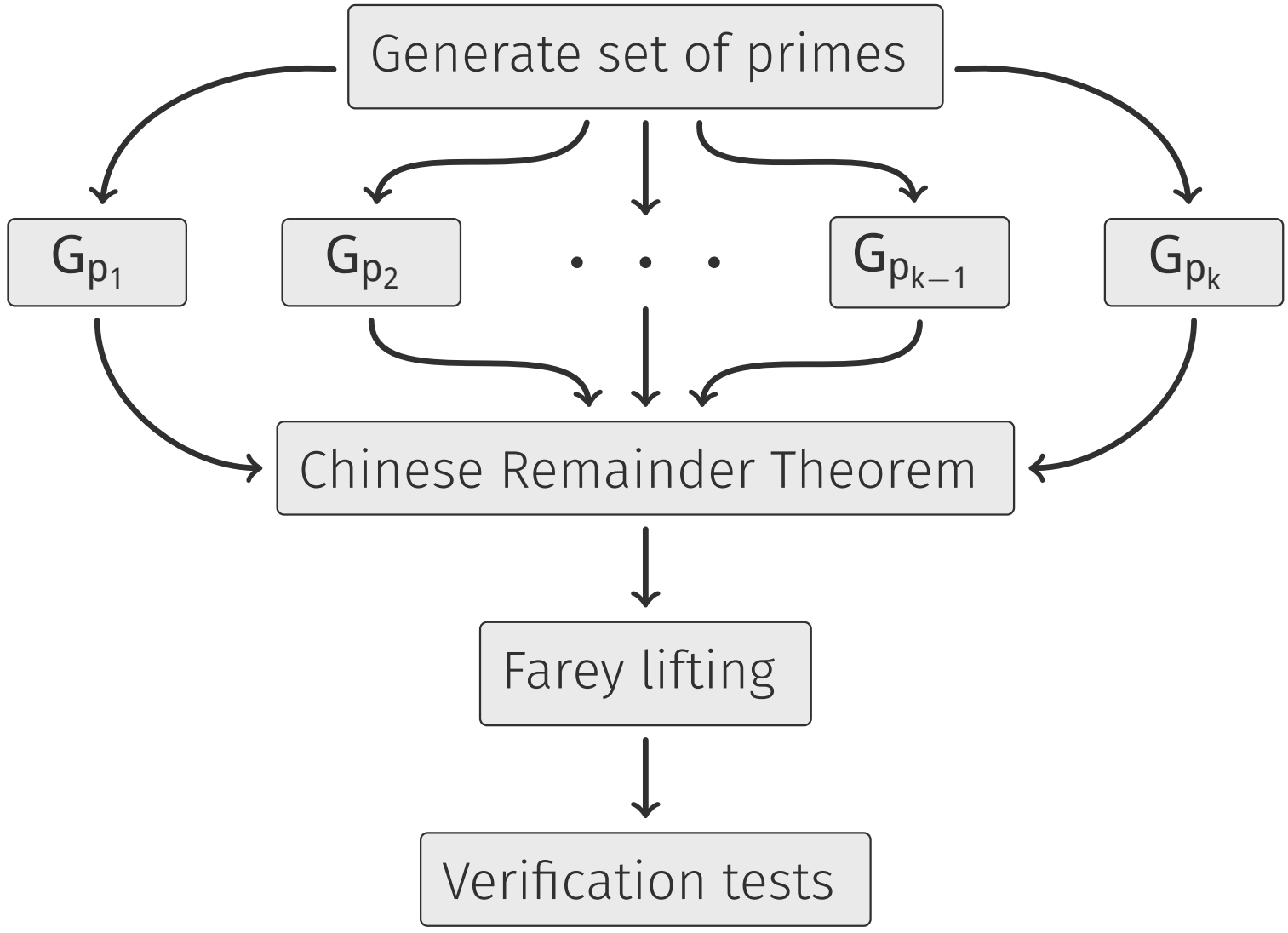
This allows **parallel** computation.

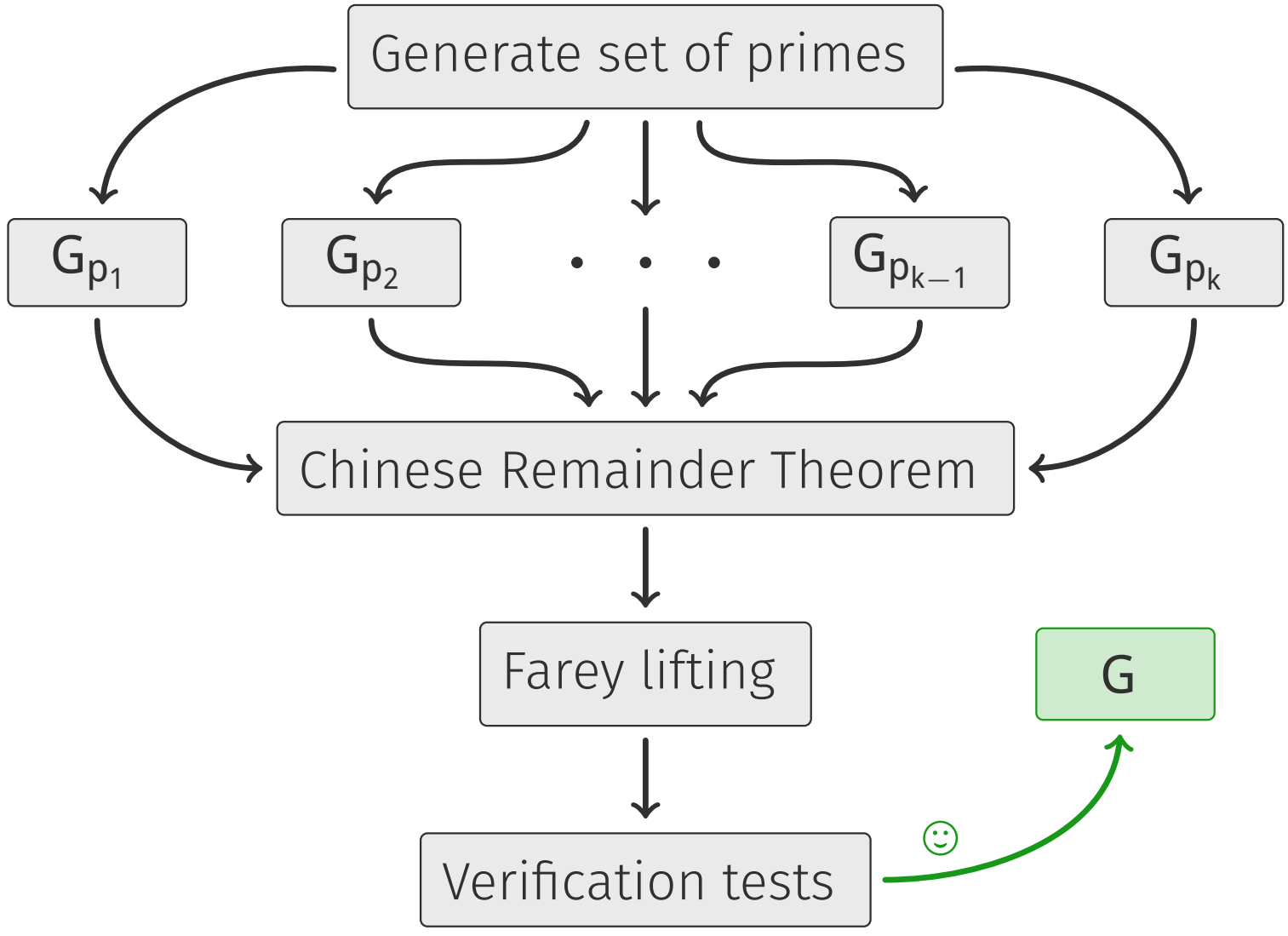

Generate set of primes

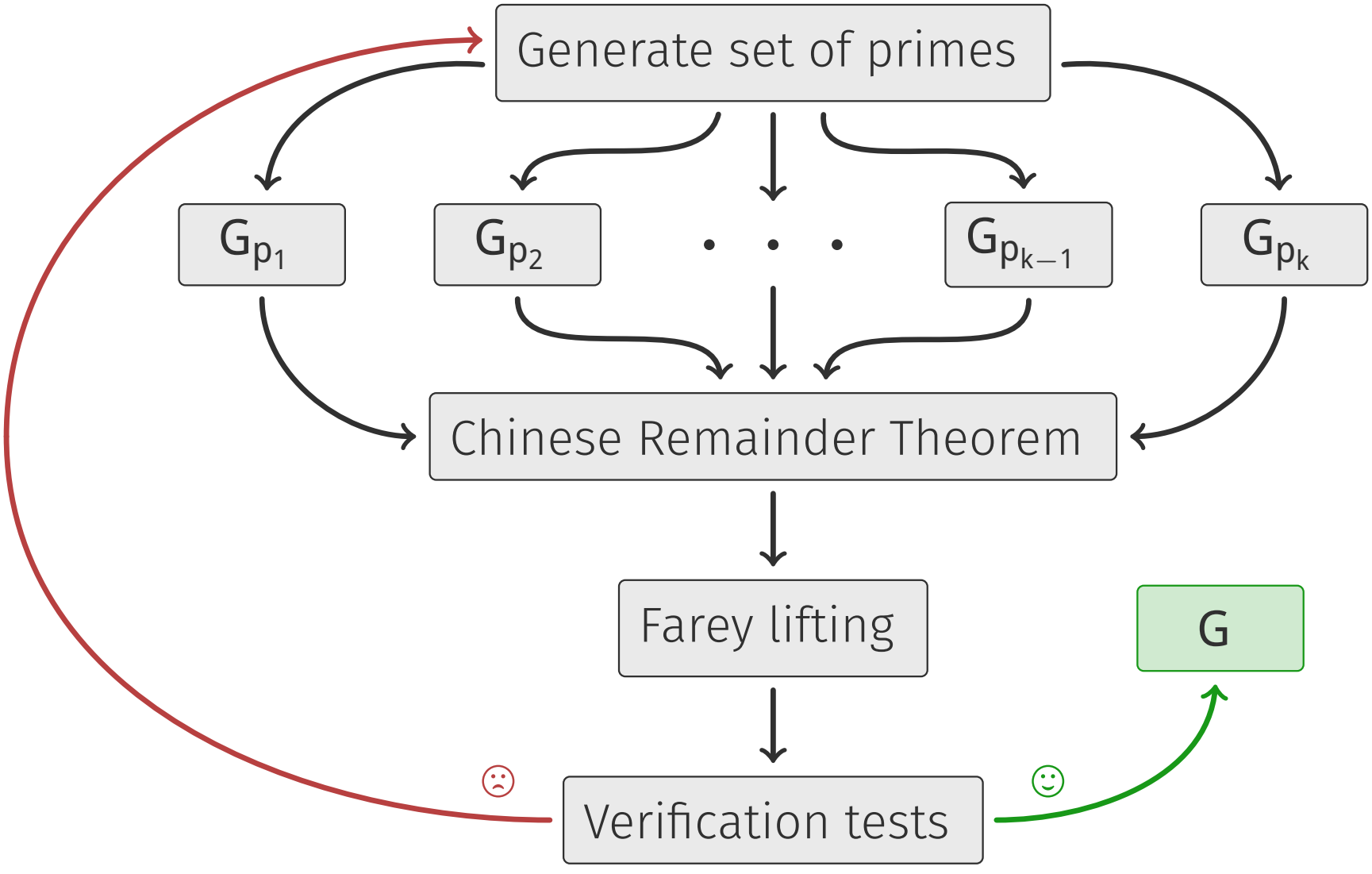

Generate set of primes
G_{p_1}
G_{p_2}
$G_{p_{k-1}}$
G_{p_k}


Generate set of primes
G_{p_1}
G_{p_2}
$G_{p_{k-1}}$
G_{p_k}
Chinese Remainder Theorem


Generate set of primes
G_{p_1}
G_{p_2}
$G_{p_{k-1}}$
G_{p_k}
Chinese Remainder Theorem
Farey lifting


Generate set of primes
G_{p_1}
G_{p_2}
$G_{p_{k-1}}$
G_{p_k}
Chinese Remainder Theorem
Farey lifting
Verification tests


Generate set of primes
G_{p_1}
G_{p_2}
$G_{p_{k-1}}$
G_{p_k}
Chinese Remainder Theorem
Farey lifting
G
Verification tests


Generate set of primes
G_{p_1}
G_{p_2}
$G_{p_{k-1}}$
G_{p_k}
Chinese Remainder Theorem
Farey lifting
G
Verification tests

Verification tests?

As in the commutative case: Effective verification tests are only known in the **graded** case.

(Still, all non-graded examples we computed were correct.)

Example set 1

Quasi-commutative graded $\mathbb{Q}$-algebra
$A = \mathbb{Q}\langle x_1, \ldots, x_n \mid x_j x_i = 2x_i x_j,\ 1 \leq i < j \leq n\rangle$,

degree reverse lexicographic order
(i.e. weight vector $\omega = (1, \ldots, 1)$).

In the corresponding Rees algebra, we compute left Gröbner bases for homogenized versions of the following benchmark systems.

Examples	sgb	m-sgb-1	m-sgb-2	m-sgb-4	m-sgb-8	m-sgb-16
cyclic(7)	11.24 m	27.66 s	16.13 s	9.66 s	7.81 s	6.64 s
cyclic(8)	55.28 h	2.51 h	1.21 h	34.65 m	27.64 m	17.13 m
katsura(9)	4.49 m	1.51 m	49.27 s	30.60 s	21.77 s	16.28 s
katsura(10)	10.65 h	26.83 m	14.54 m	8.59 m	3.53 m	3.38 m
katsura(11)	199.71 h	4.32 h	2.76 h	1.59 h	46.48 m	24.52 m
katsura(12)	-	13.78 h	7.68 h	4.40 h	2.34 h	1.46 h
katsura(13)	-	50.14 h	32.33 h	17.74 h	10.72 h	5.80 h
reimer(4)	14.62 s	3.14 s	2.69 s	1.99 s	1.58 s	1.48 s
reimer(5)	29.07 h	2.59 h	1.57 h	58.47 m	26.33 m	18.04 m
eco(15)	25.93 h	9.40 h	5.77 h	3.54 h	2.55 h	1.83 h

Example set 2 Reiffen curves

Consider the family of polynomials
$x^p + y^q + xy^{q-1} \in \mathbb{Q}[x, y]$ s.t. $q \geq p + 1$.

Correspondingly we consider the second Weyl algebra

$$D_2(\mathbb{Q}) = \mathbb{Q}\langle x, y, \partial_x, \partial_y \mid \partial_x x = x\,\partial_x + 1, \partial_x y = y\,\partial_x, \\ \partial_y y = y\,\partial_y + 1, \partial_y x = x\,\partial_y \ \rangle,$$

Want to compute the Bernstein-Sato polynomial.

(Time for computing the annihilator is negligible,
we give only the time for the elimination step.)

Examples	sgb	m-sgb-1	m-sgb-2	m-sgb-4	m-sgb-8	m-sgb-16
Reiffen(5,6)	63.86 h	12.25 m	7.21 m	4.70 m	3.45 m	2.60 m
Reiffen(6,7)	-	10.43 h	6.03 h	4.65 h	4.24 h	3.54 h
Reiffen(7,8)	-	336.25 h	212.24 h	170.32 h	146.45 h	118.17 h

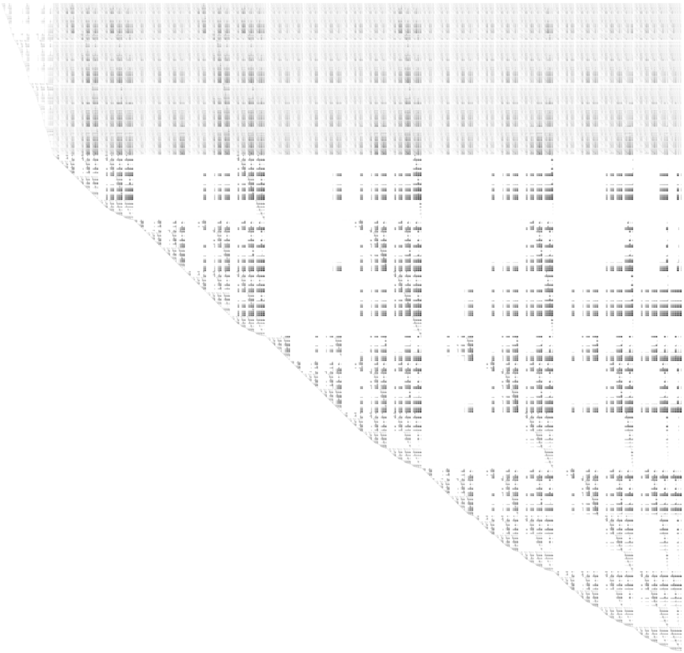
Note: The verification tests take 17% / 27% / 30% of the single core running time.

#4

Getting the linear algebra done

joint work with

Jean-Charles Faugère



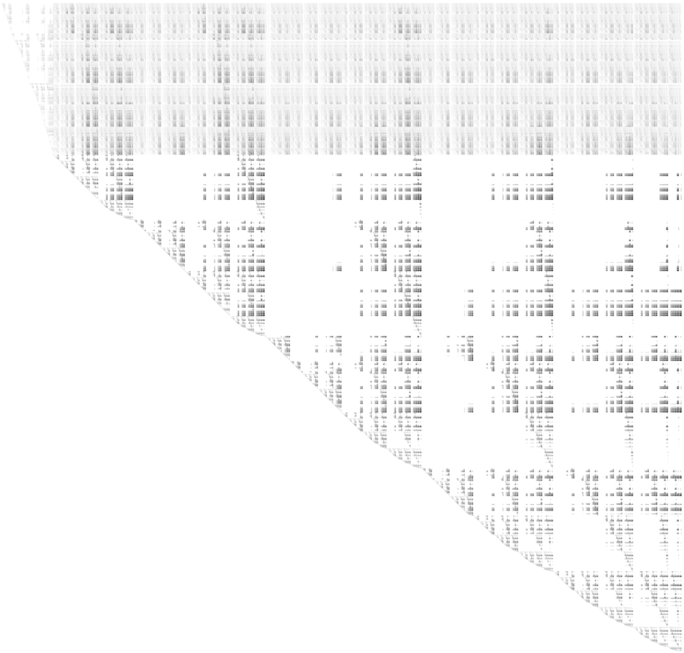
Problem

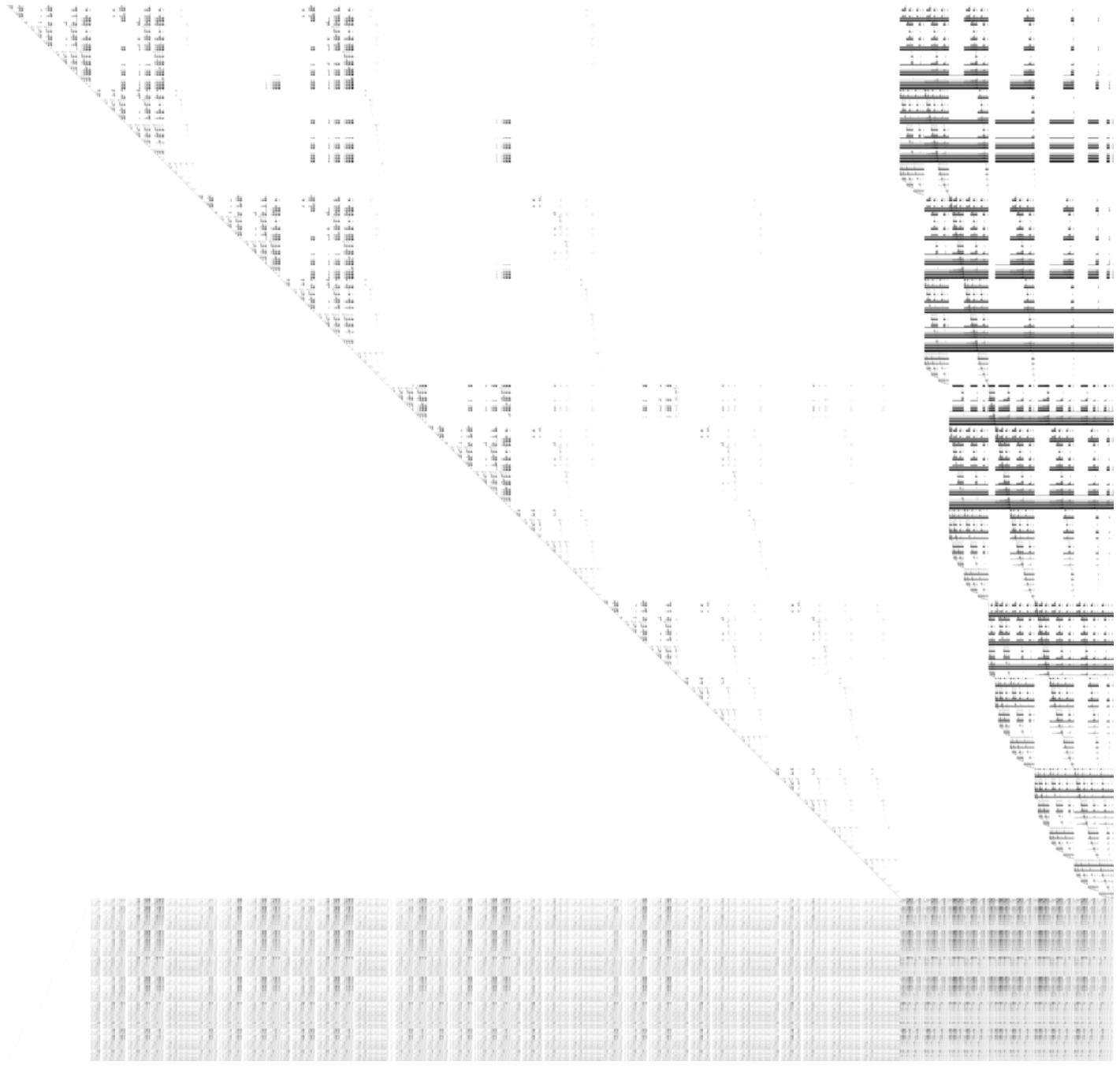
When applying Gaussian Elimination
we cannot swap columns:

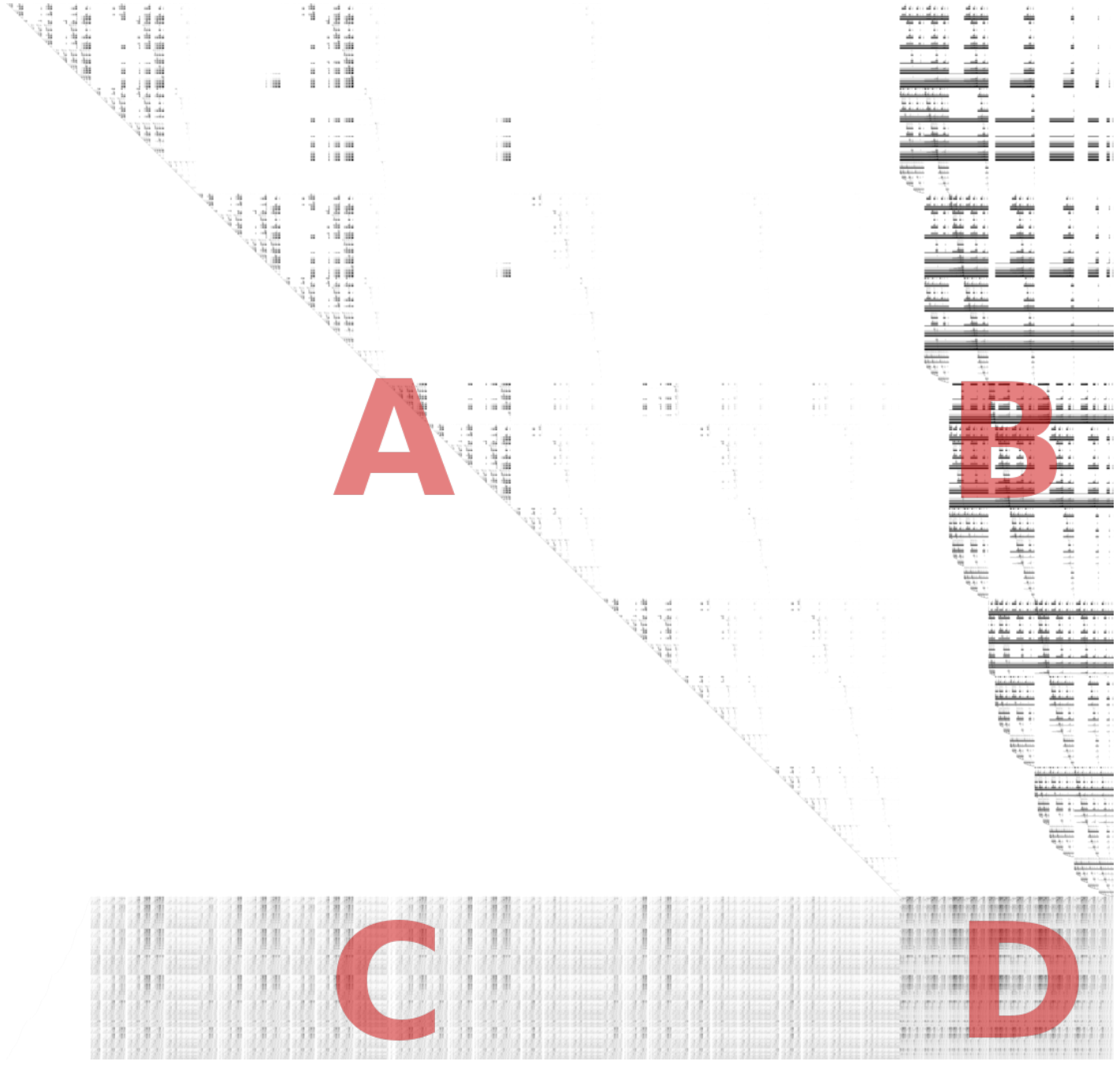
column order $=$ **monomial order**

Idea

- ▷ Order columns in a “nice” way.
- ▷ Apply specialized Gaussian Elimination.
- ▷ Reorder columns.



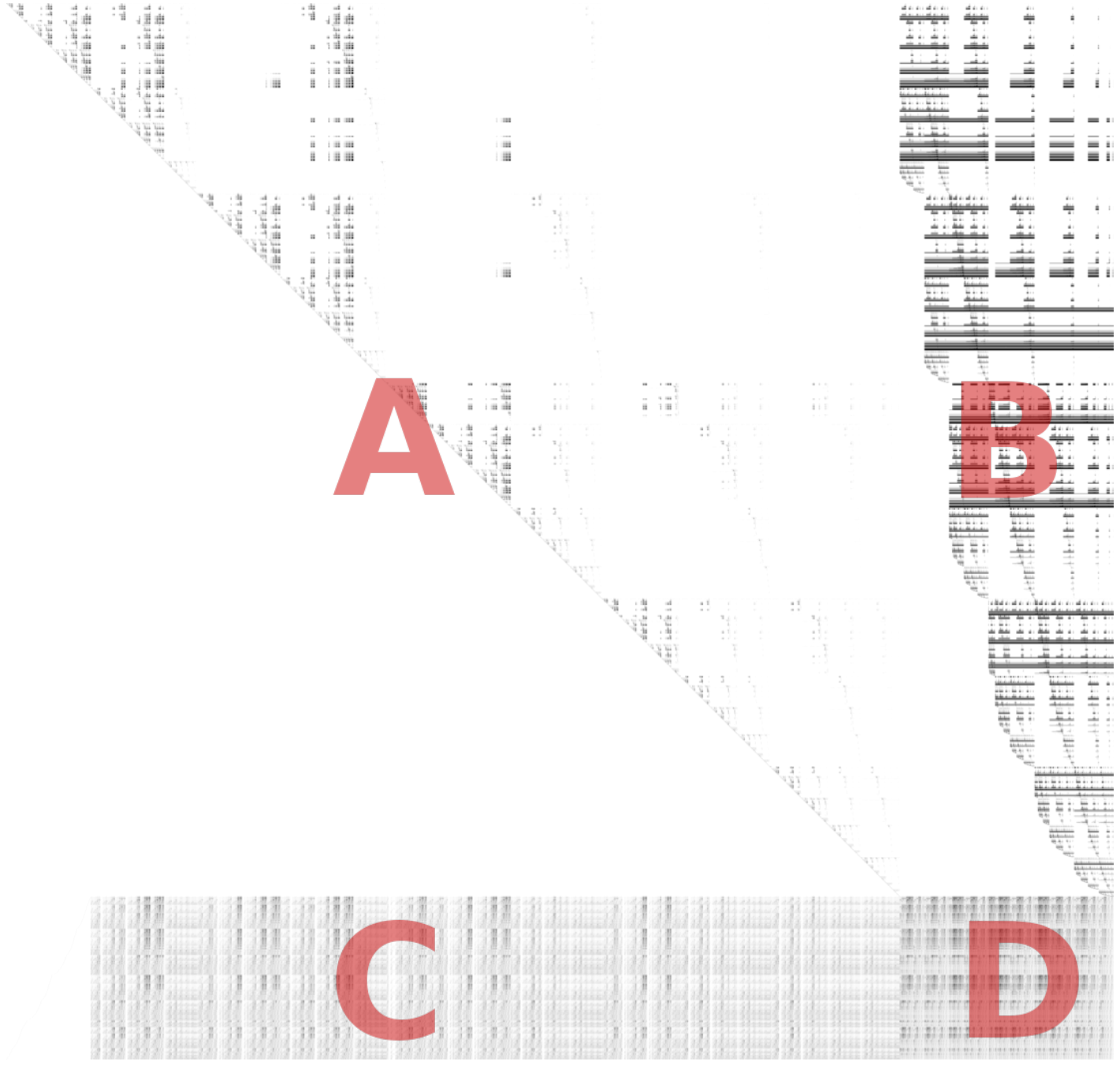


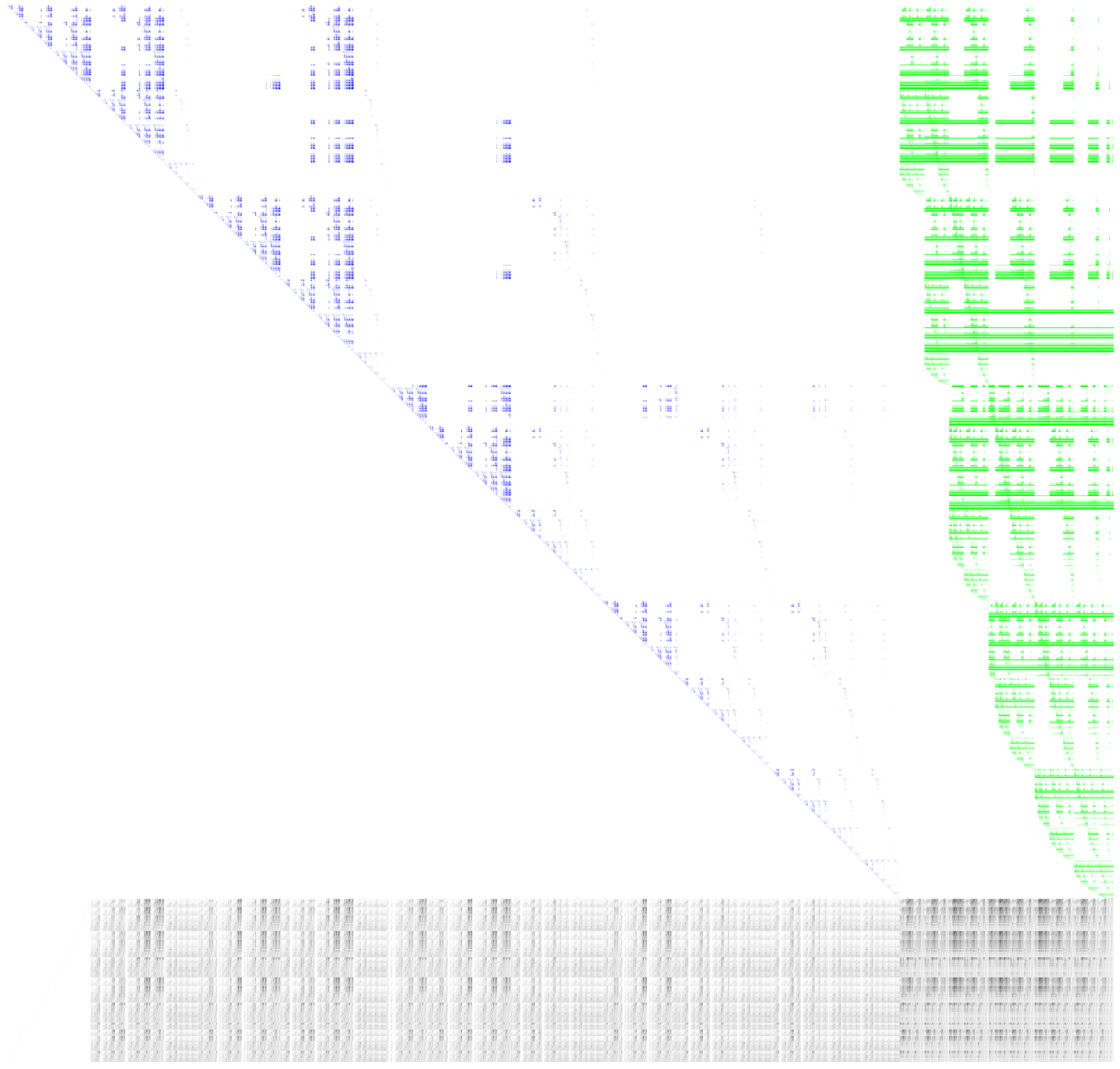

A
B
C
D

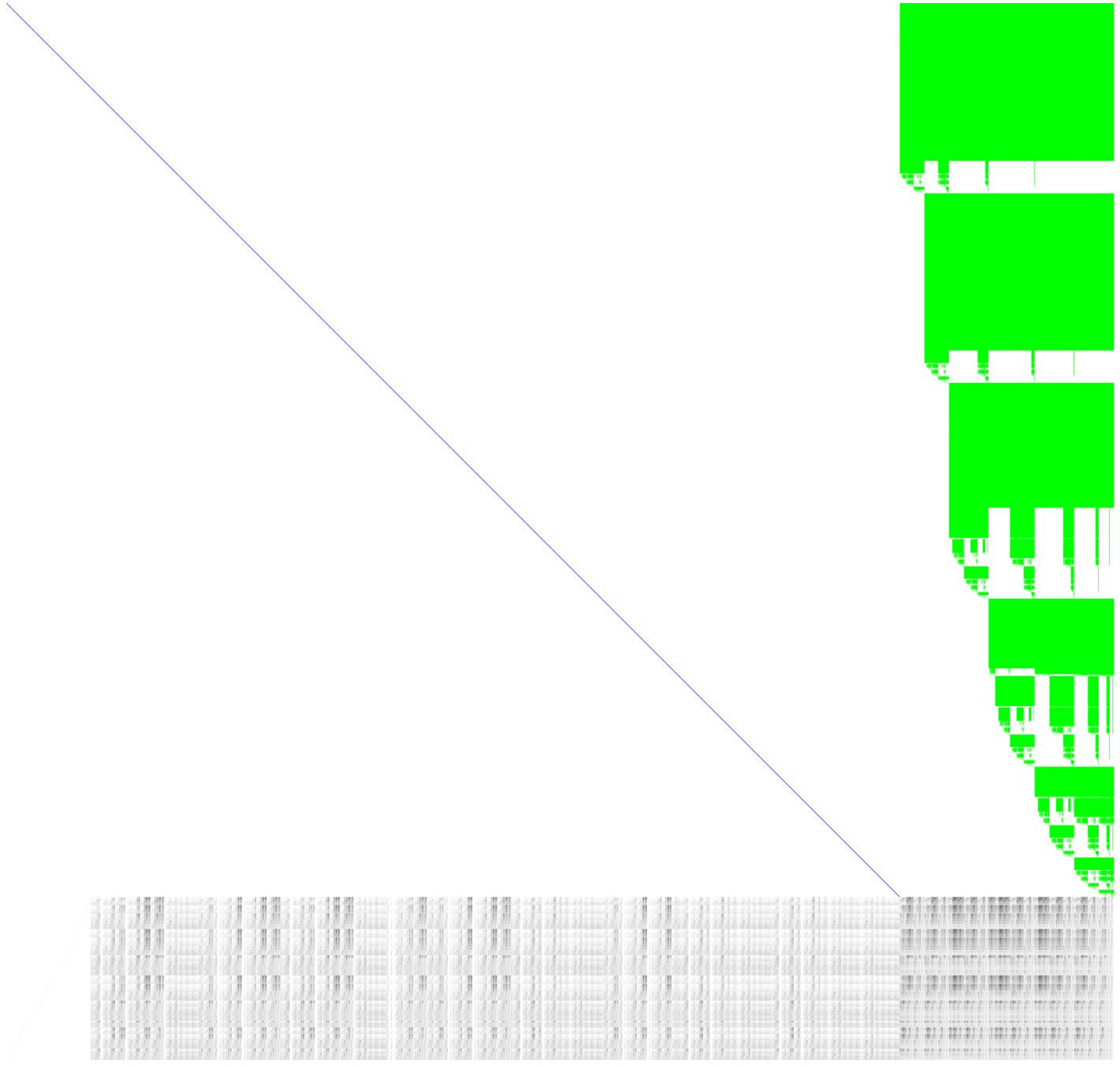
Different modes of operation

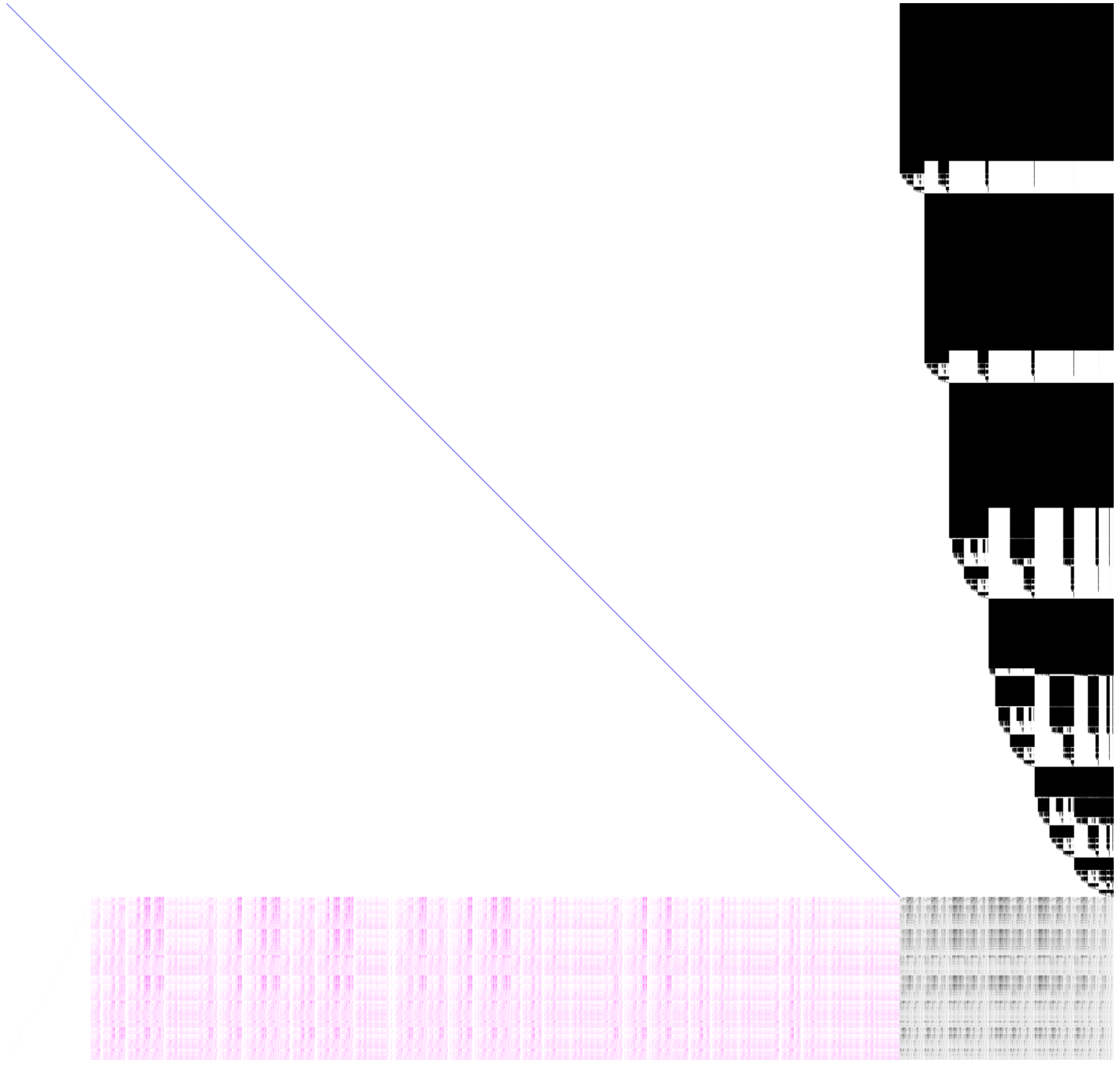
Mode 1

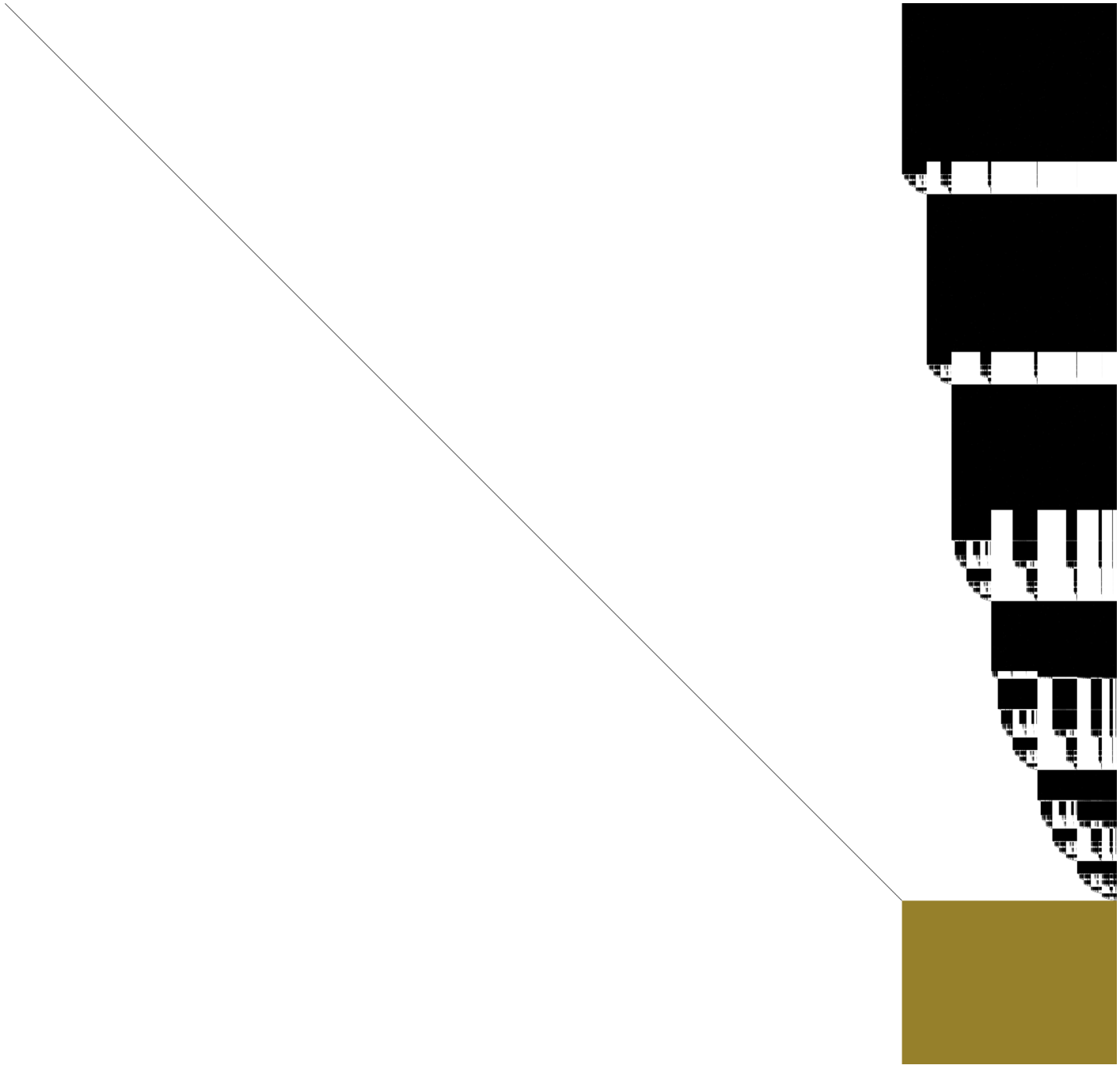
- Compute $\mathbf{A}^{-1}\mathbf{B}$.
- Reduce $\mathbf{C}$ to zero.
- Eliminate $\mathbf{D}$.

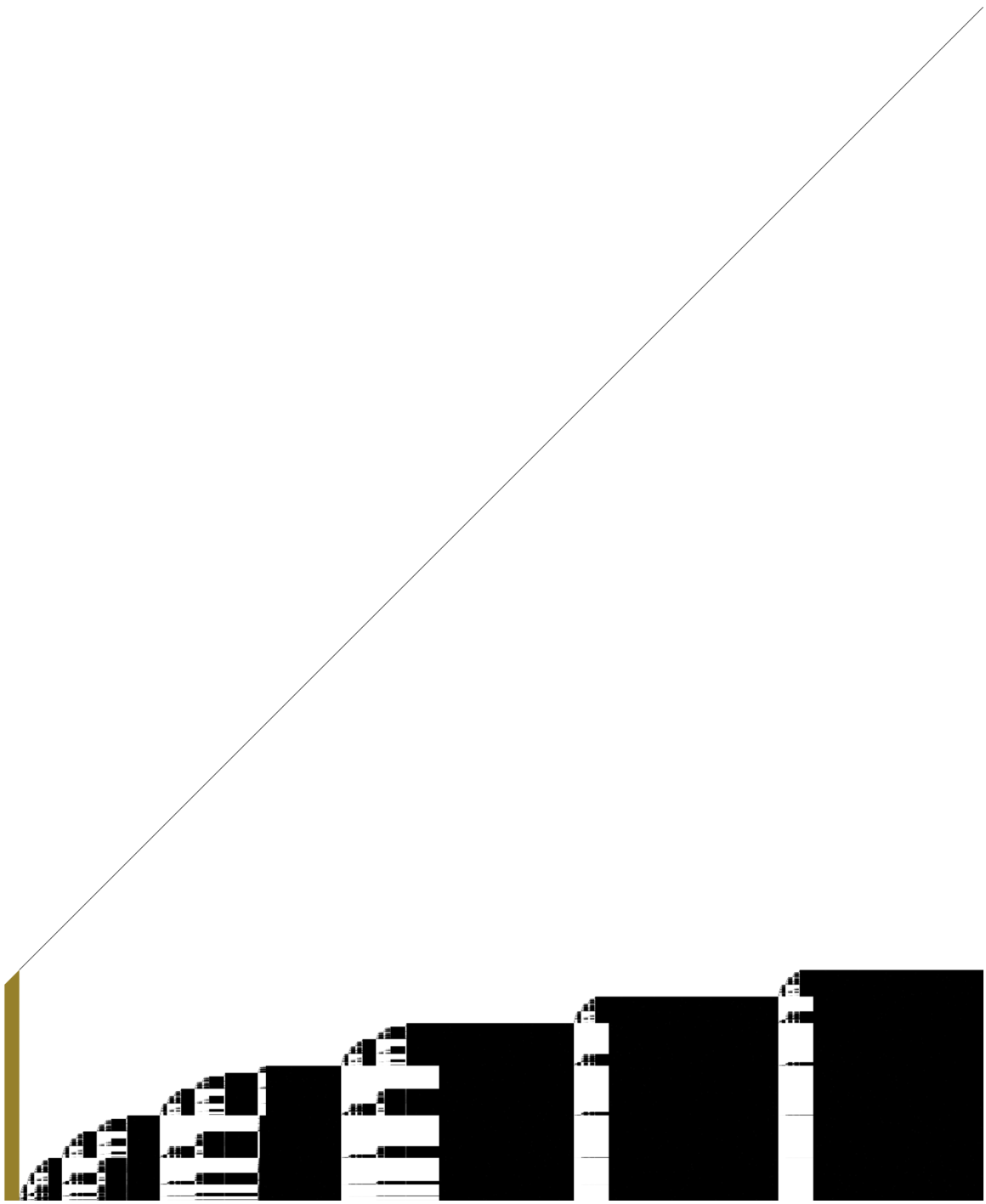

A
B
C
D





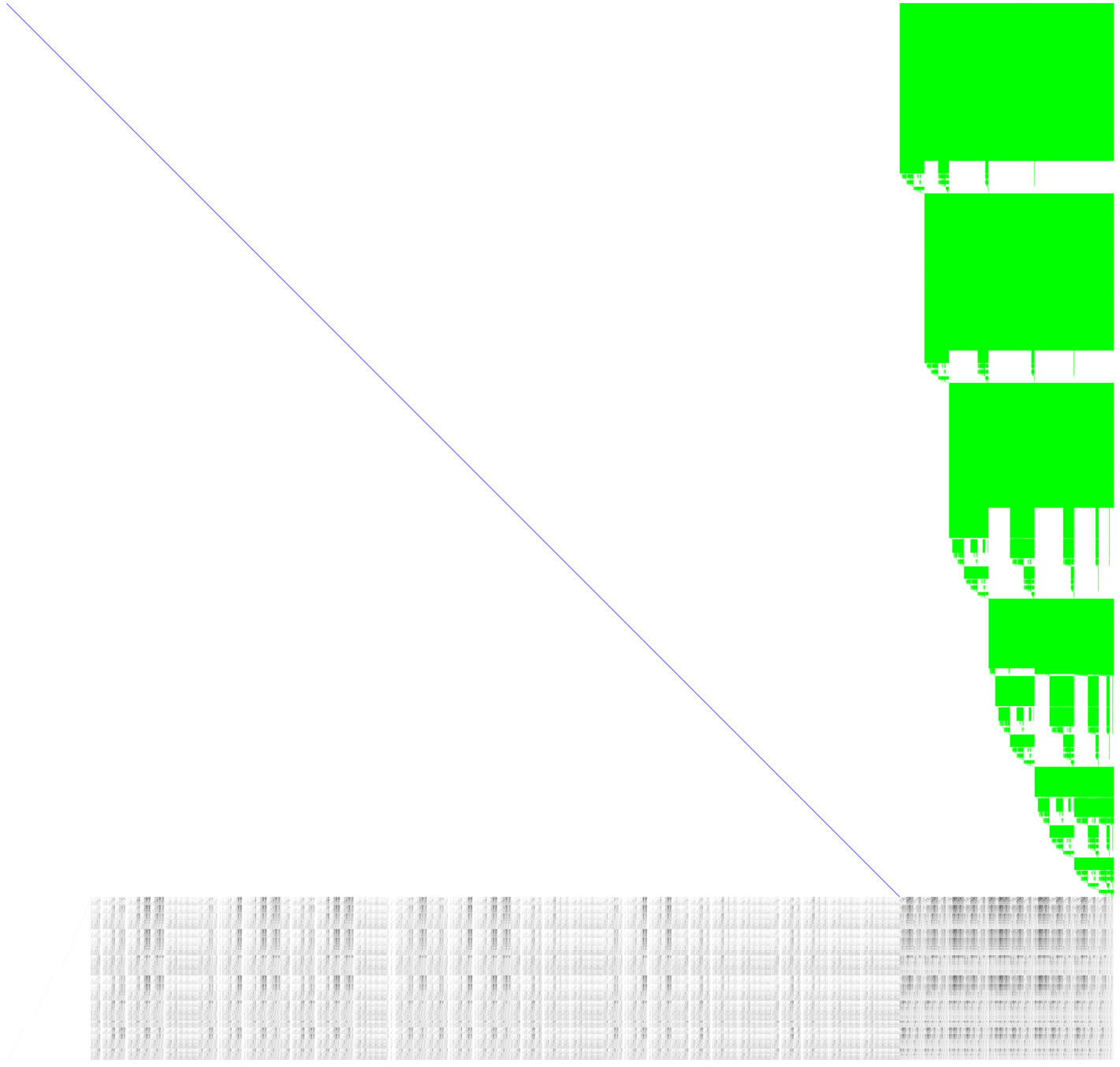


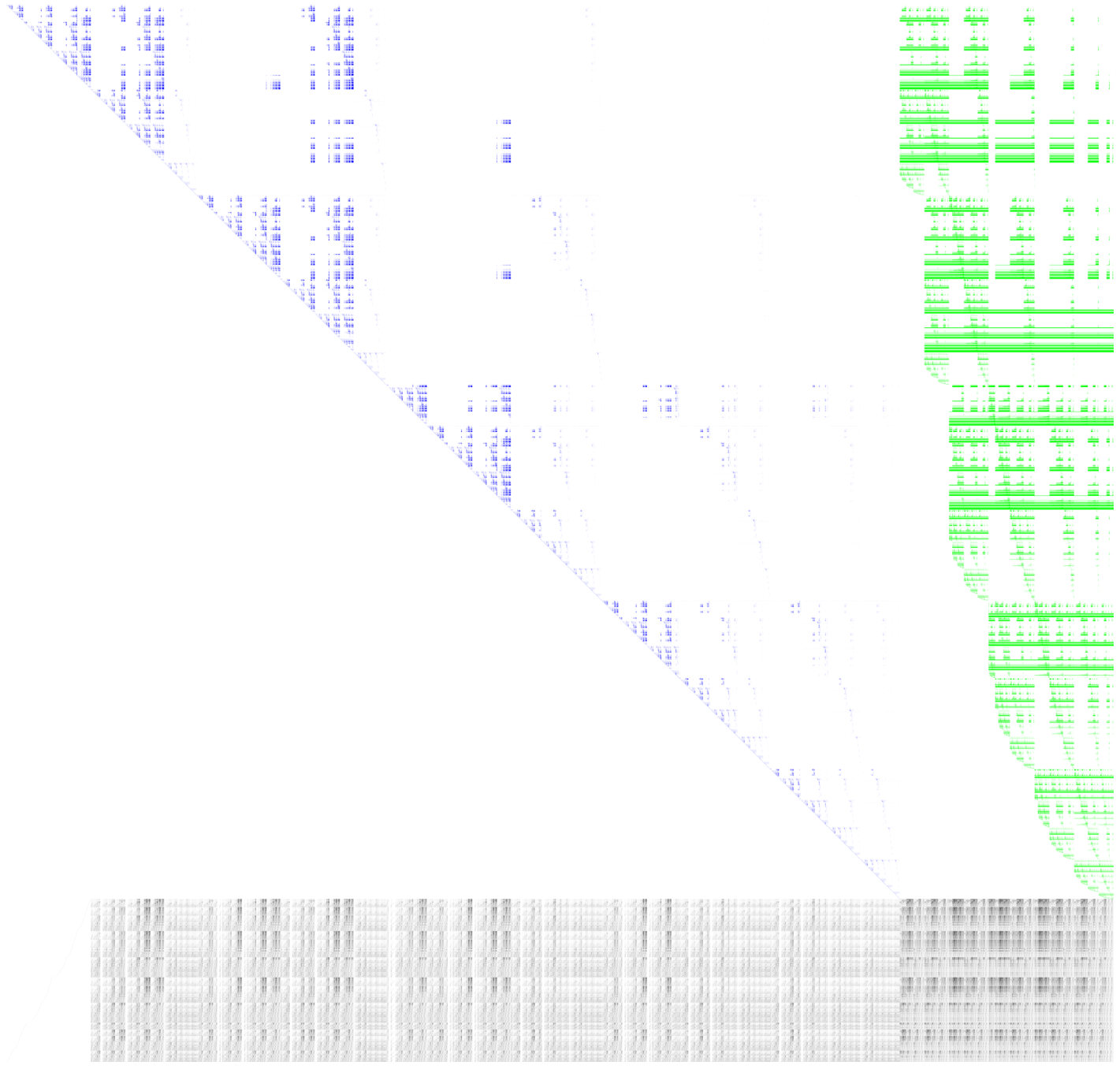




Mode 2

- Compute $A^{-1}B$.
- Reduce $\mathbf{C}$ to zero.
- Eliminate $\mathbf{D}$.



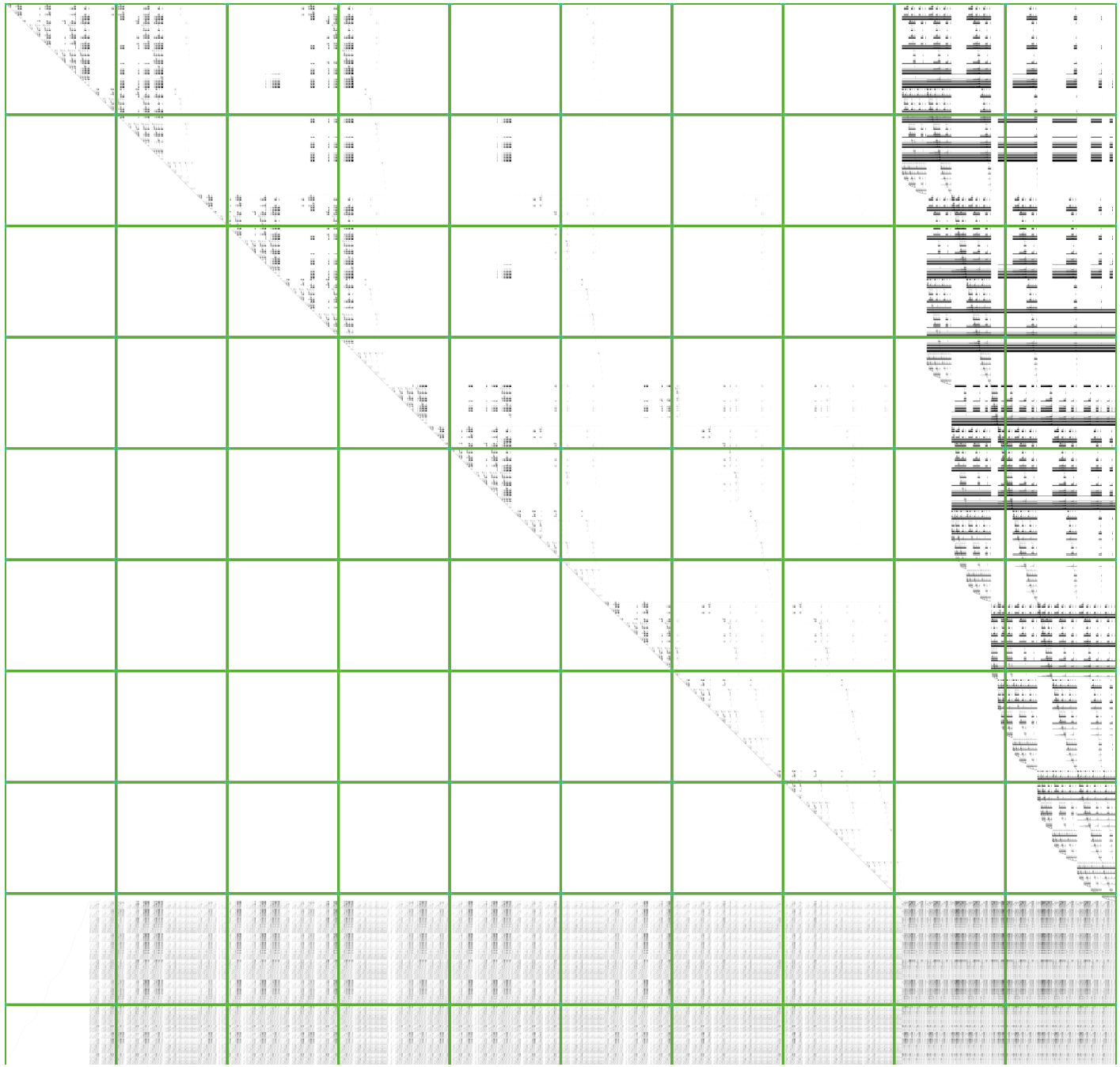


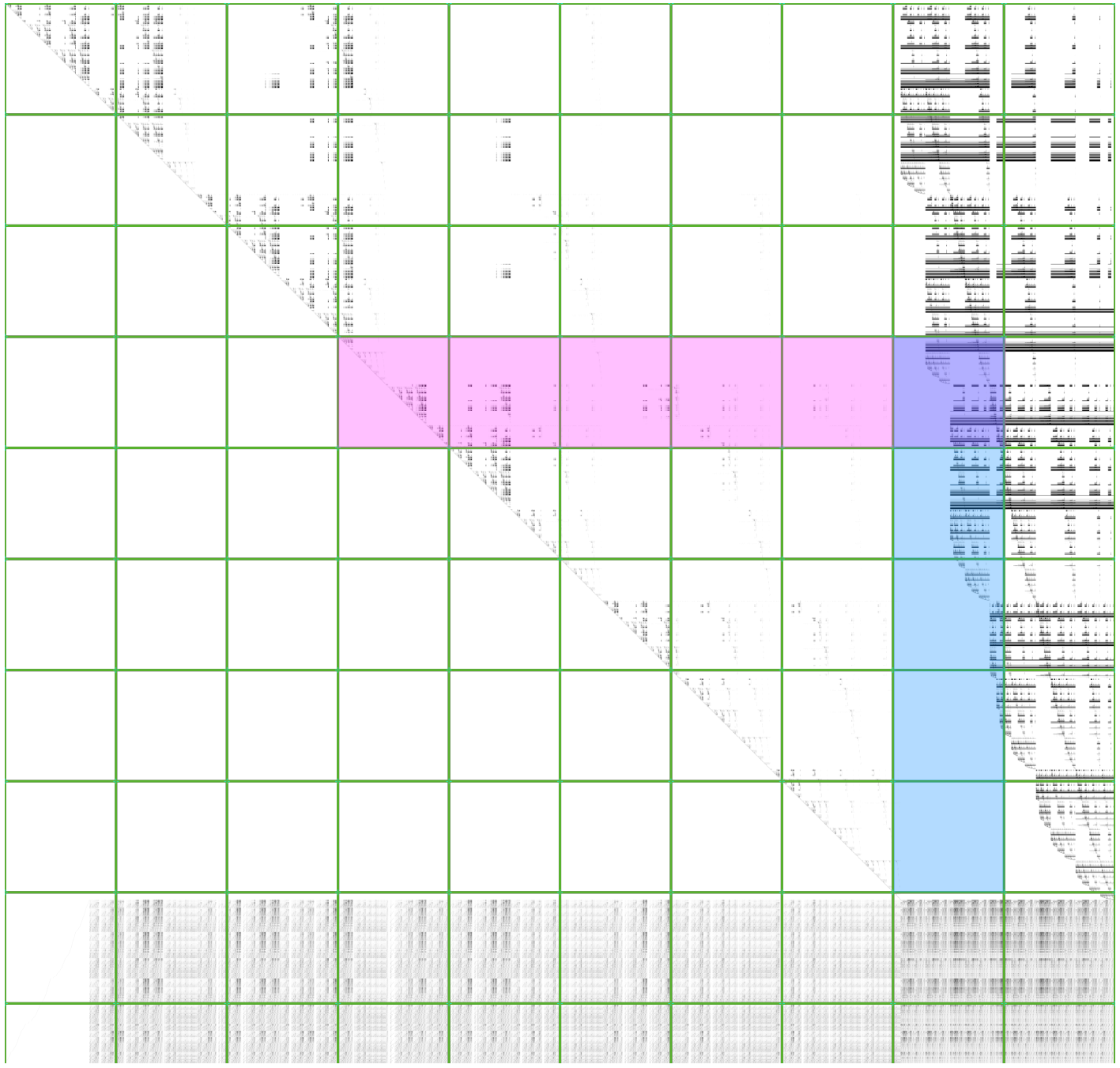
Mode 3

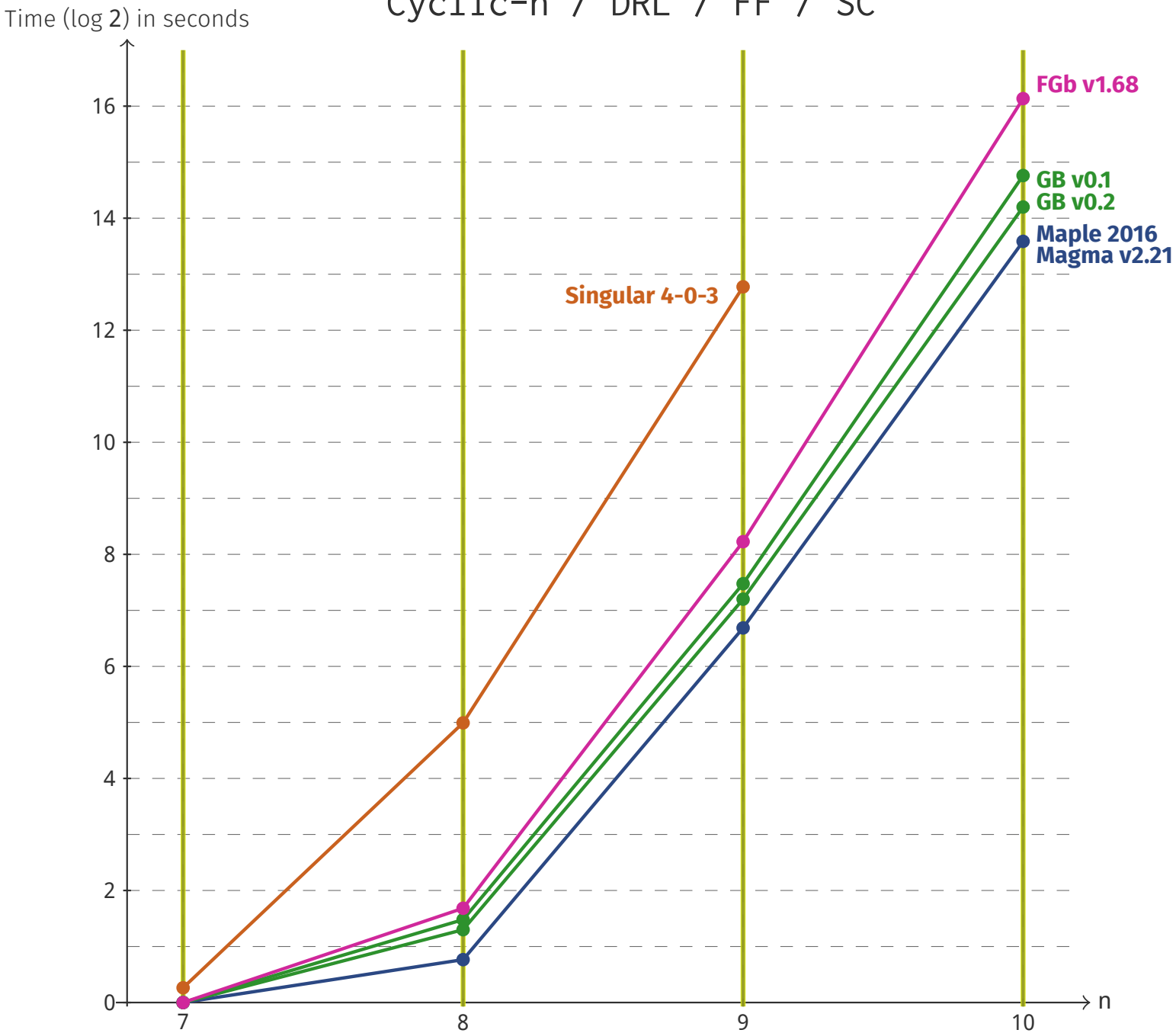
- **Partly** reduce **A**.
- Reduce **C** to zero.
- Eliminate **D**.

Parallelization

on **CPU**s via **OpenMP**






Cyclic-n / DRL / FF / SC
Time (log 2) in seconds
16
14
12
10
8
6
4
2
0
7
8
9
10
n
FGb v1.68
GB v0.1
GB v0.2
Maple 2016
Magma v2.21
Singular 4-0-3

New features in GB v0.2:

Matrix generation on the fly

Better data locality,
up to **70**% less memory usage

Adaptively chosen block sizes

Prime fields up to 2^{23} bits

Better parallel (OpenMP) scaling
(esp. for very sparse matrices)

Next steps for GB:

Better hashing, k-d-trees

ARM chips (gimme ya smartphone!)

GPU usage for linear algebra (OpenCL)

On-chip GPU usage for hashing (OpenCL)

Distributed computation

Multi-modular computation

Signature-based linear algebra

Other wishes?

Please remember

computer **ALGEBRA**

COMPUTERALGEBRA ✓

Available in **OSCAR** soonish.

aka

Once lecture time is over.

Speaking of

See you at

PASCO'17 23–24/07/2017
ISSAC'17 25–28/07/2017
SC2'17 29/07/2017

in Kaiserslautern.

Thank you for your attention.

Questions? Remarks?