Parallelizing the F4 Algorithm

Roman Pearce

July 2026

On the internet at `http://axcas.net` under "code"

- you can download the exact axf4ver2g.zip used here
- there is also a package with FGLM for developers
- both are public domain

Historical Benchmark

1994 Pentium 90 Mhz, 16MB RAM, 32-bit DOS extender

2004 iBook G4 1.07 GHz, 768MB RAM, 32-bit Mac OS X 10.5

2014 Pentium G3250 3.2 GHz, 16GB RAM, 64-bit Windows 10

2024 Core i5 1340p 4.6 GHz, 64GB RAM, 64-bit Debian 13

axf4 Gröbner basis single threaded

$p = 2^{31} - 1$	1994	2004	2014	2024
cyclic8	possible	39.282s	2.968s	1.085s
cyclic9	–	2872.067s	124.491s	56.797s
cyclic10	–	–	10678.782s	5217.471s
cyclic11	–	–	–	–

- single threaded performance is really hitting a wall
- RAM increased by a factor of 48, then 21.3, then 4
- Gröbner bases can take exponential space

High Performance Computing Tools

Vectorization:

- CPU threads operate on multiple data simultaneously
- e.g. multiply eight 32-bit integers at a time

Parallelization:

- multiple threads co-ordinate through shared memory
- e.g. 16 core/32-thread Ryzen CPU

Extra Memory:

- Solid state drives are now close in speed to RAM
- e.g. 4TB NVMe Gen5: 15GB/sec vs 90GB/sec

Coming soon:

- Intel APX doubles the number of CPU registers

Polynomial System Solving with Gröbner Bases

Let $f(x, y) = x + y^2$ and $g(x, y) = xy - 1$.

Order terms by degree in x first, breaking ties by degree in y.

We compute a syzygy by cancelling leading terms:

$$y \cdot f(x, y) - g(x, y) = y^3 + 1 = h(x, y)$$

... and dividing that by the basis (it doesn't change).

We add $h(x, y)$ to the basis and get two new syzygies:

$$y^3 \cdot f(x, y) - x \cdot h(x, y) = -x + y^5 \xrightarrow{+f} y^5 + y^2 \xrightarrow{-y^2 h} 0$$

$$y^2 \cdot g(x, y) - x \cdot h(x, y) = -x - y^2 \xrightarrow{+f} 0$$

And we have a Gröbner basis $\{x + y^2, xy - 1, y^3 + 1\}$

Gröbner Bases in Practice

- we compute a basis using a graded ordering first (F4)
- we convert that basis to a triangular system (FGLM)
- infinite solutions can be tricky (Gröbner Walk)

Let $f(x, y) = y^2 + x$ and $g(x, y) = xy - 1$, graded with $x > y$.

$$x \cdot f - y \cdot g = x^2 + y = h(x, y)$$

matrix description		x^2y^2	x^3	x^2y	y^3	xy	y^2	x	1
syzygies	$x^2 \cdot f$	1	1	0	0	0	0	0	0
	$y^2 \cdot h$	1	0	0	1	0	0	0	0
	$x \cdot g$	0	0	1	0	0	0	-1	0
	$y \cdot h$	0	0	1	0	0	1	0	0
reductors	$x \cdot h$	0	1	0	0	1	0	0	0
	$y \cdot f$	0	0	0	1	1	0	0	0
	g	0	0	0	0	1	0	0	-1
	f	0	0	0	0	0	1	1	0

Gröbner Bases in Practice

- we compute a basis using a graded ordering first (F4)
- we convert that basis to a triangular system (FGLM)
- infinite solutions can be tricky (Gröbner Walk)

Let $f(x, y) = y^2 + x$ and $g(x, y) = xy - 1$, graded with $x > y$.

$$x \cdot f - y \cdot g = x^2 + y = h(x, y)$$

	x^2y^2	x^3	x^2y	y^3	xy	y^2	x	1
	1	0	0	0	0	0	0	-1
	0	1	0	0	0	0	0	1
both syzygies	0	0	1	0	0	0	-1	0
reduced to 0	0	0	0	1	0	0	0	1
	0	0	0	0	1	0	0	-1
	0	0	0	0	0	1	1	0

- no new leading monomials so $\{f, g, h\}$ is a Gröbner basis

The F4 Algorithm

```
Algorithm F4
Input : polynomials F = [f1,...,fs], a monomial ordering <
Output: a Groebner basis G for <f1,...,fs> with respect to <
G := [];    # the partial basis
P := [];    # the array of syzygy pairs
M := [];    # the array of monomials in A
for each f in F do
  G, P := update_pairs(f);          # initial basis and pairs
while |P| > 0 do
  A, P := select_pairs(P);          # syzygies of least degree
  A, M := symbolic_preproc(A);      # build description of matrix
     A := encode_matrix(A, M);      # encode matrix sparsity pattern
     A := row_reduction(A);         # Gaussian elimination
     A := new_pivot_rows(A);        # rows with new pivots
     A := back_substitution(A);     # inter-reduce new pivots
     B := convert_to_polys(A, M);   # convert to polynomials
  for each b in B do
     G, P := update_pairs(b);       # generate new pairs
return inter_reduce(G);             # produce a reduced basis
```

Symbolic Preprocessing

- start with selected pairs
- process rows in batches, bounding new rows and monomials
- add monomials to an array (atomically)
- add reductors to the matrix (atomically)
- creation of new monomials protected by a lock

Let $f(x,y) = y^2 + x$ and $g(x,y) = xy - 1$ and $h(x,y) = x^2 + y$

matrix description		x^2y^2	x^3	x^2y	y^3	xy	y^2	x	1
syzygies	$x^2 \cdot f$	1	1	0	0	0	0	0	0
	$y^2 \cdot h$	1	0	0	1	0	0	0	0
	$x \cdot g$	0	0	1	0	0	0	-1	0
	$y \cdot h$	0	0	1	0	0	1	0	0
reductors	$x \cdot h$	0	1	0	0	1	0	0	0
	$y \cdot f$	0	0	0	1	1	0	0	0
	g	0	0	0	0	1	0	0	-1
	f	0	0	0	0	0	1	1	0

Encoding the Matrix

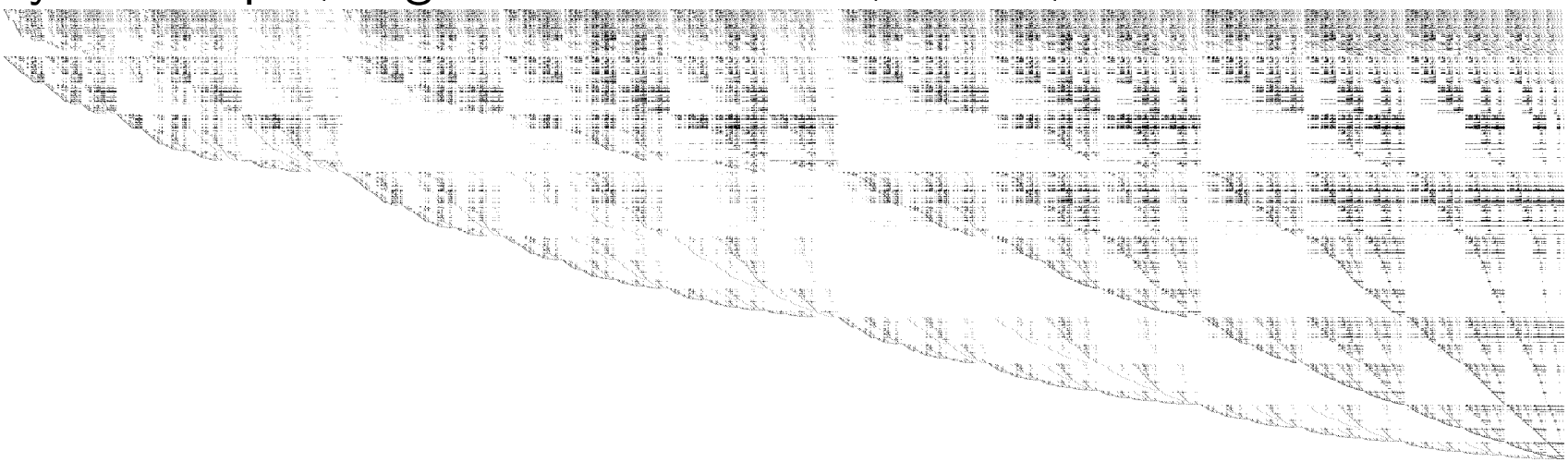
Encoding (compressing) the matrix adds to the cost of F4:

- monomials must be multiplied a second time
- pivot rows must be decoded to reduce

Encode differences into bytes where possible (used by Faugère)

column indices	10000	9990	9989		5000	4950	...
encoded indices	10000	10	1	0	5000	50	...

cyclic9 step 7, degree=8: 899 x 3123, 5.23%, encoded in 150 KB



Probabilistic Gaussian Elimination

Divide the N syzygies to be reduced into blocks of size $3\sqrt[3]{N}$:

- for each block reduce random linear combinations of the rows
- stop when sufficiently many zero reductions are encountered
- add new rows to the pivots array using compare and swap
- if a row can't be added it must be reduced further

```
degree=8 pairs=47/48 with pairs limit 2048
899 x 3123 with 146865 non-zero, 163.4 per row
1.057 bytes per non-zero, matrix encoded in 0.148 MB
94 rows to reduce, blocksize 12, max threads 8, zero reductions 1
12.8/25.5/63.8/38.3/51.1/100.0/89.4/76.6/
new=24 basis=60, step time=0.018 sec
```

- notice we limit the number of pairs in each step
- work over $\mathbb{Z}_p$ due to intermediate expression swell

Time Percentages

- update pairs uses the Gebauer and Möller strategy

The percentage of time spent in various parts of F_4 (sequential)

	gc	select	symbol	encode	reduce	bsub	decode	update
bayes148	0.8	0.4	58.9	27.5	4.5	3.3	0.0	4.5
cyclic9	0.2	0.3	15.4	15.4	62.0	3.3	0.1	3.1
jason210	1.0	1.8	53.6	26.4	11.2	3.3	0.1	0.8
katsura13	0.0	0.8	25.4	25.9	43.7	0.9	0.1	2.2
mayr42	0.9	2.2	15.5	1.6	0.8	0.2	0.0	78.5
noon9	0.2	0.9	31.9	29.7	19.0	4.3	0.1	13.3
reimer8	0.0	0.5	37.3	39.5	21.1	0.7	0.0	0.3

- bayes148, jason210 are sparse (symbolic/encode dominates)
- cyclic9, katsura13 are dense (reduce dominates)
- mayr42 is a pair update benchmark

Benchmarks on a Ryzen 9955HX 16c/32t 2.5–5.4 GHz

	$p = 2^{63} - 25$		$p = 2^{31} - 1$					
	axf4 st	axf4 32t	axf4 st	axf4 32t	msolve 1t	msolve 32t	gamba avx2	gamba avx512
bayes148	38.32	21.08	38.36	21.12	11.27	21.90	4.73	4.76
cyclic9	74.80	8.15	42.19	5.23	31.70	26.48	12.32	10.31
jason210	4.17	5.42	4.09	5.43	1.75	5.31	1.19	1.43
katsura13	44.52	4.47	28.71	3.10	11.58	10.70	5.57	5.77
mayr42	68.37	57.71	68.42	57.07	21.46	47.97	10.36	9.44
noon9	8.52	4.70	7.91	4.60	3.51	9.18	3.92	3.51
reimer8	18.68	2.25	15.52	1.83	9.63	9.36	4.73	5.44

With hyperthreading disabled (less user time = more efficient)

cyclic10 $p = 2^{31} - 1$	axf4 st	axf4 16t	msolve 1t	msolve 16t	gamba avx2	gamba avx512
user time	4255.22	6734.34	3474.20	13296.40	595.52	503.08
wall time	1h 10:59	8:37	57:58	24:12	9:59	8:26
percent cpu	99%	1303%	99%	915%	99%	99%

Cyclic-11 Benchmark

- first computed by Faugère, later by Allan Steel
- 1.1TB of memory, result (text) is about 15GB

```
STEP 135
degree=19 pairs=16440/184574 with pairs limit 131072
3741472 x 4106190 with 79248365332 non-zero, 21181.1 per row
1.031 bytes per non-zero, matrix encoded in 77956.993 MB
32880 rows to reduce, blocksize 96, max threads 343, zero reductions 1
new=1701 basis=102417, step time=49905.165 sec
...
30320 basis elements, 519201.445 sec
```

Question: is this Gröbner basis correct?

- dimension, number of solutions checks out
- we computed it twice

Thank you!