Dynamic Gröbner basis computation

John Perry

University of Southern Mississippi

2019

Outline

1. Motivation, technical background
2. Dynamic algorithms
3. Implementation
4. Conclusions, future directions

Motivation, technical background

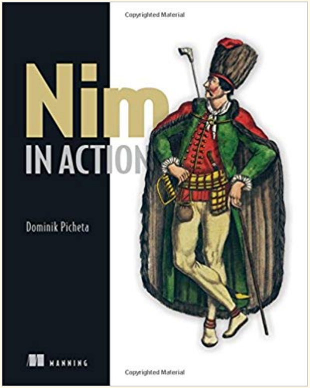
Motivation, technical background:

First World problems

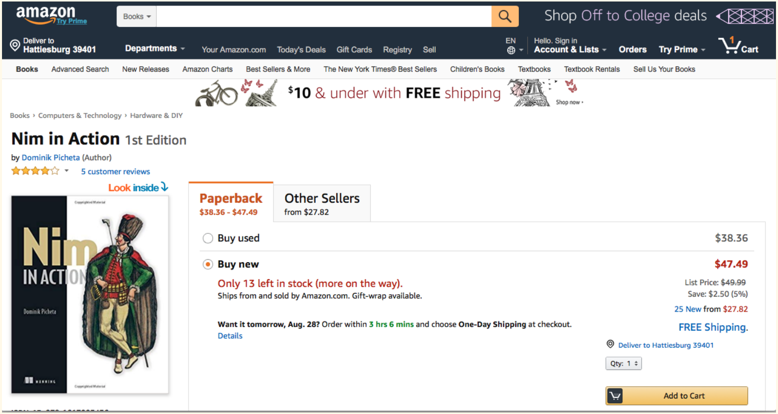
I am a Consumer.

I am a Consumer.
I find meaning in life by Consuming.

I am a Consumer.
I find meaning in life by Consuming.
Right now I want to Consume a book.



AWESOME: Amazon has it in stock!

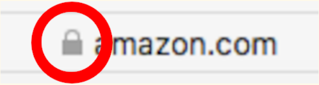


Checkout

What's this thing?

Checkout

For those who haven't noticed it before:



What's a lock doing there?

What's a lock doing there?

You had best be glad it's there.

What's a lock doing there?

You had best be glad it's there.

Excellent, unaltered image downloaded from *The Economist*. Please don't sue. Fair use principles at work.

Public-key cryptography

1. Public knowledge
   - **encryption protocol** E
   - **decryption protocol** D

Public-key cryptography

1. Public knowledge
   - **encryption protocol** E
   - **decryption protocol** D
2. Amazon broadcasts
   - **encryption key** e

Public-key cryptography

1. Public knowledge
   - **encryption protocol** E
   - **decryption protocol** D
2. Amazon broadcasts
   - **encryption key** e
3. Buyer broadcasts $c = E(m, e)$

Public-key cryptography

1. Public knowledge
   - **encryption protocol** E
   - **decryption protocol** D
2. Amazon broadcasts
   - **encryption key** e
3. Buyer broadcasts $c = E(m, e)$
4. Amazon knows
   - **decryption key** d

Public-key cryptography

1. Public knowledge
   - **encryption protocol** E
   - **decryption protocol** D
2. Amazon broadcasts
   - **encryption key** e
3. Buyer broadcasts $c = E(m, e)$
4. Amazon knows
   - **decryption key** d
5. Eavesdropper knows
   - c
   - **encryption key**

   …but cannot decrypt

How is this secure?

- D, E inverse functions: $D(E(m,e),d) = m$

How is this secure?

- D, E inverse functions: $D(E(m, e), d) = m$
- computing d “difficult” **even knowing** D, E, e

How is this secure?

- D, E inverse functions: $D(E(m,e),d) = m$
- computing d "difficult" **even knowing** D, E, e
    - factoring integers
    - discrete logarithm
    - elliptic curve arithmetic

Example

RSA Challenge: Want $200,000?

Factor this 617-digit number.

Big-time fact!

Crucial link in cryptography is RSA (1976, **R**ivest
Shamir
Adelman)

Big-time fact!

Crucial link in cryptography is RSA (1976, **R**ivest
Shamir
Adelman)

Big-time fact!

Cracking RSA communication as "easy" as

$$6 = 2 \times 3$$

RSA

Big-time fact!

Crucial link in cryptography is RSA (1976, **R**ivest
Shamir
Adelman)

Big-time fact!

Cracking RSA communication as “easy” as

$$6 = 2 \times 3$$

Big-time fact!

If that doesn’t bother you, it should.

Quantum challenge

1994 Shor: "quantum algorithm" to factor integers "fast" (in polynomial time)

RSA Challenge

Quantum challenge

1994 Shor: "quantum algorithm" to factor integers "fast" (in polynomial time)

2001 IBM "quantum factors" $15 = 3 \times 5$

Quantum challenge

- 1994 Shor: “quantum algorithm” to factor integers “fast” (in polynomial time)
- 2001 IBM “quantum factors” $15 = 3 \times 5$
- 2012 Xu, Zhu, Lu, Xhou, Peng, Du “quantum factor” $143 = 11 \times 13$

Quantum challenge

- **1994** Shor: "quantum algorithm" to factor integers "fast" (in polynomial time)
- **2001** IBM "quantum factors" $15 = 3 \times 5$
- **2012** Xu, Zhu, Lu, Xhou, Peng, Du
  "quantum factor" $143 = 11 \times 13$
- **2014** Dattani & Bryans:
  cool! they also factored $56153 = 233 \times 241$

RSA Challenge

Quantum challenge

- 1994 Shor: "quantum algorithm" to factor integers "fast" (in polynomial time)
- 2001 IBM "quantum factors" $15 = 3 \times 5$
- 2012 Xu, Zhu, Lu, Xhou, Peng, Du "quantum factor" $143 = 11 \times 13$
- 2014 Dattani & Bryans: *cool!* they also factored $56153 = 233 \times 241$ ("without the awareness of the authors")

Quantum challenge

1994 Shor: "quantum algorithm" to factor integers "fast" (in polynomial time)

2001 IBM "quantum factors" $15 = 3 \times 5$

2012 Xu, Zhu, Lu, Xhou, Peng, Du
"quantum factor" $143 = 11 \times 13$

2014 Dattani & Bryans:
cool! they also factored $56153 = 233 \times 241$
("without the awareness of the authors")

2017 NIST: requests proposals
for **post-quantum cryptography**

Motivation, technical background:

McEliece

McEliece

- Amazon chooses matrices
  - G (linear code correcting t errors)
  - S, P (nonsingular, permutation)
  - $\widehat{G} = SGP$ (public key)

- Amazon chooses matrices
  - G (linear code correcting t errors)
  - S, P (nonsingular, permutation)
  - $\widehat{G} = SGP$ (public key)
- Amazon broadcasts t, $\widehat{G}$

McEliece

- Amazon chooses matrices
  - G (linear code correcting t errors)
  - S, P (nonsingular, permutation)
  - $\widehat{G} = SGP$ (public key)
- Amazon broadcasts $t, \widehat{G}$
- Buyer broadcasts $c = m\widehat{G} + z$ (message + t errors)

McEliece

- Amazon chooses matrices
  - G (linear code correcting t errors)
  - S, P (nonsingular, permutation)
  - $\widehat{G} = SGP$ (public key)
- Amazon broadcasts $t, \widehat{G}$
- Buyer broadcasts $c = m\widehat{G} + z$ (message + t errors)
- Only Amazon knows
  - G (can identify, remove z)
  - S, P (hence S^{-1}, P^{-1})

…decrypts $m = (c - z)\,\widehat{G}^{-1}$

McEliece

- Amazon chooses matrices
  - G (linear code correcting t errors)
  - S, P (nonsingular, permutation)
  - $\widehat{G} = SGP$ (public key)
- Amazon broadcasts $t, \widehat{G}$
- Buyer broadcasts $c = m\widehat{G} + z$ (message + t errors)
- Only Amazon knows
  - G (can identify, remove z)
  - S, P (hence S^{-1}, P^{-1})

  …decrypts $m = (c - z)\,\widehat{G}^{-1}$
- Eavesdropper knows
  - $c, t, \widehat{G}$
  - encryption and decryption methods

  …but cannot decrypt

Good idea?

Big-time fact!

"Classic McEliece" susceptible to attacks? **Unknown**, but NIST accepted it for post-Quantum standard (2019).

Good idea?

Big-time fact!

“Classic McEliece” susceptible to attacks? **Unknown**, but NIST accepted it for post-Quantum standard (2019).

Big-time fact!

Why do you never hear about it?

Good idea?

Big-time fact!

“Classic McEliece” susceptible to attacks? **Unknown**, but NIST accepted it for post-Quantum standard (2019).

Big-time fact!

Why do you never hear about it?

- encryption, decryption faster than RSA, but…
- key **very large**

Good idea?

Big-time fact!

"Classic McEliece" susceptible to attacks? **Unknown**, but NIST accepted it for post-Quantum standard (2019).

Big-time fact!

Why do you never hear about it?

- encryption, decryption faster than RSA, but…
- key **very large**

Proposal: "McEliece Variants"

(2009) Choose G w/*particular form*

Bad idea!

Bad idea!

Big-time fact!

McEliece Variants cracked in < 1 s! (Faugère et al., 2010)

Bad idea!

Big-time fact!

McEliece Variants cracked in < 1 s! (Faugère et al., 2010)

How?

1. System $\rightsquigarrow$ polynomial equations
2. Structure $\rightsquigarrow$ simplify, rewrite
3. **Gröbner basis**
4. Keep "smallest" polys

Bad idea!

Big-time fact!

McEliece Variants cracked in < 1 s! (Faugère et al., 2010)

How?

1. System $\rightsquigarrow$ polynomial equations
2. Structure $\rightsquigarrow$ simplify, rewrite
3. **Gröbner basis**
4. Keep "smallest" polys

Question

Gröbner basis?

Motivation, technical background:

Gröbner bases

Analogies

	linear
structure	vector space
multipliers	field elements
workspace	subspace
presentation	spanning set
***good* presentation**	basis
transformation	Gauss-Jordan

Analogies

	linear	non-linear
structure	vector space	polynomial ring
multipliers	field elements	monomials
workspace	subspace	ideal
presentation	spanning set	basis
***good* presentation**	basis	Gröbner basis
transformation	Gauss-Jordan	Buchberger

Analogies

	linear	non-linear
structure	vector space	polynomial ring
multipliers	field elements	monomials
workspace	subspace	ideal
presentation	spanning set	basis
***good* presentation**	basis	Gröbner basis
transformation	Gauss-Jordan	Buchberger

Example: linear

$$\begin{aligned} 2x + 3y &= 4 \\ x + 2y &= 3 \end{aligned}$$

Example: linear

$$\begin{aligned} 2x + 3y &= 4 \\ x + 2y &= 3 \end{aligned} \quad \Longrightarrow \quad \begin{array}{r} 2x + 3y = 4 \\ -\quad 2x + 4y = 6 \\ \hline -y = -2 \end{array}$$

row reduction

Example: non-linear

$$x^2 + y^2 = 4$$
$$xy = 1$$

Example: non-linear

$$\begin{aligned} x^2 + y^2 &= 4 \\ xy &= 1 \end{aligned} \quad \Longrightarrow \quad \begin{array}{r} x^2y + y^3 = 4y \\ -\quad x^2y \qquad\quad = x \\ \hline y^3 = 4y - x \end{array}$$

***S*-polynomial** reduction

Gauss-Jordan reduction

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce 1st by second

until all pairs reduce to zero

return basis

Buchberger's algorithm

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1\}$

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:
(x^2, xy)

Example, again

Find a Gröbner basis for $\left\{x^2+y^2-4, xy-1\right\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:

~~(x^2, xy)~~

Example, again

Find a Gröbner basis for $\left\{x^2 + y^2 - 4, xy - 1\right\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:

$\cancel{\left(x^2, xy\right)}$

s-poly:

$\left(\cancel{x^2 y} + y^3 - 4y\right) - \left(\cancel{x^2 y} - x\right)$

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:
~~(x^2, xy)~~, (x^2, y^3), (xy, y^3)

s-poly:
$y^3 + x - 4y$

Example, again

Find a Gröbner basis for $\left\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\right\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:
~~(x^2, xy)~~, ~~(x^2, y^3)~~, (xy, y^3)

s-poly:

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:

$\cancel{(x^2, xy)}, \cancel{(x^2, y^3)}, (xy, y^3)$

s-poly:

$$\left(\cancel{x^2y^3} + y^5 - 4y^3\right) - \left(\cancel{x^2y^3} + x^3 - 4x^2y\right)$$

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:

~~(x^2, xy)~~, ~~(x^2, y^3)~~, (xy, y^3)

s-poly:

$y^5 - x^3 + 4x^2y - 4y^3$

Example, again

Find a Gröbner basis for $\left\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\right\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:

~~(x^2, xy)~~, ~~(x^2, y^3)~~, (xy, y^3)

s-poly:

$$\left(\cancel{y^5} - x^3 + 4x^2y - 4y^3\right) - \left(\cancel{y^5} + xy^2 - 4y^3\right)$$

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:
~~(x^2, xy)~~, ~~(x^2, y^3)~~, (xy, y^3)

s-poly:
$-x^3 + 4x^2y - xy^2$

Example, again

Find a Gröbner basis for $\left\{x^2+y^2-4,\ xy-1,\ y^3+x-4y\right\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:

$\not{(x^2, xy)}, \not{(x^2, y^3)}, (xy, y^3)$

s-poly:

$$\left(\not{-x^3}+4x^2y-xy^2\right)$$
$$+\left(\not{x^3}+xy^2-4x\right)$$

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:
~~(x^2, xy)~~, ~~(x^2, y^3)~~, (xy, y^3)

s-poly:
$4x^2y - 4x$

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:
~~(x^2, xy)~~, ~~(x^2, y^3)~~, (xy, y^3)

s-poly:
$(\not{4x^2y} - 4x) - (\not{4x^2y} - 4x)$

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:
~~(x^2, xy)~~, ~~(x^2, y^3)~~, (xy, y^3)

s-poly:
0

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:
~~(x^2, xy)~~, ~~(x^2, y^3)~~, (xy, y^3)

s-poly:
0

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:
~~(x^2, xy)~~, ~~(x^2, y^3)~~, ~~(xy, y^3)~~

s-poly:

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:

~~(x^2, xy)~~, ~~(x^2, y^3)~~, ~~(xy, y^3)~~

s-poly:

$$(\cancel{xy^3} - y^2) - (\cancel{xy^3} + x^2 - 4xy)$$

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:
~~(x^2, xy)~~, ~~(x^2, y^3)~~, ~~(xy, y^3)~~

s-poly:
$-x^2 + 4xy - y^2$

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:

$(x^2, xy), (x^2, y^3), (xy, y^3)$

s-poly:

$$\left(-x^2 + 4xy - y^2\right) + \left(x^2 + y^2 - 4\right)$$

Example, again

Find a Gröbner basis for $\left\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\right\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:
~~(x^2, xy)~~, ~~(x^2, y^3)~~, ~~(xy, y^3)~~

s-poly:
$4xy - 4$

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:
~~(x^2, xy)~~, ~~(x^2, y^3)~~, ~~(xy, y^3)~~

s-poly:
$(\cancel{4xy} - 4) - (\cancel{4xy} - 4)$

Example, again

Find a Gröbner basis for $\left\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\right\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:
$\cancel{(x^2, xy)}, \cancel{(x^2, y^3)}, \cancel{(xy, y^3)}$

s-poly:
0

Example, again

Find a Gröbner basis for $\left\{x^2+y^2-4, xy-1, y^3+x-4y\right\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:
~~(x^2, xy)~~, ~~(x^2, y^3)~~, ~~(xy, y^3)~~

s-poly:
0

Example, again

Find a Gröbner basis for $\{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- nonzero? add new poly

until all pairs reduce to zero

return basis

Pairs:
~~(x^2, xy)~~, ~~(x^2, y^3)~~, ~~(xy, y^3)~~

s-poly:

Analogy goes deeper

vector space bases, Gröbner bases both answer…

- existence of solutions
- dimension of solutions
- explicit description of solutions
- which vectors are in subspace

Analogy goes deeper

vector space bases, Gröbner bases both answer…

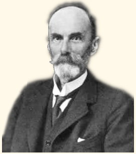
- existence of solutions
- dimension of solutions
- explicit description of solutions
- which vectors are in subspace

…in similar ways!

…still deeper! [Macaulay, 1902, Lazard, 1983]

Gaussian reduction $\rightsquigarrow$ Buchberger's algorithm

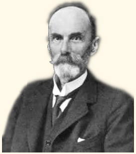
$$\begin{pmatrix} x^2 & xy & y^2 & 1 & \\ 1 & & 1 & -4 & \mathbf{g} \\ & 1 & & -1 & \mathbf{f} \end{pmatrix}$$



…still deeper! [Macaulay, 1902, Lazard, 1983]

Gaussian reduction $\rightsquigarrow$ Buchberger's algorithm

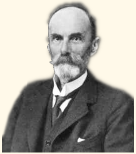
xy^2	x^2	xy	y^2	y	1	
	1		1		−4	**g**
1				−1		y**f**
		1			−1	**f**



…still deeper! [Macaulay, 1902, Lazard, 1983]

Gaussian reduction $\leftrightsquigarrow$ Buchberger's algorithm

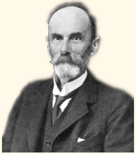
x^2y	xy^2	x^2	xy	y^2	x	y	1	
		1		1			−4	**g**
1					− 1			x**f**
	1					− 1		y**f**
			1				−1	**f**



…still deeper! [Macaulay, 1902, Lazard, 1983]

Gaussian reduction ⇜⇝ Buchberger's algorithm

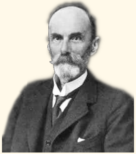
$$\left(\begin{array}{cccccccccccc}
\cdots & & x^2y & xy^2 & y^3 & x^2 & xy & y^2 & x & y & 1 & \\
 & & & & & & & & & & & \\
 & & 1 & & 1 & & & & & -4 & & y\mathbf{g} \\
 & & & & & 1 & & 1 & & & -4 & \mathbf{g} \\
 & \ddots & & & & & & \ddots & & & & \vdots \\
 & & 1 & & & & & & -1 & & & x\mathbf{f} \\
 & & & 1 & & & & & & -1 & & y\mathbf{f} \\
 & & & & & & 1 & & & & -1 & \mathbf{f}
\end{array}\right)$$



…still deeper! [Macaulay, 1902, Lazard, 1983]

Gaussian reduction $\leftrightsquigarrow$ Buchberger's algorithm

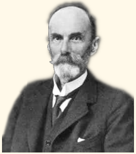
$\cdots$	x^3	x^2y	xy^2	y^3	x^2	xy	y^2	x	y	1	
	1		1					-4			$x\mathbf{g}$
		1		1					-4		$y\mathbf{g}$
					1		1			-4	$\mathbf{g}$
	$\ddots$						$\ddots$				$\vdots$
		1						-1			$x\mathbf{f}$
			1						-1		$y\mathbf{f}$
						1				-1	$\mathbf{f}$



…still deeper! [Macaulay, 1902, Lazard, 1983]

Gaussian reduction ⇜⇝ Buchberger's algorithm

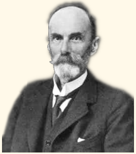
$\cdots$	x^3	x^2y	xy^2	y^3	x^2	xy	y^2	x	y	1	
$\ddots$		$\ddots$					$\ddots$				$\vdots$
	1		1					−4			$x\mathbf{g}$
		1		1					−4		$y\mathbf{g}$
					1		1			−4	$\mathbf{g}$
	$\ddots$						$\ddots$				$\vdots$
		1						−1			$x\mathbf{f}$
			1						−1		$y\mathbf{f}$
						1				−1	$\mathbf{f}$



…still deeper! [Macaulay, 1902, Lazard, 1983]

Gaussian reduction $\leftrightsquigarrow$ Buchberger's algorithm

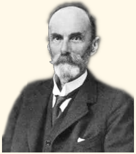
⋯	x^3	x^2y	xy^2	y^3	x^2	xy	y^2	x	y	1	
⋱		⋱					⋱				⋮
	1		1					−4			x**g**
		0		1				1	−4		y**g**
					1		1			−4	**g**
	⋱						⋱				⋮
		1						−1			x**f**
			1						−1		y**f**
						1				−1	**f**



…still deeper! [Macaulay, 1902, Lazard, 1983]

Gaussian reduction ⇝ Buchberger's algorithm

$\cdots$	x^3	x^2y	xy^2	y^3	x^2	xy	y^2	x	y	1	
$\ddots$		$\ddots$					$\ddots$				$\vdots$
	1		1					-4			xg
				1				1	-4		yg
					1		1			-4	g
	$\ddots$						$\ddots$				$\vdots$
		1						-1			xf
			1						-1		yf
						1				-1	f



GB: $\{ \quad xy-1, \quad x^2+y^2-4, \quad y^3+x-4 \quad \}$

Algorithm F4 [Faugère, 1999]

inputs F, $>$

repeat

- build "important submatrix"
- perform Gauss-Jordan

until all important submatrices triangular

$G \leftarrow$ important submatrices' rows

return G

Advantages

1. Locality of data — speed

Advantages

1. Locality of data — speed
2. Easy to parallelize — more speed

Advantages

1. Locality of data — speed
2. Easy to parallelize — more speed
3. Sparse matrix data structures, algorithms — still more speed (and well-studied)

Motivation, technical background:
Termination

Nagging question

How do we know these algorithms terminate?

basis	***pair chosen***	**pairs remaining**
$\left\{ \begin{array}{c} x^2 + y^2 - 4, \\ xy - 1 \end{array} \right\}$		$\{(1,2)\}$

Nagging question

How do we know these algorithms terminate?

basis	***pair chosen***	**pairs remaining**
$\left\{ \begin{array}{c} x^2 + y^2 - 4, \\ xy - 1 \end{array} \right\}$	$(1, 2)$	$\{\}$

Nagging question

How do we know these algorithms terminate?

basis	***pair chosen***	**pairs remaining**
$\left\{ \begin{array}{c} x^2 + y^2 - 4, \\ xy - 1, \\ y^3 + x - 4y \end{array} \right\}$		$\{(1,3),(2,3)\}$

Nagging question

How do we know these algorithms terminate?

basis	***pair chosen***	**pairs remaining**
$\left\{ \begin{array}{c} x^2 + y^2 - 4, \\ xy - 1, \\ y^3 + x - 4y \end{array} \right\}$	$(1, 3)$	$\{(2, 3)\}$

Nagging question

How do we know these algorithms terminate?

basis	***pair chosen***	**pairs remaining**
$\left\{\begin{array}{c} x^2+y^2-4, \\ xy-1, \\ y^3+x-4y \end{array}\right\}$	$(2,3)$	$\{\}$

Nagging question

How do we know these algorithms terminate?

basis	***pair chosen***	**pairs remaining**
$\left\{\begin{array}{c} x^2+y^2-4, \\ xy-1, \\ y^3+x-4y \end{array}\right\}$		

- number of pairs can increase

Nagging question

How do we know these algorithms terminate?

basis	*pair chosen*	pairs remaining
$\left\{ \begin{array}{c} x^2 + y^2 - 4, \\ xy - 1, \\ y^3 + x - 4y \end{array} \right\}$		

- number of pairs can increase
- can it increase without end?

Hilbert Basis Theorem

"Every polynomial ideal over a field is finitely generated"

Hilbert Basis Theorem

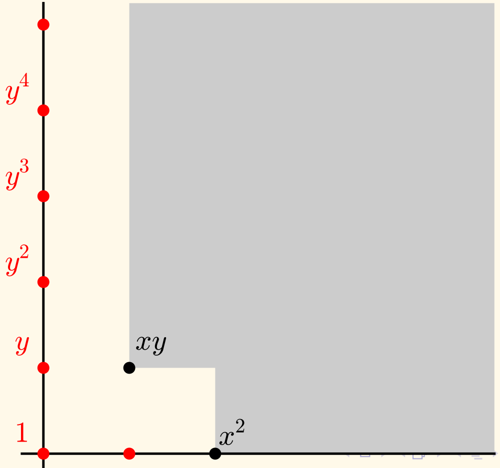
"Every polynomial ideal over a field is finitely generated"

new polys $\Longrightarrow$ "fewer" irreducible terms

“Every polynomial ideal over a field is finitely generated”

new polys $\Longrightarrow$ “fewer” irreducible terms

Well-Ordering of $\mathbb{N}$, generalized

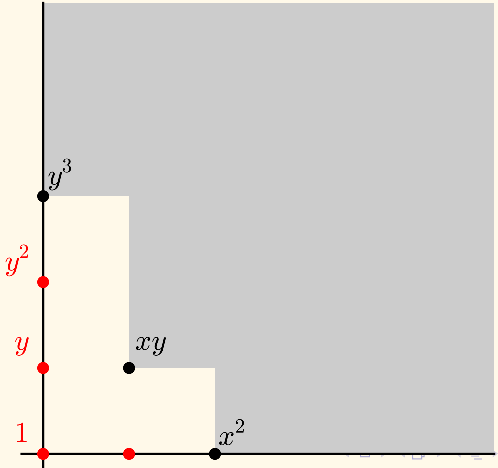


Hilbert Basis Theorem

"Every polynomial ideal over a field is finitely generated"

new polys $\Longrightarrow$ "fewer" irreducible terms

Well-Ordering of $\mathbb{N}$, generalized



Term orderings

- Termination tied to term ordering
- What is a term ordering? *technical details*

How large can they grow?

benchmark	***#vars***	***#polys***	***avg #mons***	***secs (Buch)***
butcher8	8	30	44	2.64
noon5	5	72	111	0.26
kotsireas	6	92	150	7.19

How large can they grow?

benchmark	*#vars*	*#polys*	*avg #mons*	*secs (Buch)*
cyclic-4	4	7	6	.007
cyclic-5	5	38	21	.053
cyclic-6	6	99	60	.835
cyclic-7	7	443	325	101.6
cyclic-8	8	1033	N/C	>2000
cyclic-9	9	5601	N/C	hours

How large can they grow?

benchmark	***#vars***	***#polys***	***avg #mons***	***secs (Buch)***
cyclic-4	4	7	6	.007
cyclic-5	5	38	21	.053
cyclic-6	6	99	60	.835
cyclic-7	7	443	325	101.6
cyclic-8	8	1033	N/C	>2000
cyclic-9	9	5601	N/C	hours

5601 is a lot. Can we make it smaller?

How *low* can we go?

benchmark	***#vars***	***#polys***	***avg #mons***	***ratios***
cyclic-4	4	4	4	.57, .67
cyclic-5	5	11	13	.29, .62
cyclic-6	6	26	41	.26, .68
cyclic-7	7	106	150	.24, .46
cyclic-8	8	404	N/C	.39, ?
cyclic-9	9	1996	N/C	.36, ?

How do we get these smaller bases?

Come back for part 2.

Dynamic algorithms

Dynamic algorithms:
Idea

Triangularize?

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & \\ 1 & 1 & 1 & & \\ 1 & 1 & & & \\ 1 & & & & \end{pmatrix}$$

Optimization in linear algebra

Triangularize?

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & \\ 1 & 1 & 1 & & \\ 1 & 1 & & & \\ 1 & & & & \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ & 1 & 1 & 1 & 1 \\ & & 1 & 1 & 1 \\ & & & 1 & 1 \\ & & & & 1 \end{pmatrix}$$

No reduction needed!

Optimization in linear algebra

Triangularize?

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & \\ 1 & 1 & 1 & & \\ 1 & 1 & & & \\ 1 & & & & \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ & 1 & 1 & 1 & 1 \\ & & 1 & 1 & 1 \\ & & & 1 & 1 \\ & & & & 1 \end{pmatrix}$$

No reduction needed!

Can we do this with Gröbner bases?

Why?

1. Why not?

Why?

1. Why not?
2. Might be "practical"
    - faster?
        - maybe...
        - ...but not if overhead is too large

Why?

1. Why not?

2. Might be "practical"

   - faster?
     - maybe…
     - …but not if overhead is too large

   - smaller basis?
     - faster application
     - tradeoff could be worth it!

No major CAS does this

Why not?

- monomials $\leftrightsquigarrow$ columns
- swap columns? $\rightsquigarrow$ out of order!
- out of order? $\rightsquigarrow$ not Gröbner!

hard to implement w/out breaking other things

No major CAS does this

Why not?

- monomials $\leftrightsquigarrow$ columns
- swap columns? $\rightsquigarrow$ out of order!
- out of order? $\rightsquigarrow$ not Gröbner!

hard to implement w/out breaking other things

Non-trivial problem

For example

$$
\begin{pmatrix}
\cdots & x^3 & x^2y & xy^2 & y^3 & x^2 & xy & y^2 & x & y & 1 & \\
\ddots & & \ddots & & & & & \ddots & & & & \vdots \\
 & 1 & & 1 & & & & & -4 & & & x\mathbf{g} \\
 & & 1 & & 1 & & & & & -4 & & y\mathbf{g} \\
 & & & & & 1 & & 1 & & & -4 & \mathbf{g} \\
 & \ddots & & & & & & \ddots & & & & \vdots \\
 & & 1 & & & & & & -1 & & & x\mathbf{f} \\
 & & & 1 & & & & & & -1 & & y\mathbf{f} \\
 & & & & & & 1 & & & & -1 & \mathbf{f}
\end{pmatrix}
$$

For example

$$
\begin{pmatrix}
\cdots & x^3 & x^2y & xy^2 & y^3 & x^2 & xy & y^2 & x & y & 1 & \\
\ddots & & \ddots & & & & & \ddots & & & & \vdots \\
& 1 & & 1 & & & & & -4 & & & xg \\
& & \color{red}{1} & & \color{red}{1} & & & & & \color{red}{-4} & & yg \\
& & & & & 1 & & 1 & & & -4 & g \\
& \ddots & & & & & & \ddots & & & & \vdots \\
& & \color{red}{1} & & & & & & \color{red}{-1} & & & xf \\
& & & 1 & & & & & & -1 & & yf \\
& & & & & & 1 & & & & -1 & f
\end{pmatrix}
$$

For example

$$
\left(\begin{array}{cccccccccccc}
\cdots & x^3 & \color{red}{x^2y} & xy^2 & y^3 & x^2 & xy & \color{red}{y^2} & x & y & 1 & \\
\ddots & & \ddots & & & & & \ddots & & & & \vdots \\
 & 1 & & 1 & & & & & -4 & & & x\mathbf{g} \\
 & & \color{red}{1} & & 1 & & & & & -4 & & y\mathbf{g} \\
 & & & & & 1 & & \color{red}{1} & & & -4 & \mathbf{g} \\
 & \ddots & & & & & & \ddots & & & & \vdots \\
 & & \color{red}{1} & & & & & & -1 & & & x\mathbf{f} \\
 & & & 1 & & & & & & -1 & & y\mathbf{f} \\
 & & & & & & 1 & & & & -1 & \mathbf{f}
\end{array}\right)
$$

For example

$$\begin{pmatrix}
\cdots & x^3 & y^2 & xy^2 & y^3 & x^2 & xy & x^2y & x & y & 1 & \\
\ddots & & \ddots & & & & & \ddots & & & & \vdots \\
& 1 & & 1 & & & & & -4 & & & x\mathbf{g} \\
& & & & 1 & & & 1 & & -4 & & y\mathbf{g} \\
& & 1 & & & 1 & & & & & -4 & \mathbf{g} \\
& \ddots & & & & & & \ddots & & & & \vdots \\
& & & & & & & 1 & -1 & & & x\mathbf{f} \\
& & & 1 & & & & & & -1 & & y\mathbf{f} \\
& & & & & & 1 & & & & -1 & \mathbf{f}
\end{pmatrix}$$

For example

$$
\begin{pmatrix}
\cdots & x^3 & y^2 & xy^2 & y^3 & x^2 & xy & x^2y & x & y & 1 & \\
\ddots & & \ddots & & & & & \ddots & & & & \vdots \\
 & 1 & & 1 & & & & & -4 & & & x\mathbf{g} \\
 & & & & 1 & & & 1 & & -4 & & y\mathbf{g} \\
 & & 1 & & & 1 & & & & & -4 & \mathbf{g} \\
 & \ddots & & & & & & \ddots & & & & \vdots \\
 & & & & & & & 1 & -1 & & & x\mathbf{f} \\
 & & & 1 & & & & & & -1 & & y\mathbf{f} \\
 & & & & & & 1 & & & & -1 & \mathbf{f}
\end{pmatrix}
$$

$\{x^2y + x - 4y, y^2 + x^2 - 4, x^2y - 1\}$ not GB under any order

Past work

Dynamic Buchberger implemented, improved, explored

[Caboara, 1993]	AlPI, CoCoA (lost)
[Golubitsky, unpublished]	Maple
[Caboara & Perry, 2014]	Sage
[Hashemi & Talaashrafi, 2016]	Sage
[Perry, 2017]	C++
[Langeloh, 2019]	Sage

*[Gritzmann & Sturmfels, 1993] on theory, not implementation

Dynamic algorithms:

Ordering the columns

Term orderings

- **W**eighted **deg**ree ordering: $\omega = (\omega_1, \ldots, \omega_n)$
- Dot product with exponent vector

- **W**eighted **deg**ree ordering: $\omega = (\omega_1, \ldots, \omega_n)$
- Dot product with exponent vector

Example

$$(1, 0, 0)$$

$$x^3 + y^4 + x^2 z$$

$$3 > 0, 2$$

Term orderings

- **W**eighted **deg**ree ordering: $\omega = (\omega_1, \ldots, \omega_n)$
- Dot product with exponent vector

Example

$$(1,1,1)$$

$$x^3 + y^4 + x^2 z$$

$$4 > 3, 3$$

Term orderings

- **W**eighted **deg**ree ordering: $\omega = (\omega_1, \ldots, \omega_n)$
- Dot product with exponent vector

Example

$$(1, 1, 3)$$

$$x^3 + y^4 + x^2 z$$

$$5 > 3, 4$$

Term orderings

- **W**eighted **deg**ree ordering: $\omega = (\omega_1, \ldots, \omega_n)$
- Dot product with exponent vector

Example

$$x^3 + y^4 + x^2 z$$

Break ties w/additional vectors

Effect

$$\left\{ \begin{array}{c} x_1 + x_2 + \cdots + x_7, \\ x_1 x_2 + x_2 x_3 + \cdots + x_7 x_1, \\ x_1 x_2 x_3 + x_2 x_3 x_4 + \cdots + x_7 x_1 x_2, \\ \vdots \\ x_1 x_2 x_3 x_4 x_5 x_6 x_7 - h^7 \end{array} \right\}$$

#-spolys			
#polys			
time (sec)			

Effect

$$\left\{\begin{array}{c} x_1 + x_2 + \cdots + x_7, \\ x_1x_2 + x_2x_3 + \cdots + x_7x_1, \\ x_1x_2x_3 + x_2x_3x_4 + \cdots + x_7x_1x_2, \\ \vdots \\ x_1x_2x_3x_4x_5x_6x_7 - h^7 \end{array}\right\}$$

	lex $\begin{pmatrix} 1 & & \\ & 1 & \\ & & \ddots \end{pmatrix}$		
#-spolys	5463		
#polys	977		
time (sec)	14401		

Effect

$$\left\{ \begin{array}{c} x_1 + x_2 + \cdots + x_7, \\ x_1x_2 + x_2x_3 + \cdots + x_7x_1, \\ x_1x_2x_3 + x_2x_3x_4 + \cdots + x_7x_1x_2, \\ \vdots \\ x_1x_2x_3x_4x_5x_6x_7 - h^7 \end{array} \right\}$$

	lex $\begin{pmatrix} 1 & & \\ & 1 & \\ & & \ddots \end{pmatrix}$	grevlex $\begin{pmatrix} \cdots & 1 & 1 & 1 \\ \cdots & 1 & 1 & \\ \cdots & 1 & & \end{pmatrix}$	
#-spolys	5463	2199	
#polys	977	443	
time (sec)	14401	11.1	

Effect

$$\left\{\begin{array}{c} x_1+x_2+\cdots+x_7, \\ x_1x_2+x_2x_3+\cdots+x_7x_1, \\ x_1x_2x_3+x_2x_3x_4+\cdots+x_7x_1x_2, \\ \vdots \\ x_1x_2x_3x_4x_5x_6x_7-h^7 \end{array}\right\}$$

	lex	grevlex	wdeg
	$\begin{pmatrix} 1 & & \\ & 1 & \\ & & \ddots \end{pmatrix}$	$\begin{pmatrix} \cdots & 1 & 1 & 1 \\ \cdots & 1 & 1 & \\ \cdots & 1 & & \end{pmatrix}$	2526, 1461, 2625, 2639, 1, 2702, 2703, 1714
#-spolys	5463	2199	404
#polys	977	443	107
time (sec)	14401	11.1	5.0

Compute ordering, subject to constraints?

$$x_1^{\alpha_1} \cdots x_n^{\alpha_n} \quad > \quad x_1^{\beta_1} \cdots x_n^{\beta_n}$$

$$\alpha \cdot \omega \quad > \quad \beta \cdot \omega$$

$$\sum (\alpha_i - \beta_i)\, \omega_i \quad > 0$$

Compute ordering, subject to constraints?

$$\left\{ \left\{ \sum (\alpha_i - \beta_i)\,\omega_i > 0 \right\}_{\mathbf{x}^{\boldsymbol{\beta}} \in \operatorname{Supp}\{g\}} \right\}_{g \in G}$$

inexact: simplex (GLPK)

exact: double description method (PPL)

Why inexact simplex?

Simplex can be performed exactly, so why use it inexactly?

Why inexact simplex?

Simplex can be performed exactly, so why use it inexactly?
CAS's require *integer* entries

- simplex w/float *fast*
- simplex w/int ***slower than molasses in winter***

Why inexact simplex?

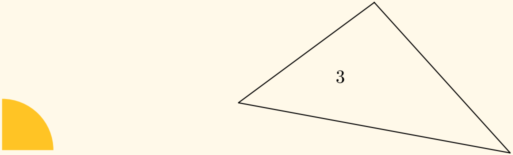
Simplex can be performed exactly, so why use it inexactly?
CAS's require *integer* entries

- simplex w/float *fast*
- simplex w/int ***slower than molasses in winter***

Besides, we want a ***skeleton***

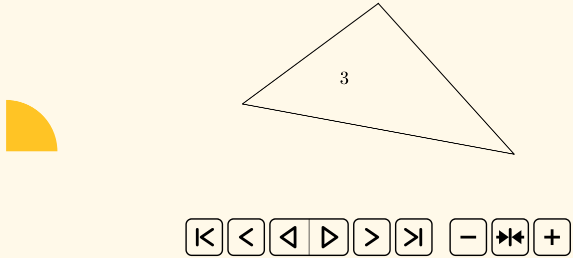
Skeleton?

- add monomials $\Longrightarrow$ add constraints



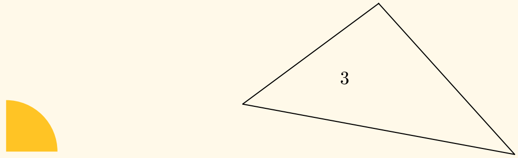
Skeleton?

- add monomials $\Longrightarrow$ add constraints



Skeleton?

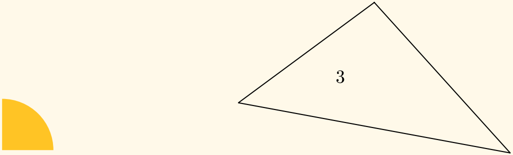
- add monomials $\Longrightarrow$ add constraints



- ***skeleton:*** extreme vectors ("corners")

Skeleton?

- add monomials $\Longrightarrow$ add constraints



- ***skeleton:*** extreme vectors (“corners”)
- use to identify potential leading monomials

The trouble with simplex

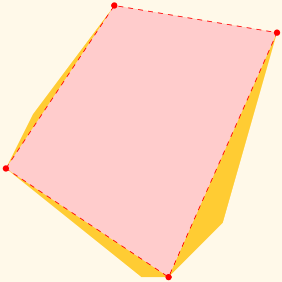
Finds only one extreme vector at a time

The trouble with simplex

Finds only one extreme vector at a time

…*useful*, but *incomplete*

Finds only one extreme vector at a time



…*useful*, but *incomplete*

Double description method

- Exact method
- Iterative
  - add constraints as needed
  - detect, discard redundant constraints
- Well-studied
  - Motzkin et al., 1953
  - Fukuda and Prodon, 1996
  - [Zolotykh, 2012]
- Worst case not great (exponential complexity)

Dynamic algorithms:

Evaluating orderings

Hilbert heuristic

Definition ($HF(d)$)

minimize Hilbert data (dimension, number of residues)

Educated?

Efficient?

Hilbert heuristic

Definition ($HF(d)$)

minimize Hilbert data (dimension, number of residues)

Educated?

- invariant of homogeneous ideal
  - approaching from above
  - proposed by smart people [Gritzmann & Sturmfels, 1993, Caboara, 1993]

Efficient?

Hilbert heuristic

Definition ($HF(d)$)

minimize Hilbert data (dimension, number of residues)

Educated?

- invariant of homogeneous ideal
  - approaching from above
  - proposed by smart people [Gritzmann & Sturmfels, 1993, Caboara, 1993]

Efficient?

- well-studied [Bigatti, 1997, Roune, 2010]
  - HP coeffs can grow *very large*

Critical pairs heuristic

Definition (β)

minimize number of critical pairs after adding monomial

Educated?

Efficient?

Critical pairs heuristic

Definition (β)

minimize number of critical pairs after adding monomial

Educated?

- want to minimize
- related to invariant of homogeneous ideal (Betti number)
- proposed by smart person (Eder)

Efficient?

Critical pairs heuristic

Definition (β)

minimize number of critical pairs after adding monomial

Educated?

- want to minimize
- related to invariant of homogeneous ideal (Betti number)
- proposed by smart person (Eder)

Efficient?

- well-studied [Gebauer & Möller, 1988]
- minimal overhead

Implementation

Implementation:

Dynamic algorithms

Buchberger Algorithm

inputs bad basis, $>$

repeat

- choose unconsidered pair
- reduce s-poly
- non-zero poly? add new poly

until all pairs reduce to zero

return basis

Dynamic Buchberger Algorithm

inputs bad basis

choose $>$
repeat

- choose unconsidered pair
- reduce s-poly
- non-zero poly? add new poly
  - reconsider $>$

until all pairs reduce to zero

return basis

F4 Algorithm

inputs bad basis, $>$

repeat

- build “important submatrix”
- perform Gauss-Jordan

until all important submatrices triangular

$G \leftarrow$ important submatrices’ rows

return G

Dynamic F4 Algorithm

inputs bad basis

choose $>$

repeat

- build "important submatrix"
- perform Gauss-Jordan
- non-zero rows?
    - reconsider $>$

until all important submatrices triangular

$G \leftarrow$ important submatrices' rows

return G

DynGB

Work-in-progress

- "restricted" algorithm
  (once chosen, lm's cannot change)
- portability: {C++11} ∪ {GMP, GLPK, PPL}
  - parallelism via STL `thread` / `async`

Implementation?

DynGB

Work-in-progress

- "restricted" algorithm
  (once chosen, lm's cannot change)
- portability: {C++11} ∪ {GMP, GLPK, PPL}
  - parallelism via STL `thread` / `async`
- works, but not a speed demon
  - slight disadvantage from weighted term ordering
  - Dynamic Buchberger remarkably slow
  - Dynamic F4 OK

Implementation?

DynGB

Work-in-progress

- "restricted" algorithm
  (once chosen, lm's cannot change)
- portability: {C++11} ∪ {GMP, GLPK, PPL}
  - parallelism via STL `thread` / `async`
- works, but not a speed demon
  - slight disadvantage from weighted term ordering
  - Dynamic Buchberger remarkably slow
  - Dynamic F4 OK
- learned a lot of things along the way

DynGB

Work-in-progress

- "restricted" algorithm
  (once chosen, lm's cannot change)
- portability: {C++11} ∪ {GMP, GLPK, PPL}
  - parallelism via STL `thread` / `async`
- works, but not a speed demon
  - slight disadvantage from weighted term ordering
  - Dynamic Buchberger remarkably slow
  - Dynamic F4 OK
- learned a lot of things along the way
- eventual plan is to fold dynamic code into existing CAS
  (e.g., Eder's F4 code)

Reinventing the wheel?

Why not modify a CAS?

Reinventing the wheel?

Why not modify a CAS? It's not easy.

- decades-old code and/or non-existent documentation

Reinventing the wheel?

Why not modify a CAS? It's not easy.

- decades-old code and/or non-existent documentation
- organized, optimized, tuned for static computation
  - 67% penalty for same work (sometimes more)
    [i.e., grevlex v. wgrevlex(1,...,1)]
  - pulling one thread unravels the whole cloth

Reinventing the wheel?

Why not modify a CAS? It's not easy.

"Dive in and destroy; we'll sort it out later."

Reinventing the wheel?

Why not modify a CAS? It's not easy.

"Dive in and destroy; we'll sort it out later."

I spent 2 years trying to modify 2 existing CAS's.
I have not given up, but I needed a reference implementation.

Sample performance

Cyclic-7h

	Buch	DBuch	F4	DF4
time (sec)	17.6	9.5	4.0	1.0
s-polys	2199	396	2199	409
size of basis	443	106	443	108
highest degree	20	14	20	15

Mid-2012 MacBook Pro, 2.5 GHz Intel Core i5,
16 GB RAM, MacOS 10.13 High Sierra

Sample performance

Cyclic-8h

	Buch	DBuch	F4	DF4
time	775	1519	148	28
s-polys	7026	2048	7025	1991
size of basis	1182	415	1182	404
highest degree	30	18	30	17

Mid-2012 MacBook Pro, 2.5 GHz Intel Core i5,
16 GB RAM, MacOS 10.13 High Sierra

Demonstration

This is where I demonstrate the implementation.

Hopefully it works just as well as the last time I tried it.

If it doesn't, I will cry.

Summary for offline readers

Computed homogeneous Cyclic-8 GB

- DynF4 computes a basis of 404 polynomials in 30 sec
- Sage / SINGULAR computes a basis of >1000 polynomials in 60 sec

A larger example

Homogeneous Cyclic-9, $\mathbb{Z}_{43}$

processor	2.5 GHz
basis size	1996
time	2246 sec (37 min)
static overhead	732 sec
reducing	1046 sec
dynamic overhead	450 sec
memory used	5.6 GB

SINGULAR (Buchberger)

processor	2.5 GHz	
time	4h 20m	($\times 7$)
basis size	5601	($\times 2.8$)

Mathic GB (static F4)

processor	3.6 GHz	
time	11m	($\times 2/5$)
memory used	11GB	($\times 2$)
basis size	5602	($\times 2.8$)

Eder's GB (static F4)

processor	3.1 GHz	
time	5m	($\times 1/6$)
memory used	1GB	($\times 1/5$)
basis size	5601	($\times 2.8$)

Implementation:

Cone evolution?

Question #2

What can we say about the tradeoff
of an approximate skeleton v. an exact one?

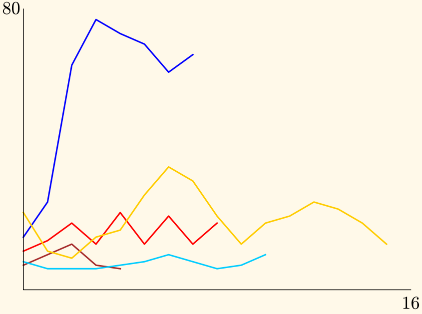
Question #2

What can we say about the tradeoff
of an approximate skeleton v. an exact one?

Similar results in [Hashemi & Talaashrafi, 2016]
(independent work)

Size of skeleton (vectors) v. number of refinements

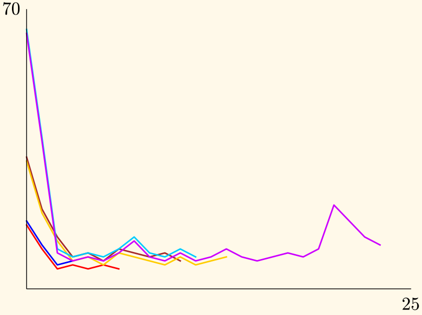
not too frightening:



top-bot (from left): eco8, Es 1, Es 2, kotsireas

Size of skeleton (vectors) v. number of refinements

downright pleasant:

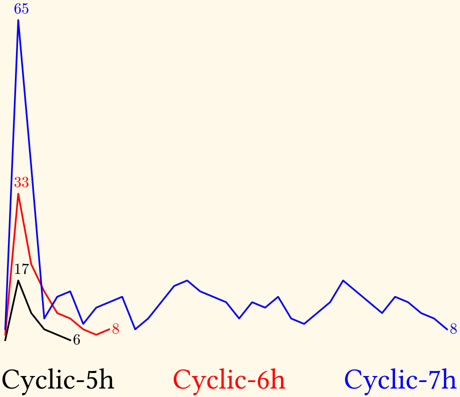


top-bot (from left): Cyc7, Cyc6, Cyc5

Cone evolution

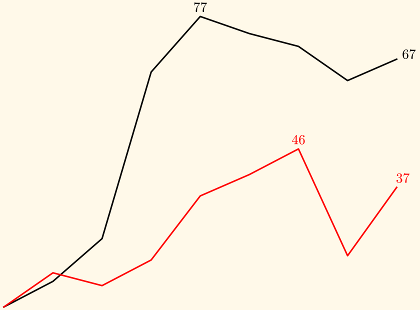
Size of skeleton (vectors) v. number of refinements

also reassuring:



Size of skeleton (vectors) v. number of refinements

not *too* disconcerting:



Cab Es1 (rand binom) Cab Es2 (int prog binom)

Exact or inexact?

Exact solver typically smaller & faster…

Cyc-7h, $\mathbb{Z}_{32003}$	$\lvert basis \rvert$	$\#s$	time (sec)
inexact	401	1373	25.2
exact	107	404	4.99
static	443	2199	10.8

Exact or inexact?

…but not always…

Cyc-6h, $\mathbb{Z}_{32003}$	$\vert basis\vert$	$\#s$	time (sec)
inexact	30	81	0.15
exact	38	110	0.09
static	99	389	0.15

Exact or inexact?

…and in some cases there's no choice!

4×4	$\|basis\|$	$\#s$	time (sec)
inexact	7	6	.418
exact	*	*	*
static	44	202	.396

Not all the news is great

"4×4" system [Yan, 1998]

- ~500 $\longrightarrow$ ~60,000 vectors in 2 refinements
- exact approaches choke / terminate after 2 minutes
- approximate approaches compute GB in half a second!
    - GLPK (simplex)

Implementation:
Hilbert v. Betti

Benchmark systems

Caboara 1

	$\lvert$ basis $\rvert$	$\#s$	time (sec)
static	239	1188	.562
betti	154	711	.281
degree	95	396	.183
hilbert	34	137	.089
random ($\times$3)	63	266	.141
	346	1685	1.63
	138	625	.273

Benchmark systems

Caboara 2

	$\lvert$ basis $\rvert$	$\#s$	time (sec)
static	414	6781	42.96
betti	6	681	.451
degree	6	33	.011
hilbert	7	30	.113
	6	63	.026
random (×3)	6	63	.026
	6	30	.018

Benchmark systems

4×4

	$\lvert$ basis $\rvert$	$\#s$	time (sec)
static	44	202	.404
betti	7	7	.208
degree	7	7	.401
hilbert	7	7	.404
	47	233	8.62
random (×3)	67	379	86.9
	33	144	5.40

*GLPK only

Cyc-7

	$\vert$ basis $\vert$	$\#s$	time (sec)
static	209	2099	10.7
betti	81	2634	43.0
degree	54	2099	50.5
hilbert	54	2147	54.8
random ($\times$3)	35	2227	51.1
	46	2721	43.7
	52	2214	50.4

Benchmark systems

Cyc-7h

	$\lvert$ basis $\rvert$	$\#s$	time (sec)
static	443	2199	14.927
betti	109	404	4.75
degree	106	399	5.05
hilbert	107	404	5.05
	170	842	27.7
random ($\times$3)	155	756	25.8
	214	907	13.0

Benchmark systems

Buch85

	$\lvert$ basis $\rvert$	$\#s$	time (sec)
static	59	412	.039
betti	15	57	.489
degree	11	22	.150
hilbert	11	27	.156
random (×3)	11	20	.227
	11	25	.172
	14	30	.163

Benchmark systems

butcher8

	|basis|	#s	time (sec)
static	29	435	.441
betti	10	386	1.27
degree	*	*	*
hilbert	29	366	.367
	*	*	*
random (×3)	8	260	.519
	8	167	.253

*terminated after several seconds

Benchmark systems

eco8

	$\lvert$basis$\rvert$	#s	time (sec)
static	59	351	.306
betti	21	231	.896
degree	*	*	*
hilbert	12	362	1.54
	8	513	4.70
random (×3)	8	247	.796
	8	382	2.41

*terminated after several seconds

Benchmark systems

kotsireas

	| basis |	#s	time (sec)
static	92	546	1.14
betti	23	405	1.35
degree	6	731	11.0
hilbert	27	311	.809
	*	*	*
random (×3)	6	737	11.1
	*	*	*

Benchmark systems

katsura8

	$\|$basis$\|$	$\#s$	time (sec)
static	143	886	10.7
betti	95	1348	85.1
degree	15	1588	51.8
hilbert	133	822	13.3
	15	1776	72.5
random ($\times 3$)	*	*	*
	30	1266	66.7

Benchmark systems

katsura8h

	$\|$basis$\|$	$\#s$	time (sec)
static	143	886	11.0
betti	154	925	15.7
degree	130	790	15.8
hilbert	249	1745	38.2
random (×3)	183	1191	25.0
	207	1369	26.5
	329	2332	96.3

noon5

	$\|$basis$\|$	$\#s$	time (sec)
static	72	267	.074
betti	72	266	.096
degree	15	577	5.88
hilbert	53	314	.542
	15	577	5.91
random ($\times 3$)	15	578	5.82
	15	577	5.90

Benchmark systems

s9_1

	$\lvert$ basis $\rvert$	$\#s$	time (sec)
static	15	32	.011
betti	14	28	.020
degree	8	14	.017
hilbert	14	28	.021
	8	13	.018
random ($\times$3)	8	13	.020
	8	13	.019

Heuristic performance

Dynamic sometimes longer?

- weighted degree
- longer intermediate polys
- fewer pairs, smaller basis $\not\Rightarrow$ faster computation

Heuristic performance

Dynamic sometimes longer?

- weighted degree
- longer intermediate polys
- fewer pairs, smaller basis $\not\Rightarrow$ faster computation

No heuristic has advantage?!?

Heuristic performance

Dynamic sometimes longer?

- weighted degree
- longer intermediate polys
- fewer pairs, smaller basis $\not\Rightarrow$ faster computation

No heuristic has advantage?!? *Even random choice* competes?

Why is random sometimes better?

- finitely many orderings (equiv. classes, really)

Why is random sometimes better?

- finitely many orderings (equiv. classes, really)
- sometimes "very" finite!

Why is random sometimes better?

- finitely many orderings (equiv. classes, really)
- sometimes "very" finite!

Example (Cyclic-4)

96 possible orderings via gfan

Why is random sometimes better?

- finitely many orderings (equiv. classes, really)
- sometimes "very" finite!

Example (Cyclic-4)

96	possible orderings (via gfan)
4–8	GB size
16	size 4
16	size 5

Why is random sometimes better?

- finitely many orderings (equiv. classes, really)
- sometimes "very" finite!

Example (Cyclic-4)

96	possible orderings (via gfan)
4–8	GB size
16	size 4
16	size 5
33%	probability of choosing "small" basis!

Why is random sometimes better?

- finitely many orderings (equiv. classes, really)
- sometimes "very" finite!

Example (Cyclic-4)

96	possible orderings	via gfan
4–8	GB size	
16	size 4	
16	size 5	
33%	probability of choosing "small" basis!	

For most systems tried, random is *terrible*

Conclusions, future directions

Summary

- Dynamic algorithms seek out "good" orderings
  - can be faster, smaller… but no guarantee
  - metrics need further investigation
- implementation shows promise (IMHO)
- Dynamic F5 (Candice Mitchell)

Thank you!

Massimo Caboara

Università di Pisa

Christian Eder

Николай Юрьевич Золотых

Centre for Computer Algebra

William Stein

Казанский Федеральный Университет

University of Southern Mississippi

NASA Space Grant Consortium

Галина Валеревна Заморий

Bibliography: General

- Bigatti, "Computation of Hilbert-Poincaré Series." *JPAA* 119 (1997), 237–253.
- Buchberger, *An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal.* PhD Thesis (1965), University of Innsbruck. English translation in *JSC* 41 (2006) 475–511.
- Gebauer and Möller, "On an Installation of Buchberger's Algorithm," *JSC* 6 (1988) 275–286.
- Lazard, "Gröbner bases, Gaussian elimination, and resolution of systems of algebraic equations." *EUROCAL '83*, Springer LNCS 162, 146–156.
- Macaulay, "On some formulæ in elimination," *Proceedings of the London Mathematical Society* 33 (1902) 3–27.
- Roune, "A Slice Algorithm for Corners and Hilbert-Poincaré Series of Monomial Ideals." *ISSAC 2010*, AMC Press, 115–122.
- Yan, "The Geobucket Data Structure for Polynomials." *JSC* 25 (1998) 295–293.

Bibliography: Dynamic algorithm

- Caboara, "A Dynamic Algorithm for Gröbner basis computation." *ISSAC '93*, ACM Press, 275–283.
- Caboara and Perry, "Reducing the size and number of linear programs in a dynamic Gröbner basis algorithm." *AAECC* 25 (2014) 99–117.
- O. Golubitsky, "Converging term order sequences and the dynamic Buchberger algorithm." Unpublished preprint received via private communication.
- Gritzmann and Sturmfels, "Minkowski Addition of Polytopes: Computational Complexity and Applications to Gröbner Bases." *SIAM J. Disc. Math* 6 (1993) 246–269.
- Hashemi and Talaashrafi, "A Note on Dynamic Gröbner Bases Computation." *CASC 2016*, Springer, 276–288.
- Langeloh, "Unrestricted dynamic Gröbner Basis algorithms." Master's Thesis, 2019.
- Mora and Robbiano, "The Gröbner fan of an ideal." *JSC* 6 (1988) 183–208.
- Perry, "Exploring the Dynamic Buchberger Algorithm." ISSAC 2017, ACM Press, 365–372.
- Robbiano, "On the theory of graded structures." *JSC* 2 (1986) 139–170.

Bibliography: Faugère

- Faugère, "A new efficient algorithm for computing Gröbner bases (F_4)." *Journal of Pure and Applied Algebra* 139 (1999) 61–88.
- Faugère, "A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5)." *ISSAC '02*, ACM Press, 75–82.
- Faugère and Joux, "Algebraic cryptanalysis of Hidden Field Equation (HFE) cryptosystems using Gröbner bases." *Advances in Cryptology* (2003) 44–60.