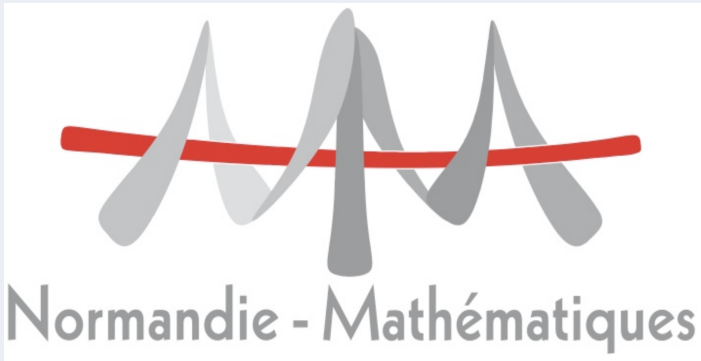
Vincent Neiger
LIP6, Sorbonne Université, France

discovering **algebraic relations** through **polynomial matrix** and **Gröbner basis** computations

Journée scientifique commune
Fédérations NormaSTIC et Normandie-Mathématiques
Caen, 31 March 2026



outline

- first guess, then prove
- algorithms & complexity
- finding univariate relations
- finding multivariate relations

outline

- **first guess, then prove**
  - linearly recurrent sequences
  - differentially finite power series
  - guessing algebraicity or D-finiteness
- **algorithms & complexity**
- **finding univariate relations**
- **finding multivariate relations**

a little game: guess the next term

given a few initial terms $u_0, u_1, u_2, u_3, \ldots$,
- can you guess the next term of the sequence?
- can you guess, more globally, $(u_n)_{n\in\mathbb{N}}$?
- yes, you can, but how and why?

0, 1, 2, 3, 4, 5, **?**

3, 6, 12, 24, 48, 96, **?**

1, 1, 2, 3, 5, 8, **?**

0, 1, 3, 6, 10, 15, **?**

1, 1, 2, 4, 10, 26, **?**

a little game: guess the next term

given a few initial terms $u_0, u_1, u_2, u_3, \ldots$,
- can you guess the next term of the sequence?
- can you guess, more globally, $(u_n)_{n\in\mathbb{N}}$?
- yes, you can, but how and why?

$0, 1, 2, 3, 4, 5,$ **6**

$u_n = n$, integers

$3, 6, 12, 24, 48, 96,$ **?**

$1, 1, 2, 3, 5, 8,$ **?**

$0, 1, 3, 6, 10, 15,$ **?**

$1, 1, 2, 4, 10, 26,$ **?**

a little game: guess the next term

given a few initial terms $u_0, u_1, u_2, u_3, \ldots$,
- can you guess the next term of the sequence?
- can you guess, more globally, $(u_n)_{n\in\mathbb{N}}$?
- yes, you can, but how and why?

0, 1, 2, 3, 4, 5, **6** $\qquad$ $u_n = n$, integers

3, 6, 12, 24, 48, 96, **192** $\qquad$ $u_n = 3 \cdot 2^n$, geometric

1, 1, 2, 3, 5, 8, **?**

0, 1, 3, 6, 10, 15, **?**

1, 1, 2, 4, 10, 26, **?**

a little game: guess the next term

given a few initial terms $u_0, u_1, u_2, u_3, \ldots$,
- can you guess the next term of the sequence?
- can you guess, more globally, $(u_n)_{n\in\mathbb{N}}$?
- yes, you can, but how and why?

0, 1, 2, 3, 4, 5, **6**

3, 6, 12, 24, 48, 96, **192**

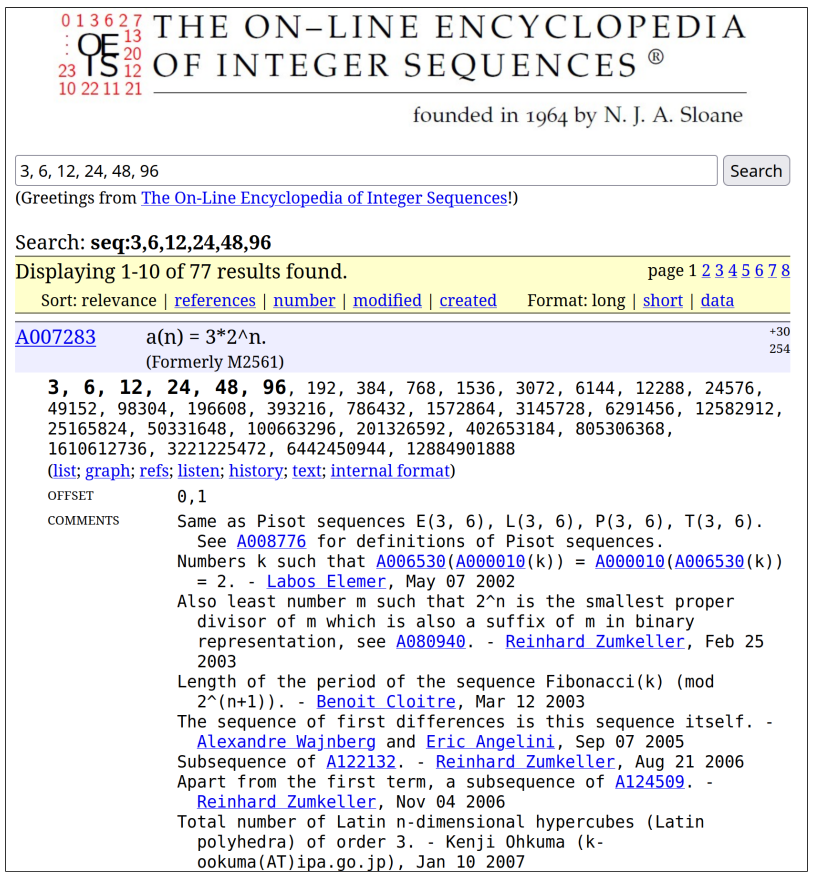
1, 1, 2, 3, 5, 8, **?**

0, 1, 3, 6, 10, 15, **?**

1, 1, 2, 4, 10, 26, **?**

$u_n = n$, integers

$u_n = 3 \cdot 2^n$, geometric



THE ON–LINE ENCYCLOPEDIA OF INTEGER SEQUENCES®

founded in 1964 by N. J. A. Sloane

3, 6, 12, 24, 48, 96 Search

(Greetings from The On-Line Encyclopedia of Integer Sequences!)

Search: **seq:3,6,12,24,48,96**

Displaying 1-10 of 77 results found. page 1 2 3 4 5 6 7 8

Sort: relevance | references | number | modified | created Format: long | short | data

A007283 a(n) = 3*2^n. +30 254

(Formerly M2561)

3, 6, 12, 24, 48, 96, 192, 384, 768, 1536, 3072, 6144, 12288, 24576, 49152, 98304, 196608, 393216, 786432, 1572864, 3145728, 6291456, 12582912, 25165824, 50331648, 100663296, 201326592, 402653184, 805306368, 1610612736, 3221225472, 6442450944, 12884901888

(list; graph; refs; listen; history; text; internal format)

OFFSET 0,1

COMMENTS Same as Pisot sequences E(3, 6), L(3, 6), P(3, 6), T(3, 6). See A008776 for definitions of Pisot sequences.
Numbers k such that A006530(A000010(k)) = A000010(A006530(k)) = 2. - Labos Elemer, May 07 2002
Also least number m such that 2^n is the smallest proper divisor of m which is also a suffix of m in binary representation, see A080940. - Reinhard Zumkeller, Feb 25 2003
Length of the period of the sequence Fibonacci(k) (mod 2^(n+1)). - Benoit Cloitre, Mar 12 2003
The sequence of first differences is this sequence itself. - Alexandre Wajnberg and Eric Angelini, Sep 07 2005
Subsequence of A122132. - Reinhard Zumkeller, Aug 21 2006
Apart from the first term, a subsequence of A124509. - Reinhard Zumkeller, Nov 04 2006
Total number of Latin n-dimensional hypercubes (Latin polyhedra) of order 3. - Kenji Ohkuma (k-ookuma(AT)ipa.go.jp), Jan 10 2007

a little game: guess the next term

given a few initial terms $u_0, u_1, u_2, u_3, \ldots$,
- can you guess the next term of the sequence?
- can you guess, more globally, $(u_n)_{n\in\mathbb{N}}$?
- yes, you can, but how and why?

0, 1, 2, 3, 4, 5, **6**

$u_n = n$, integers

3, 6, 12, 24, 48, 96, **192**

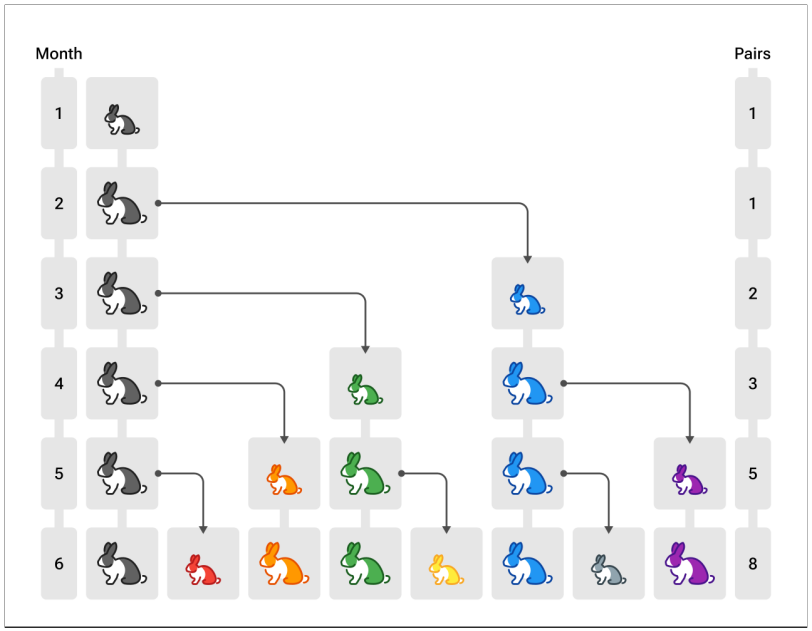
$u_n = 3 \cdot 2^n$, geometric

1, 1, 2, 3, 5, 8, **13**

$u_n = \frac{\varphi^n - \psi^n}{\sqrt{5}}$, Fibonacci

0, 1, 3, 6, 10, 15, **?**

1, 1, 2, 4, 10, 26, **?**



a little game: guess the next term

given a few initial terms $u_0, u_1, u_2, u_3, \ldots$,
- can you guess the next term of the sequence?
- can you guess, more globally, $(u_n)_{n\in\mathbb{N}}$?
- yes, you can, but how and why?

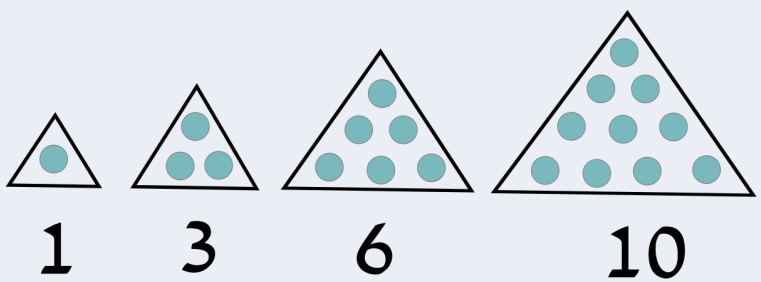
0, 1, 2, 3, 4, 5, **6** — $u_n = n$, integers

3, 6, 12, 24, 48, 96, **192** — $u_n = 3 \cdot 2^n$, geometric

1, 1, 2, 3, 5, 8, **13** — $u_n = \frac{\varphi^n - \psi^n}{\sqrt{5}}$, Fibonacci

0, 1, 3, 6, 10, 15, **21** — $u_n = \frac{n(n+1)}{2}$, triangular

1, 1, 2, 4, 10, 26, **?**



a little game: guess the next term

given a few initial terms $u_0, u_1, u_2, u_3, \ldots$,
- can you guess the next term of the sequence?
- can you guess, more globally, $(u_n)_{n\in\mathbb{N}}$?
- yes, you can, but how and why?

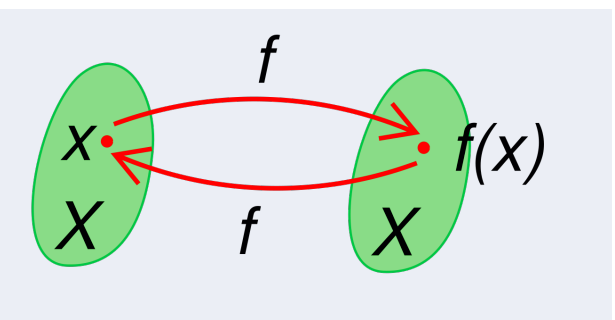
0, 1, 2, 3, 4, 5, **6** — $u_n = n$, integers

3, 6, 12, 24, 48, 96, **192** — $u_n = 3 \cdot 2^n$, geometric

1, 1, 2, 3, 5, 8, **13** — $u_n = \frac{\varphi^n - \psi^n}{\sqrt{5}}$, Fibonacci

0, 1, 3, 6, 10, 15, **21** — $u_n = \frac{n(n+1)}{2}$, triangular

1, 1, 2, 4, 10, 26, **76** — involutions, $u_n = \sum_{k=0}^{\lfloor n/2 \rfloor} \frac{n!}{2^k k!(n-2k)!}$



a little game: guess the next term

given a few initial terms $u_0, u_1, u_2, u_3, \ldots$,
- can you guess the next term of the sequence?
- can you guess, more globally, $(u_n)_{n\in\mathbb{N}}$?
- yes, you can, but how and why?

0, 1, 2, 3, 4, 5, **6**	$u_{n+1} = u_n + 1$	$u_n = n$, integers
3, 6, 12, 24, 48, 96, **192**	$u_{n+1} = 2u_n$	$u_n = 3 \cdot 2^n$, geometric
1, 1, 2, 3, 5, 8, **13**	$u_{n+2} = u_{n+1} + u_n$	$u_n = \frac{\varphi^n - \psi^n}{\sqrt{5}}$, Fibonacci
0, 1, 3, 6, 10, 15, **21**	$u_{n+1} = u_n + n + 1$ $u_{n+2} = 2u_{n+1} - u_n + 1$	$u_n = \frac{n(n+1)}{2}$, triangular
1, 1, 2, 4, 10, 26, **76**	$u_{n+2} = u_{n+1} + (n+1)u_n$	involutions, $u_n = \sum_{k=0}^{\lfloor n/2 \rfloor} \frac{n!}{2^k k!(n-2k)!}$

these sequences are constant-recursive or polynomial-recursive

↑ with constant coefficients ↑

linearly recurrent sequences

↓ with polynomial coefficients ↓

$\mathbb{K}$ is a field characteristic zero: $\mathbb{K} = \mathbb{Q}, \mathbb{C}, \overline{\mathbb{Q}}, \ldots$

a sequence $(u_n) \in \mathbb{K}^{\mathbb{N}}$ is **C-recursive** of order m if

$$p_0 u_n + \cdots + p_{m-1} u_{n+m-1} + u_{n+m} = 0 \text{ for all } n$$

for some coefficients $p_0, \ldots, p_{m-1} \in \mathbb{K}$ not all zero

each term is deduced from the m previous terms

- geometric $u_{n+1} = q u_n$
- integers $u_{n+2} = 2u_{n+1} - u_n$
- Fibonacci $u_{n+2} = u_{n+1} + u_n$
- triangular numbers, etc.

↑ with constant coefficients ↑

linearly recurrent sequences

↓ with polynomial coefficients ↓

$\mathbb{K}$ is a field characteristic zero: $\mathbb{K} = \mathbb{Q}, \mathbb{C}, \overline{\mathbb{Q}}, \ldots$

a sequence $(u_n) \in \mathbb{K}^{\mathbb{N}}$ is **C-recursive** of order m if

$$p_0 u_n + \cdots + p_{m-1} u_{n+m-1} + u_{n+m} = 0 \text{ for all } n$$

for some coefficients $p_0, \ldots, p_{m-1} \in \mathbb{K}$ not all zero

each term is deduced from the m previous terms

- geometric $u_{n+1} = q u_n$
- integers $u_{n+2} = 2u_{n+1} - u_n$
- Fibonacci $u_{n+2} = u_{n+1} + u_n$
- triangular numbers, etc.

generating series is **rational**:

$$\sum_{n \in \mathbb{N}} u_n x^n = \frac{q(x)}{p(x)}$$

↑ with constant coefficients ↑

linearly recurrent sequences

↓ with polynomial coefficients ↓

$\mathbb{K}$ is a field characteristic zero: $\mathbb{K} = \mathbb{Q}, \mathbb{C}, \overline{\mathbb{Q}}, \ldots$

a sequence $(u_n) \in \mathbb{K}^{\mathbb{N}}$ is **C-recursive** of order m if

$$p_0 u_n + \cdots + p_{m-1} u_{n+m-1} + u_{n+m} = 0 \text{ for all } n$$

for some coefficients $p_0, \ldots, p_{m-1} \in \mathbb{K}$ not all zero

each term is deduced from the m previous terms

- geometric $u_{n+1} = q u_n$
- integers $u_{n+2} = 2u_{n+1} - u_n$
- Fibonacci $u_{n+2} = u_{n+1} + u_n$
- triangular numbers, etc.

generating series is **rational**:

$$\sum_{n \in \mathbb{N}} u_n x^n = \frac{q(x)}{p(x)}$$

↑ with constant coefficients ↑

linearly recurrent sequences

↓ with polynomial coefficients ↓

a sequence $(u_n) \in \mathbb{K}^{\mathbb{N}}$ is **P-recursive** of order m if

$$p_0(n) u_n + p_1(n) u_{n+1} + \cdots + p_m(n) u_{n+m} = 0 \text{ for all } n$$

for some coefficients $p_0, \ldots, p_m \in \mathbb{K}[n]$ not all zero

- C-recursive $\Rightarrow$ P-recursive
- involutions $u_{n+2} = u_{n+1} + (n+1) u_n$
- hypergeometric $u_{n+1} = \frac{p_0(n)}{p_1(n)} u_n$
- harmonic numbers $u_n = \sum_{k=1}^{n} \frac{1}{k}$

$\mathbb{K}$ is a field characteristic zero: $\mathbb{K} = \mathbb{Q}, \mathbb{C}, \overline{\mathbb{Q}}, \ldots$

a sequence $(u_n) \in \mathbb{K}^{\mathbb{N}}$ is **C-recursive** of order m if

$$p_0 u_n + \cdots + p_{m-1} u_{n+m-1} + u_{n+m} = 0 \text{ for all } n$$

for some coefficients $p_0, \ldots, p_{m-1} \in \mathbb{K}$ not all zero

each term is deduced from the m previous terms

- geometric $u_{n+1} = q u_n$
- integers $u_{n+2} = 2u_{n+1} - u_n$
- Fibonacci $u_{n+2} = u_{n+1} + u_n$
- triangular numbers, etc.

generating series is **rational**:

$$\sum_{n \in \mathbb{N}} u_n x^n = \frac{q(x)}{p(x)}$$

↑ with constant coefficients ↑

linearly recurrent sequences

↓ with polynomial coefficients ↓

generating series

$$f(x) = \sum_{n \in \mathbb{N}} u_n x^n$$

is **D-finite (!?)**

a sequence $(u_n) \in \mathbb{K}^{\mathbb{N}}$ is **P-recursive** of order m if

$$p_0(n) u_n + p_1(n) u_{n+1} + \cdots + p_m(n) u_{n+m} = 0 \text{ for all } n$$

for some coefficients $p_0, \ldots, p_m \in \mathbb{K}[n]$ not all zero

- C-recursive $\Rightarrow$ P-recursive
- involutions $u_{n+2} = u_{n+1} + (n+1)u_n$
- hypergeometric $u_{n+1} = \frac{p_0(n)}{p_1(n)} u_n$
- harmonic numbers $u_n = \sum_{k=1}^{n} \frac{1}{k}$

power series: algebraicity and D-finiteness

recall: (u_n) is P-recursive $\Leftrightarrow$ $f(x)$ is D-finite

a power series $f(x) = \sum_{n\in\mathbb{N}} u_n x^n$

- is **D-finite** if $p_0 f + p_1 f' + p_2 f'' + \cdots + p_m f^{(m)} = 0$
- is **algebraic** if $p_0 + p_1 f + p_2 f^2 + \cdots + p_m f^m = 0$

for some polynomials $p_0(x), \ldots, p_m(x) \in \mathbb{K}[x]$

· algebraic $\Rightarrow$ D-finite [Abel 1827]
· D-finite: $\sqrt{1-x} + \sqrt{1+x}$, $\exp(x)$, $\sin(x)$, $\cos(x)$, $\ln(1-x)$
· not D-finite: $\tan(x)$ (note: nonlinear equation $\tan' = 1 + \tan^2$)

power series: algebraicity and D-finiteness

recall: (u_n) is P-recursive $\Leftrightarrow$ $f(x)$ is D-finite

a power series $f(x) = \sum_{n\in\mathbb{N}} u_n x^n$

- is **D-finite** if $p_0 f + p_1 f' + p_2 f'' + \cdots + p_m f^{(m)} = 0$
- is **algebraic** if $p_0 + p_1 f + p_2 f^2 + \cdots + p_m f^m = 0$

for some polynomials $p_0(x), \ldots, p_m(x) \in \mathbb{K}[x]$

· algebraic $\Rightarrow$ D-finite [Abel 1827]
· D-finite: $\sqrt{1-x} + \sqrt{1+x}$, $\exp(x)$, $\sin(x)$, $\cos(x)$, $\ln(1-x)$
· not D-finite: $\tan(x)$ (note: nonlinear equation $\tan' = 1 + \tan^2$)

P-recursive sequences / D-finite series are ubiquitous:

- 60% of the *Handbook of Mathematical Functions* [Abramowitz-Stegun 1964]
- enumerative combinatorics (e.g. counting walks)
- algebraic number theory (e.g. proving algebraicity or transcendence)
- computer algebra (e.g. getting faster algorithms)

software support is available:

Maple `gfun` — Mathematica `HolonomicFunctions` — SageMath `ore_algebra`

“first guess, then prove”

[Pólya 1954] mathematics presented in a finished form consists of proofs; however, mathematics in the making consists of guesses

get data $\rightarrow$ **make conjectures** $\rightarrow$ **find proof**

automatic guessing of algebraic relations

"first guess, then prove"

[Pólya 1954] mathematics presented in a finished form consists of proofs; however, mathematics in the making consists of guesses

get data → **make conjectures** → **find proof**

automatic guessing of algebraic relations

A simple but powerful technique which has become an important tool in experimental mathematics takes as input the first few terms of an infinite sequence and returns as output a plausible hypothesis for a recurrence equation that the sequence may satisfy, or a plausible hypothesis for a differential equation satisfied by its generating function. The principle is known as automated guessing as it somehow makes a guess how the infinite sequence continues beyond the finitely many terms supplied as input. In certain situations where sufficient additional information is available about the sequence at hand, automated guessing can be combined with other techniques from computer algebra that confirm that the guessed equation is correct. One of many successful applications of this paradigm is the proof of the qTSPP conjecture [21].

[Kauers-Koutschan, 2022]

"first guess, then prove"

[Pólya 1954] mathematics presented in a finished form consists of proofs; however, mathematics in the making consists of guesses

get data → **make conjectures** → **find proof**

automatic guessing of algebraic relations

A simple but powerful technique which has become an important tool in experimental mathematics takes as input the first few terms *[in]* terms $u_0, u_1, \ldots, u_{d-1} \in \mathbb{K}$ output a plausible hypothesis for a recurrence equation that the sequence may satisfy, or a plausible hypothesis for a differential equation satisfied by its generating function. The principle is known as automated guessing as it some- *[out]* polynomial coefficients $p_0, p_1, \ldots, p_m$ of
- recurrence equation for $(u_n)_n$
- algebraic/differential equation for $\sum_{n\in\mathbb{N}} u_n x^n$

s beyond the tions where sequence at hand, automated guessing can be combined with other techniques from computer algebra that confirm that the guessed equation is correct. One of many successful applications of this paradigm is the proof of the qTSPP conjecture [21].

[Kauers-Koutschan, 2022]

"first guess, then prove"

[Pólya 1954] mathematics presented in a finished form consists of proofs; however, mathematics in the making consists of guesses

get data → **make conjectures** → **find proof**

automatic guessing of algebraic relations

A simple but powerful technique which has become an important tool in experimental mathematics takes as input the first few terms *[in]* terms $u_0, u_1, \ldots, u_{d-1} \in \mathbb{K}$ output a plausible hypothesis for a recurrence equation that the sequence may satisfy, or a plausible hypothesis for a differential equation satisfied by its generating function. The principle is known as automated guessing as it some- *[out]* polynomial coefficients $p_0, p_1, \ldots, p_m$ of
· recurrence equation for $(u_n)_n$
· algebraic/differential equation for $\sum_{n\in\mathbb{N}} u_n x^n$

s beyond the tions where sequence at hand, automated guessing can be combined with other techniques from computer algebra that confirm that the guessed equation is correct. One of many successful applications of this paradigm is the proof of the qTSPP conjecture [21].

[Kauers-Koutschan, 2022]

core algorithmic tool:
Hermite-Padé approximants
[Hermite 1893] [Padé 1894]

using the guessed relation:
- unroll next terms
- good data structure
- automatic proof of identities
  e.g. $\cos^2 + \sin^2 = 1$

"first guess, then prove"

[Pólya 1954] mathematics presented in a finished form consists of proofs; however, mathematics in the making consists of guesses

get data $\rightarrow$ **make conjectures** $\rightarrow$ **find proof**

automatic guessing of algebraic relations

"although it rarely happens in practice, a guessed equation may be incorrect"

$u_n = \lfloor (n+1)\tanh(\pi) \rfloor \quad \rightarrow \quad$ is $f(x) = \sum_n u_n x^n$ rational? algebraic? D-finite?

correct. One of many successful applications of this paradigm is the proof of the qTSPP conjecture [21]. proof?

[Kauers-Koutschan, 2022]

"first guess, then prove"

[Pólya 1954] mathematics presented in a finished form consists of proofs; however, mathematics in the making consists of guesses

get data → **make conjectures** → **find proof**

automatic guessing of algebraic relations

"although it rarely happens in practice, a guessed equation may be incorrect"

$u_n = \lfloor (n+1)\tanh(\pi) \rfloor$ → is $f(x) = \sum_n u_n x^n$ rational? algebraic? D-finite?

$d = 10$

```
sage: d = 10
....: seq = [floor((n+1) * tanh(pi)) for n in range(d)]
....: Series.<x> = PowerSeriesRing(QQ)
....: f = Series(seq) + O(x**d)
....: f.pade(d/2, d/2-1)
x/(x^2 - 2*x + 1)
```

looks **rational**!

correct. One of many successful applications of this paradigm is the proof of the qTSPP conjecture [21]. proof?

[Kauers-Koutschan, 2022]

"first guess, then prove"

[Pólya 1954] mathematics presented in a finished form consists of proofs; however, mathematics in the making consists of guesses

get data → **make conjectures** → **find proof**

automatic guessing of algebraic relations

"although it rarely happens in practice, a guessed equation may be incorrect"

$u_n = \lfloor (n+1)\tanh(\pi) \rfloor$ → is $f(x) = \sum_n u_n x^n$ rational? algebraic? D-finite?

$d = 100$

```
sage: d = 100
....: seq = [floor((n+1) * tanh(pi)) for n in range(d)]
....: Series.<x> = PowerSeriesRing(QQ)
....: f = Series(seq) + O(x**d)
....: f.pade(d/2, d/2-1)
x/(x^2 - 2*x + 1)
```

looks **rational**!

correct. One of many successful applications of this paradigm is the proof of the qTSPP conjecture [21].

proof?

[Kauers-Koutschan, 2022]

"first guess, then prove"

[Pólya 1954] mathematics presented in a finished form consists of proofs; however, mathematics in the making consists of guesses

get data → **make conjectures** → **find proof**

automatic guessing of algebraic relations

"although it rarely happens in practice, a guessed equation may be incorrect"

$u_n = \lfloor (n+1)\tanh(\pi) \rfloor$ → is $f(x) = \sum_n u_n x^n$ rational? algebraic? D-finite?

$d = 500$

```
sage: d = 500
....: seq = [floor((n+1) * tanh(pi)) for n in range(d)]
....: Series.<x> = PowerSeriesRing(QQ)
....: f = Series(seq) + O(x**d)
....: f.pade(d/2, d/2-1)
x/(x^2 - 2*x + 1)
```

looks **rational**!

correct. One of many successful applications of this paradigm is the proof of the qTSPP conjecture [21]. proof?

[Kauers-Koutschan, 2022]

"first guess, then prove"

[Pólya 1954] mathematics presented in a finished form consists of proofs; however, mathematics in the making consists of guesses

get data $\rightarrow$ **make conjectures** $\rightarrow$ **find proof**

automatic guessing of algebraic relations

"although it rarely happens in practice, a guessed equation may be incorrect"

$u_n = \lfloor (n+1)\tanh(\pi) \rfloor \quad \rightarrow \quad$ is $f(x) = \sum_n u_n x^n$ rational? algebraic? D-finite?

$d = 600$

```
sage: d = 600
....: seq = [floor((n+1) * tanh(pi)) for n in range(d)]
....: Series.<x> = PowerSeriesRing(QQ)
....: f = Series(seq) + O(x**d)
....: f.pade(d/2, d/2-1)
(x^267 + x^266 + x^265 + x^264 + x^263 + x^262 + x^261 + x^260 + x^259 + x^258 + x^257 + x^256
+ x^255 + x^254 + x^253 + x^252 + x^251 + x^250 + x^249 + x^248 + x^247 + x^246 + x^245 + x^244
 + x^243 + x^242 + x^241 + x^240 + x^239 + x^238 + x^237 + x^236 + x^235 + x^234 + x^233 + x^23
2 + x^231 + x^230 + x^229 + x^228 + x^227 + x^226 + x^225 + x^224 + x^223 + x^222 + x^221 + x^2
20 + x^219 + x^218 + x^217 + x^216 + x^215 + x^214 + x^213 + x^212 + x^211 + x^210 + x^209 + x^
208 + x^207 + x^206 + x^205 + x^204 + x^203 + x^202 + x^201 + x^200 + x^199 + x^198 + x^197 + x
^196 + x^195 + x^194 + x^193 + x^192 + x^191 + x^190 + x^189 + x^188 + x^187 + x^186 + x^185 +
x^184 + x^183 + x^182 + x^181 + x^180 + x^179 + x^178 + x^177 + x^176 + x^175 + x^174 + x^173 +
 x^172 + x^171 + x^170 + x^169 + x^168 + x^167 + x^166 + x^165 + x^164 + x^163 + x^162 + x^161
+ x^160 + x^159 + x^158 + x^157 + x^156 + x^155 + x^154 + x^153 + x^152 + x^151 + x^150 + x^149
 + x^148 + x^147 + x^146 + x^145 + x^144 + x^143 + x^142 + x^141 + x^140 + x^139 + x^138 + x^13
7 + x^136 + x^135 + x^134 + x^133 + x^132 + x^131 + x^130 + x^129 + x^128 + x^127 + x^126 + x^1
25 + x^124 + x^123 + x^122 + x^121 + x^120 + x^119 + x^118 + x^117 + x^116 + x^115 + x^114 + x^
113 + x^112 + x^111 + x^110 + x^109 + x^108 + x^107 + x^106 + x^105 + x^104 + x^103 + x^102 + x
^101 + x^100 + x^99 + x^98 + x^97 + x^96 + x^95 + x^94 + x^93 + x^92 + x^91 + x^90 + x^89 + x^8
```

oops. . .

correct. One
the proof of t

[Kauers-Kou

"first guess, then prove"

[Pólya 1954] mathematics presented in a finished form consists of proofs; however, mathematics in the making consists of guesses

get data $\rightarrow$ **make conjectures** $\rightarrow$ **find proof**

automatic guessing of algebraic relations

need for **fast algorithms** and **efficient implementations**

[Bostan-Kauers '10] "the generating function for Gessel walks is algebraic"

surprising; we had no reason to suspect that $G(t;x,y)$ *is algebraic, as even the specialization* $G(t;0,0)$ *was widely believed to be transcendental*

the minimal polynomial of $G(t;x,y)$ *is huge, having a total size of* $\sim$*30Gb*

expressions far too large to be processed efficiently even by standard computer algebra systems such as Maple or Mathematica

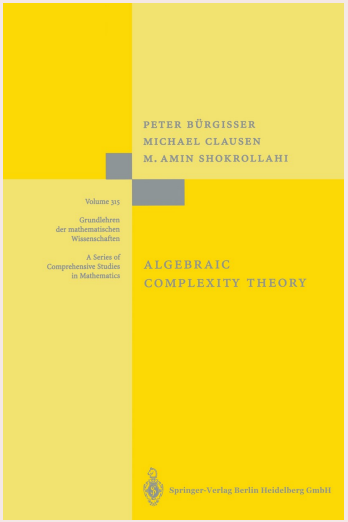
we needed careful implementations of sophisticated algorithms

outline

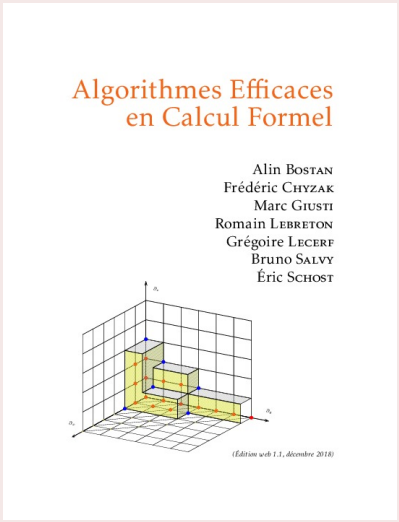
- **first guess, then prove**
  - linearly recurrent sequences
  - differentially finite power series
  - guessing algebraicity or D-finiteness
- **algorithms & complexity**
- **finding univariate relations**
- **finding multivariate relations**

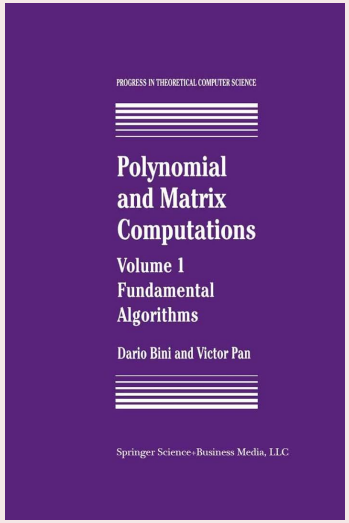
outline

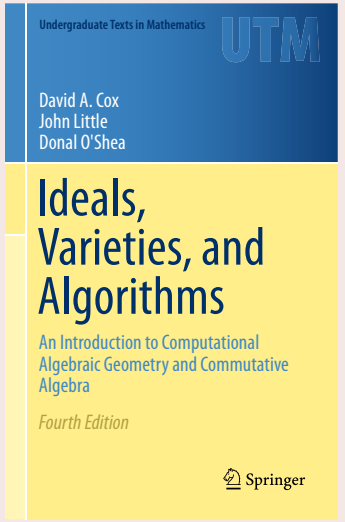
- **first guess, then prove**
  - linearly recurrent sequences
  - differentially finite power series
  - guessing algebraicity or D-finiteness
- **algorithms & complexity**
  - algebraic computations
  - asymptotic complexity bounds
  - software performance
- **finding univariate relations**
- **finding multivariate relations**

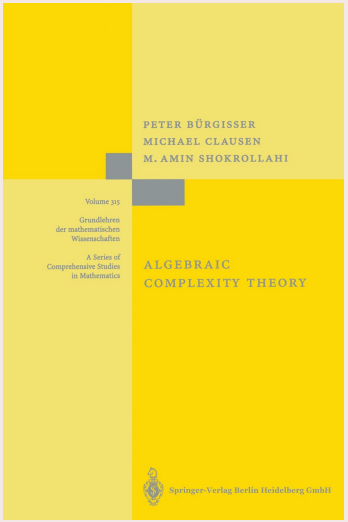

PETER BÜRGISSER
MICHAEL CLAUSEN
M. AMIN SHOKROLLAHI
Volume 315
Grundlehren der mathematischen Wissenschaften
A Series of Comprehensive Studies in Mathematics
ALGEBRAIC COMPLEXITY THEORY
Springer-Verlag Berlin Heidelberg GmbH

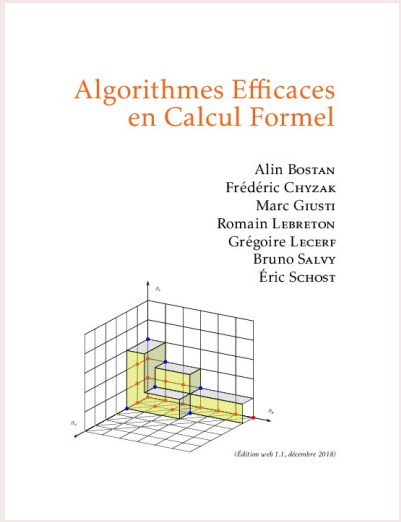

Modern Computer Algebra
Third Edition
Joachim von zur Gathen and Jürgen Gerhard
CAMBRIDGE
CAMBRIDGE
more information – www.cambridge.org/9781107039032

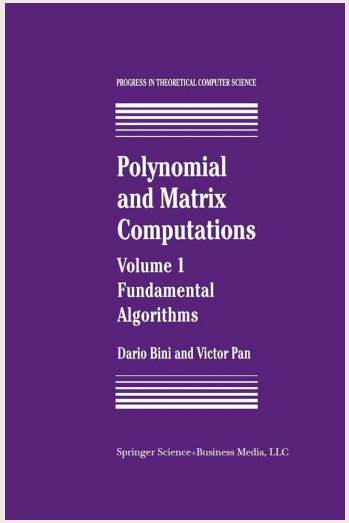

Algorithmes Efficaces en Calcul Formel
Alin Bostan
Frédéric Chyzak
Marc Giusti
Romain Lebreton
Grégoire Lecerf
Bruno Salvy
Éric Schost
(Édition web 1.1, décembre 2018)

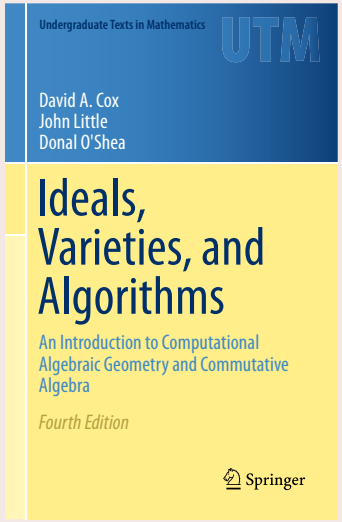

PROGRESS IN THEORETICAL COMPUTER SCIENCE
Polynomial and Matrix Computations
Volume 1
Fundamental Algorithms
Dario Bini and Victor Pan
Springer Science+Business Media, LLC


Undergraduate Texts in Mathematics
UTM
David A. Cox
John Little
Donal O'Shea
Ideals, Varieties, and Algorithms
An Introduction to Computational Algebraic Geometry and Commutative Algebra
Fourth Edition
Springer









MAGMA
COMPUTER • ALGEBRA

MATLAB®

PARI GP

Maple™

WOLFRAM
MATHEMATICA 12

sage

LinBox
Exact Linear Algebra

GIVARO
Finite Fields

FFLAS-FFPACK
Matrix kernels

GMP
Arbitrary precision
Integers

BLAS
Linear Algebra
Subroutines

SymPy

CoCoA

Euclid's GCD -300

Gaussian elimination 179

Newton's method 1669

computer algebra

design of **fast algorithms**
and **software implementations**
for **exact computations**
with **mathematical objects**

LLL '82, NFS '88

FFT 1805, '65

Buchberger '76

Strassen '69

Karatsuba '62

"fast": measuring efficiency

efficient algorithms for polynomials, matrices, power series, ...
with coefficients in some base field $\mathbb{K}$

- low **complexity bound**
- low **execution time**

..., low memory usage, low power consumption, ...

- field $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$ for p prime
- rational numbers $\mathbb{K} = \mathbb{Q}$

- finite extensions $\mathbb{K}[x]/\langle f(x)\rangle$
- rational fractions $\mathbb{K}(x, y, z)$

"fast": measuring efficiency

efficient algorithms for polynomials, matrices, power series, ...
with coefficients in some base field $\mathbb{K}$

- low **complexity bound**
- low **execution time**

..., low memory usage, low power consumption, ...

- field $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$ for p prime
- rational numbers $\mathbb{K} = \mathbb{Q}$

- finite extensions $\mathbb{K}[x]/\langle f(x)\rangle$
- rational fractions $\mathbb{K}(x, y, z)$

algebraic complexity bounds
$\rightsquigarrow$ count basic operations in $\mathbb{K}$

- operations $+, -, \times, \div, =$
- notation $O(\cdot)$ and $\tilde{O}(\cdot)$
- this talk: no parallelism, single thread

"fast": measuring efficiency

efficient algorithms for polynomials, matrices, power series, ...
with coefficients in some base field $\mathbb{K}$

- low **complexity bound**
- low **execution time**

..., low memory usage, low power consumption, ...

- field $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$ for p prime
- rational numbers $\mathbb{K} = \mathbb{Q}$

- finite extensions $\mathbb{K}[x]/\langle f(x)\rangle$
- rational fractions $\mathbb{K}(x, y, z)$

algebraic complexity bounds
$\rightsquigarrow$ count basic operations in $\mathbb{K}$

- operations $+, -, \times, \div, =$
- notation $O(\cdot)$ and $\tilde{O}(\cdot)$
- this talk: no parallelism, single thread

for all runtimes in this talk:

- $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$ with prime $p \approx 2^{60} \approx 10^{18}$
- this laptop with CPU AMD zen4

practical performance
$\rightsquigarrow$ measure software runtime

"fast": measuring efficiency

efficient algorithms for polynomials, matrices, power series, ...
with coefficients in some base field $\mathbb{K}$

- low **complexity bound**
- low **execution time**

..., low memory usage, low power consumption, ...

- field $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$ for p prime
- rational numbers $\mathbb{K} = \mathbb{Q}$

- finite extensions $\mathbb{K}[x]/\langle f(x)\rangle$
- rational fractions $\mathbb{K}(x, y, z)$

algebraic complexity bounds
$\rightsquigarrow$ count basic operations in $\mathbb{K}$

- operations $+, -, \times, \div, =$
- notation $O(\cdot)$ and $\tilde{O}(\cdot)$
- this talk: no parallelism, single thread

related?

for all runtimes in this talk:

- $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$ with prime $p \approx 2^{60} \approx 10^{18}$
- this laptop with CPU AMD zen4

practical performance
$\rightsquigarrow$ measure software runtime

univariate polynomials: multiplication

$f = 87x^7 + 74x^6 + 60x^5 + 46x^4 + 16x^3 + 41x^2 + 86x + 69$

$f \in \mathbb{K}[x]_{<8} \quad \longrightarrow$ univariate polynomial in x of degree < 8 over $\mathbb{K}$

fundamental operations on polynomials of degree $< d$:

- addition and Horner's evaluation are linear: $O(d)$
- naive multiplication is quadratic: $O(d^2)$

[Karatsuba'62] $\quad M(d) \in O(d^{1.58})$

breakthrough: **subquadratic** polynomial multiplication

$$f \cdot g = x^d \cdot f_1 \cdot g_1 + x^{d/2} \cdot ((f_0 + f_1) \cdot (g_0 + g_1) - f_1 \cdot g_1 - f_0 \cdot g_0) + f_0 \cdot g_0$$

univariate polynomials: multiplication

$f = 87x^7 + 74x^6 + 60x^5 + 46x^4 + 16x^3 + 41x^2 + 86x + 69$

$f \in \mathbb{K}[x]_{<8} \longrightarrow$ univariate polynomial in x of degree < 8 over $\mathbb{K}$

fundamental operations on polynomials of degree $< d$:

- addition and Horner's evaluation are linear: $O(d)$
- naive multiplication is quadratic: $O(d^2)$

[Karatsuba'62] $M(d) \in O(d^{1.58})$

breakthrough: **subquadratic** polynomial multiplication

[Schönhage-Strassen'71] [Nussbaumer'80] [Cantor-Kaltofen'91] $M(d) \in O(d \log(d) \log\log(d))$

breakthrough: **quasi-linear** polynomial multiplication

research still active, with recent progress by [Harvey-van der Hoeven-Lecerf]

- change of representation by evaluation-interpolation
- FFT techniques using (virtual) roots of unity
- used in practice as soon as $d \approx 100$ $(\mathbb{K} = \mathbb{Z}/p\mathbb{Z})$

note: $M(d) \sim 9d \log_2(d)$
if provided a "good" root of unity

matrices: multiplication

$$\mathbf{A} = \begin{bmatrix} 28 & 68 & 75 & 70 \\ 38 & 25 & 75 & 55 \\ 24 & 1 & 56 & 28 \end{bmatrix} \in \mathbb{K}^{3\times 4} \longrightarrow 3 \times 4 \text{ matrix over } \mathbb{K}$$

fundamental operations on $m \times m$ matrices:

- addition is "quadratic": $O(m^2)$ operations in $\mathbb{K}$
- naive multiplication is cubic: $O(m^3)$

[Strassen'69]

breakthrough: **subcubic** matrix multiplication

- complexity exponent $\omega \approx 2.81$
- used in practice for $m \geqslant$ a few 100s

i.e. $O(m^\omega)$ complexity

- best-known exponent $\omega \approx 2.372$
  [Alman-Duan-Vassilevska Williams-Xu-Xu-Zhou'25]
- "galactic" algorithms: strongly impractical as such

Numer. Math. 13, 354—356 (1969)

Gaussian Elimination is not Optimal

VOLKER STRASSEN*

Received December 12, 1968

1. Below we will give an algorithm which computes the coefficients of the product of two square matrices A and B of order n from the coefficients of A and B with less than $4.7 \cdot n^{\log 7}$ arithmetical operations (all logarithms in this paper are for base 2, thus $\log 7 \approx 2.8$; the usual method requires approximately $2n^3$ arithmetical operations). The algorithm induces algorithms for inverting a matrix of order n, solving a system of n linear equations in n unknowns, computing a determinant of order n etc. all requiring less than const $n^{\log 7}$ arithmetical operations.

This fact should be compared with the result of KLYUYEV and KOKOVKIN-SHCHERBAK [1] that Gaussian elimination for solving a system of linear equations is optimal if one restricts oneself to operations upon rows and columns as a whole. We also note that WINOGRAD [2] modifies the usual algorithms for matrix multiplication and inversion and for solving systems of linear equations, trading roughly half of the multiplications for additions and subtractions.

Numer. Math. 13, 354—356 (1969)

general methodology: algorithmic reductions

→ concentrate efforts on: **fast basic routines** + **good reductions**

Gaussian Elimination is not Optimal

VOLKER STRASSEN*

Received December 12, 1968

1. Below we will give an algorithm which computes the coefficients of the product of two square matrices A and B of order n from the coefficients of A and B with less than $4.7 \cdot n^{\log 7}$ arithmetical operations (all logarithms in this paper are for base 2, thus $\log 7 \approx 2.8$; the usual method requires approximately $2n^3$ arithmetical operations). The algorithm induces algorithms for inverting a matrix of order n, solving a system of n linear equations in n unknowns, computing a determinant of order n etc. all requiring less than const $n^{\log 7}$ arithmetical operations.

This fact should be compared with the result of KLYUYEV and KOKOVKIN-SHCHERBAK [1] that Gaussian elimination for solving a system of linear equations is optimal if one restricts oneself to operations upon rows and columns as a whole. We also note that WINOGRAD [2] modifies the usual algorithms for matrix multiplication and inversion and for solving systems of linear equations, trading roughly half of the multiplications for additions and subtractions.

Numer. Math. 13, 354—356 (1969)

general methodology: algorithmic reductions

→ concentrate efforts on: **fast basic routines** + **good reductions**

Gaussian Elimination is not Optimal

VOLKER STRASSEN*

Received December 12, 1968

1. Below we will give an algorithm which computes the coefficients of the product of two square matrices A and B of order n from the coefficients of A and B with less than $4.7 \cdot n^{\log 7}$ arithmetical operations (all logarithms in this paper are for base 2, thus $\log 7 \approx 2.8$; the usual method requires approximately $2n^3$ arithmetical operations). The algorithm induces algorithms for inverting a matrix of order n, solving a system of n linear equations in n unknowns, computing a determinant of order n etc. all requiring less than const $n^{\log 7}$ arithmetical operations.

- small prime FFT in NTL:
  ⇝ about **5500 lines** of C++
  ⇝ target operation: FFT
  (including 1200 lines for vectorized version and 1100 for machine word arithmetic...)

- polynomials in $\mathbb{Z}/p\mathbb{Z}[x]$:
  ⇝ about **5500 lines** as well
  ⇝ target operations include:
  · multiplication, truncated inversion, division,
  · interpolation, multipoint evaluation,
  · XGCD, Berlekamp-Massey, resultant,
  · power projection, modular composition, ...

```
sage: M.degree_matrix(shifts=[-1,2], row_wise=False)
[ 0 -2 -1]
[ 5 -2 -2]
```

hermite_form(*include_zero_rows=True, transformation=False*)

Return the Hermite form of this matrix.

The Hermite form is also normalized, i.e., the pivot polynomials are monic.

INPUT:

- include_zero_rows – boolean (default: True); if False, the zero rows in the output deleted
- transformation – boolean (default: False); if True, return the transformation mat

OUTPUT:

```
sage: M.<x> = GF(7)[]
sage: A = matrix(M, 2, 3, [x, 1, 2*x, x, 1+x, 2])
sage: A.hermite_form()
[      x       1     2*x]
[      0       x 5*x + 2]
sage: A.hermite_form(transformation=True)
(
[      x       1     2*x]  [1 0]
[      0       x 5*x + 2], [6 1]
)
sage: A = matrix(M, 2, 3, [x, 1, 2*x, 2*x, 2, 4*x])
sage: A.hermite_form(transformation=True, include_zero_rows=False)
([  x   1 2*x], [0 4])
sage: H, U = A.hermite_form(transformation=True, include_zero_rows=True); H, U
(
[  x   1 2*x]  [0 4]
[  0   0   0], [5 1]
)
sage: U * A == H
True
sage: H, U = A.hermite_form(transformation=True, include_zero_rows=False)
sage: U * A
[  x   1 2*x]
sage: U * A == H
True
```

See also: is_hermite() .

is_hermite(*row_wise=True, lower_echelon=False, include_zero_vectors=True*)

Return a boolean indicating whether this matrix is in Hermite form.

```
164     // order that remains to be dealt with
165     VecLong rem_order(order);
166
167     // indices of columns/orders that remain to be dealt with
168     VecLong rem_index(cdim);
169     std::iota(rem_index.begin(), rem_index.end(), 0);
170
171     // all along the algorithm, shift = shifted row degrees of approximant ba
172     // (initially, input shift = shifted row degree of the identity matrix)
173
174     while (not rem_order.empty())
175     {
176         /** Invariant:
177          *  - appbas is a shift-ordered weak Popov approximant basis for
178          *  (pmat,reached_order) where doneorder is the tuple such that
179          *  -->reached_order[j] + rem_order[j] == order[j] for j appearing in
180          *  -->reached_order[j] == order[j] for j not appearing in rem_index
181          *  - shift == the "input shift"-row degree of appbas
182          *  - residual == submatrix of columns (appbas * pmat)[:,j] for all j
187             j = std::distance(rem_order.begin(), std::max_element(rem_order.b
    );
188
189         long deg = order[rem_index[j]] - rem_order[j];
190
191         // record the coefficients of degree deg of the column j of residual
192         // also keep track of which of these are nonzero,
193         // and among the nonzero ones, which is the first with smallest shift
194         Vec<zz_p> const_residual;
195         const_residual.SetLength(rdim);
196         VecLong indices_nonzero;
197         long piv = -1;
198         for (long i = 0; i < rdim; ++i)
199         {
200             const_residual[i] = coeff(residual[i][j],deg);
201             if (const_residual[i] != 0)
202             {
203                 indices_nonzero.push_back(i);
204                 if (piv<0 || shift[i] < shift[piv])
205                     piv = i;
206             }
207         }
208
209         // if indices_nonzero is empty, const_residual is already zero, there
210         if (not indices_nonzero.empty())
211         {
212             // update all rows of appbas and residual in indices_nonzero exce
```

matrices **software** polynomials

open-source mathematics software system

Python/Cython

high-performance exact linear algebra

LinBox – fflas-ffpack *C/C++*

high-performance polynomials (and more)

FLINT & NTL *C/C++*

- choice of algorithms, use of reductions
- data structures, cache efficiency
- SIMD vectorization instructions
- multithreading, Grids, GPUs, . . . (*)
- but, really, first: choice of algorithms

matrices **software** polynomials

```
sage: M.<x> = GF(7)[]
sage: A = matrix(M, 2, 3, [x, 1, 2*x, x, 1+x, 2])
sage: A.hermite_form()
[      x       1     2*x]
[      0       x 5*x + 2]
sage: A.hermite_form(transformation=True)
(
[      x       1     2*x]  [1 0]
[      0       x 5*x + 2], [6 1]
)
sage: A = matrix(M, 2, 3, [x, 1, 2*x, 2*x, 2, 4*x])
sage: A.hermite_form(transformation=True, include_zero_rows=False)
([  x   1 2*x], [0 4])
sage: H, U = A.hermite_form(transformation=True, include_zero_rows=True); H, U
(
[  x   1 2*x]  [0 4]
[  0   0   0], [5 1]
)
sage: U * A == H
True
sage: H, U = A.hermite_form(transformation=True, include_zero_rows=False)
sage: U * A
[  x   1 2*x]
sage: U * A == H
True
```

See also: `is_hermite()`.

`is_hermite`(*row_wise=True, lower_echelon=False, include_zero_vectors=True*)

Return a boolean indicating whether this matrix is in Hermite form.

```
187            j = std::distance(rem_order.begin(), std::max_element(rem_order.b
    );
188
189        long deg = order[rem_index[j]] - rem_order[j];
190
191        // record the coefficients of degree deg of the column j of residual
192        // also keep track of which of these are nonzero,
193        // and among the nonzero ones, which is the first with smallest shift
194        Vec<zz_p> const_residual;
195        const_residual.SetLength(rdim);
196        VecLong indices_nonzero;
197        long piv = -1;
198        for (long i = 0; i < rdim; ++i)
199        {
200            const_residual[i] = coeff(residual[i][j],deg);
201            if (const_residual[i] != 0)
202            {
203                indices_nonzero.push_back(i);
204                if (piv<0 || shift[i] < shift[piv])
205                    piv = i;
206            }
207        }
208
209        // if indices_nonzero is empty, const_residual is already zero, there
210        if (not indices_nonzero.empty())
211        {
212            // update all rows of appbas and residual in indices_nonzero exce
```

src/mat_lzz_pX_approximant.cpp

open-source mathematics software system

Python/Cython

---

high-performance exact linear algebra

LinBox – fflas-ffpack *C/C++*

---

high-performance polynomials (and more)

FLINT & NTL *C/C++*

- choice of algorithms, use of reductions
- data structures, cache efficiency
- SIMD vectorization instructions
- multithreading, Grids, GPUs, . . . (*)
- but, really, first: choice of algorithms

matrices **software** polynomials

"Yes, this is optimal. No, you should not use it."

"The asymptotically fastest algorithms are rarely used."

"Algorithms with the best asymptotic complexity are often not the fastest in practice."

open-source mathematics software system

Python/Cython

---

high-performance exact linear algebra

LinBox – fflas-ffpack *C/C++*

---

high-performance polynomials (and more)

FLINT & NTL *C/C++*

- choice of algorithms, use of reductions
- data structures, cache efficiency
- SIMD vectorization instructions
- multithreading, Grids, GPUs, . . . (*)
- but, really, first: choice of algorithms

matrices **software** polynomials

"Yes, this is optimal. No, you should not use it."

"The asymptotically fastest algorithms are rarely used."

"Algorithms with the best asymptotic complexity are often not the fastest in practice."

[Cormen-Leiserson-Rivest-Stein, 3rd ed.]

From a practical point of view, Strassen's algorithm is often not the method of choice [among others due to] *the constant factor hidden in the* $O(n^{2.81})$ *running time*

[Knuth, the Art of Computer Programming Vol.2]

Strassen's scheme would not begin to excel [. . .] until $n \approx 250$*; and such enormous matrices rarely occur in practice unless they are very sparse, when other techniques apply*

open-source mathematics software system

Python/Cython

high-performance exact linear algebra

LinBox – fflas-ffpack *C/C++*

high-performance polynomials (and more)

FLINT & NTL *C/C++*

- choice of algorithms, use of reductions
- data structures, cache efficiency
- SIMD vectorization instructions
- multithreading, Grids, GPUs, . . . (*)
- but, really, first: choice of algorithms

matrices **software** polynomials

[Cormen-Leiserson-Rivest-Stein, 3rd ed.]

From a practical point of view, Strassen's method of the constant running time

[amming Vol.2]

egin to excel h enormous e unless they are very sparse, when other techniques apply

"Yes, this is optimal. No, you shou

"The asy algorithm

"Algorith asymptotic c not the fas practice."

an **unfortunate myth** [Bailey-Lee-Simon 1990]

Even recently published reference works have propagated the unfounded assertion that Strassen's algorithm is not suitable for matrices of reasonable size. In fact, for some new workstations, Strassen is faster for matrices as small as 16 x 16; for Cray systems, the crossover point is roughly 128

open-source mathematics software system

Python/Cython

high-performance exact linear algebra

LinBox – fflas-ffpack *C/C++*

high-performance polynomials (and more)

FLINT & NTL *C/C++*

- choice of algorithms, use of reductions
- data structures, cache efficiency
- SIMD vectorization instructions
- multithreading, Grids, GPUs, . . . (*)
- but, really, first: choice of algorithms

matrices **software** polynomials

what you can compute in about 1 second

using FLINT, over $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$ with prime $p \approx 2^{60} \approx 10^{18}$

matrix operation		size	time
▸ `Gauss`	$A = PLU$	2026	0.99s
▸ `LinSys`	$Ax = y$	2026	1.02s
▸ `MatMul`	$C = AB$	1650	1.03s
▸ `Inverse`	$B = A^{-1}$	1250	1.02s
▸ `MinPoly`	$\mu_A(x)$	1000	0.93s

polynomial operation		size	time
▸ `PolMul`	$c = ab$	$8 \cdot 10^6$	0.94s
▸ `Divide`	$a = bq + r$	$6 \cdot 10^6$	1.10s
▸ `MP Eval.`	$a(x_i) = y_i$	$4 \cdot 10^5$	0.96s
▸ `Interp.`	$a(x_i) = y_i$	$3 \cdot 10^5$	0.99s
▸ `XGCD`	$au + bv = g$	$2 \cdot 10^5$	1.23s

outline

- **first guess, then prove**
  - linearly recurrent sequences
  - differentially finite power series
  - guessing algebraicity or D-finiteness
- **algorithms & complexity**
  - algebraic computations
  - asymptotic complexity bounds
  - software performance
- **finding univariate relations**
- **finding multivariate relations**

outline

- **first guess, then prove**
  - linearly recurrent sequences
  - differentially finite power series
  - guessing algebraicity or D-finiteness
- **algorithms & complexity**
  - algebraic computations
  - asymptotic complexity bounds
  - software performance
- **finding univariate relations**
  - Hermite-Padé approximation
  - fast divide and conquer scheme
  - unbalanced degrees and Popov bases
- **finding multivariate relations**

vector approximation/interpolation

Padé approximation:

[in] a power series f at precision d

[out] small-degree polynomials p, q such that $qf = p \bmod x^d$

Cauchy interpolation:

[in] d points $(\alpha_i, \beta_i)_{1 \leqslant i \leqslant d}$ in $\mathbb{K}^2$

[out] small-degree polynomials p, q such that $q(\alpha_i)\beta_i = p(\alpha_i)$ for all i

extensions to **vector equations**:

vector approximation/interpolation

Padé approximation:
[in] a power series f at precision d
[out] small-degree polynomials p, q such that $qf = p \bmod x^d$

Cauchy interpolation:
[in] d points $(\alpha_i, \beta_i)_{1 \leqslant i \leqslant d}$ in $\mathbb{K}^2$
[out] small-degree polynomials p, q such that $q(\alpha_i)\beta_i = p(\alpha_i)$ for all i

extensions to **vector equations**:

Sur la généralisation des fractions continues algébriques;

PAR M. H. PADÉ,

Docteur ès Sciences mathématiques,
Professeur au lycée de Lille.

[1894, Journal de mathématiques pures et appliquées]

INTRODUCTION.

M. Hermite s'est, dans un travail récemment paru (¹), occupé de la généralisation des fractions continues algébriques. La question est de déterminer les polynomes $X_1, X_2, \ldots, X_n$, de degrés $\mu_1, \mu_2, \ldots, \mu_n$, qui satisfont à l'équation

$$p_1 f_1 + \cdots + p_m f_m = 0 \bmod x^d$$

$$S_1 X_1 + S_2 X_2 + \ldots + S_n X_n = S\, x^{\mu_1+\mu_2+\ldots+\mu_n+n-1},$$

$S_1, S_2, \ldots, S_n$ étant des séries entières données, et S une série également entière. Ou plutôt, il s'agit d'obtenir un algorithme qui permette le calcul de proche en proche de ces systèmes de n polynomes, et qui

vector approximation/interpolation

Padé approximation:
[in] a power series f at precision d
[out] small-degree polynomials p, q
such that $qf = p \bmod x^d$

Cauchy interpolation:
[in] d points $(\alpha_i, \beta_i)_{1 \leqslant i \leqslant d}$ in $\mathbb{K}^2$
[out] small-degree polynomials p, q
such that $q(\alpha_i)\beta_i = p(\alpha_i)$ for all i

extensions to **vector equations**:

- Hermite-Padé approximation [Hermite 1893] [Padé 1894] $\alpha_1 = \cdots = \alpha_d = 0$
- vector rational interpolation [Cauchy 1821] [Mahler 1968] $\alpha_i \neq \alpha_j$

unified: vector M-Padé approximation

[in] polynomials $f_1, \ldots, f_m \in \mathbb{K}[x]_{<d}$
[in] polynomial $M = (x - \alpha_1) \cdots (x - \alpha_d)$
[in] degree bounds $s_1, \ldots, s_m \in \mathbb{Z}_{>0}$

[out] polynomials $p_1, \ldots, p_m \in \mathbb{K}[x]$ such that

- $\deg(p_i) < s_i$ for all i
- $p_1 f_1 + \cdots + p_m f_m = 0 \bmod M$

[Mahler 1968]
[van Barel-Bultheel 1992]
[Beckermann 1990, 1992]

polynomial matrices enter the arena

omitting degree constraints, the set of solutions is

$$\mathcal{S} = \{(p_1, \ldots, p_m) \in \mathbb{K}[x]^m \mid p_1 f_1 + \cdots + p_m f_m = 0 \bmod M\}$$

recall $M(x) = \prod_{1 \leqslant i \leqslant d} (x - \alpha_i)$

$\mathcal{S}$ is a "free $\mathbb{K}[x]$-module of rank m": admits a basis consisting of m elements

basis of solutions:

- square nonsingular matrix $\mathbf{P}$ in $\mathbb{K}[x]^{m \times m}$
- each row of $\mathbf{P}$ is a solution $[p_{i,1} \ \cdots \ p_{i,m}]$
- any solution is a $\mathbb{K}[x]$-combination $\mathbf{uP}$, $\mathbf{u} \in \mathbb{K}[x]^{1 \times m}$

i.e. $\mathcal{S}$ is the $\mathbb{K}[x]$-row space of $\mathbf{P}$

polynomial matrices enter the arena

omitting degree constraints, the set of solutions is

$$\mathcal{S} = \{(p_1, \ldots, p_m) \in \mathbb{K}[x]^m \mid p_1 f_1 + \cdots + p_m f_m = 0 \bmod M\}$$

recall $M(x) = \prod_{1 \leqslant i \leqslant d} (x - \alpha_i)$

$\mathcal{S}$ is a "free $\mathbb{K}[x]$-module of rank m": admits a basis consisting of m elements

basis of solutions:

- square nonsingular matrix $\mathbf{P}$ in $\mathbb{K}[x]^{m \times m}$
- each row of $\mathbf{P}$ is a solution $[p_{i,1} \ \cdots \ p_{i,m}]$
- any solution is a $\mathbb{K}[x]$-combination $\mathbf{uP}$, $\mathbf{u} \in \mathbb{K}[x]^{1 \times m}$

i.e. $\mathcal{S}$ is the $\mathbb{K}[x]$-row space of $\mathbf{P}$

computing a basis of $\mathcal{S}$ with "minimal degrees"

- is much more powerful than just one small-degree solution
- is the fastest known strategy anyway (!)

$\rightsquigarrow$ canonical, minimal basis: shifted Popov forms

polynomial matrices: basic operations

$$\mathbf{A} = \begin{bmatrix} 3x+4 & x^3+4x+1 & 4x^2+3 \\ 5 & 5x^2+3x+1 & 5x+3 \\ 3x^3+x^2+5x+3 & 6x+5 & 2x+1 \end{bmatrix} \in \mathbb{K}[x]^{3\times 3}$$

3×3 matrix of degree 3 with entries in $\mathbb{Z}/7\mathbb{Z}[x]$

operations on $\mathbb{K}[x]^{m\times m}_{<d}$

- combination of matrix and polynomial computations
- addition in $O(m^2 d)$, naive multiplication in $O(m^3 d^2)$
- inversion, determinant, division with remainder, ...

[Cantor-Kaltofen'91]

multiplication in $\tilde{O}(m^\omega d)$

precisely: $O(m^\omega d \log(d) + m^2 d \log(d) \log\log(d))$

familiar examples of polynomial matrices?

polynomial matrices: basic operations

$$\mathbf{A} = \begin{bmatrix} 3x+4 & x^3+4x+1 & 4x^2+3 \\ 5 & 5x^2+3x+1 & 5x+3 \\ 3x^3+x^2+5x+3 & 6x+5 & 2x+1 \end{bmatrix} \in \mathbb{K}[x]^{3\times 3}$$

3×3 matrix of degree 3 with entries in $\mathbb{Z}/7\mathbb{Z}[x]$

operations on $\mathbb{K}[x]^{m\times m}_{<d}$

- combination of matrix and polynomial computations
- addition in $O(m^2 d)$, naive multiplication in $O(m^3 d^2)$
- inversion, determinant, division with remainder, ...

[Cantor-Kaltofen'91]

multiplication in $\tilde{O}(m^\omega d)$

precisely: $O(m^\omega d \log(d) + m^2 d \log(d) \log\log(d))$

small matrices with large degree:

extended GCD $au + bv = g = \gcd(a, b)$ for polynomials $a, b \in \mathbb{K}[x]_{\leqslant d}$

$\rightsquigarrow$ corresponds to a polynomial matrix transformation $\begin{bmatrix} u & v \\ \tilde{b} & \tilde{a} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} g \\ 0 \end{bmatrix}$

- fastest known "half-gcd" algorithms use this viewpoint

[Knuth, 1970] [Schönhage, 1971] [Brent-Gustavson-Yun, 1980] [van der Hoeven, 2025]

polynomial matrices: basic operations

$$\mathbf{A} = \begin{bmatrix} 3x+4 & x^3+4x+1 & 4x^2+3 \\ 5 & 5x^2+3x+1 & 5x+3 \\ 3x^3+x^2+5x+3 & 6x+5 & 2x+1 \end{bmatrix} \in \mathbb{K}[x]^{3\times 3}$$

3×3 matrix of degree 3 with entries in $\mathbb{Z}/7\mathbb{Z}[x]$

operations on $\mathbb{K}[x]^{m\times m}_{<d}$

- combination of matrix and polynomial computations
- addition in $O(m^2 d)$, naive multiplication in $O(m^3 d^2)$
- inversion, determinant, division with remainder, ...

[Cantor-Kaltofen'91]

multiplication in $\tilde{O}(m^\omega d)$

precisely: $O(m^\omega d \log(d) + m^2 d \log(d) \log\log(d))$

large matrices with small degrees:
characteristic polynomial $\det(x\mathbf{I}_m - \mathbf{A}) \in \mathbb{K}[x]$ of a matrix $\mathbf{A} \in \mathbb{K}^{m\times m}$
$\rightsquigarrow$ determinant of polynomial matrix $x\mathbf{I}_m - \mathbf{A} \in \mathbb{K}[x]^{m\times m}$

- fastest known algorithm uses this viewpoint [N.-Pernet, 2021]
- gradually transforms $x\mathbf{I}_m - \mathbf{A}$ to smaller matrices with larger degrees

polynomial matrices: canonical forms

consider some arbitrary polynomial matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} x^8+4x^7+4x^6+5x^4+4x^3+3x^2+3x+6 & 5x^9+2x^8+2x^7+2x^6+4x^5+4x^4+4x^3+5x^2+3x+5 & 2x^9+5x^8+3x^6+4x^5+2x^3+4x^2+x+2 & 6x^{10}+3x^8+x^7+x^6+2x^5+3x^4+2x^3+x+2 \\ 6x^6+3x^5+2x^3+4x^2+6 & 2x^7+5x^6+4x^5+6x^4+4x^3+3x^2+4x+3 & 5x^7+2x^6+x^5+3x^4+3x^3+6x^2+x+5 & x^8+6x^5+4x^4+5x^3+2x+2 \\ 5x^6+3x^4+6x^3+4x^2+5x & 4x^7+x^6+5x^5+x^4+6x^2+4x+1 & 3x^7+6x^6+5x^5+5x^4+3x^3+1 & 2x^8+6x^7+x^6+6x^5+x^2+2x \\ 2x^6+6x^5+3x^4+5x^3+3x^2+3x+4 & 3x^7+x^6+x^5+4x^4+2x^3+x^2+3x+3 & 4x^7+6x^6+3x^5+2x^4+3x^3+2x^2+2x+4 & 5x^8+2x^7+3x^6+2x^5+4x^4+3x^3+5x^2+5x+3 \end{bmatrix}$$

by elementary row operations, $\mathbf{A}$ is transformed into...

Popov form $\mathbf{P} = \begin{bmatrix} x & 3 & 1 & 6 \\ 0 & x+6 & 3 & 6 \\ 2 & 2 & x+4 & 5 \\ 0 & 2 & 2 & x+4 \end{bmatrix}$

Hermite form $\mathbf{H} = \begin{bmatrix} x^3+2x^2+3x+4 & 0 & 0 & 0 \\ x^2+x+5 & x+5 & 0 & 0 \\ 3x^2+6x+1 & 6 & 1 & 0 \\ 3x^2+5x+1 & 3 & 0 & 1 \end{bmatrix}$

- more generally: $\mathbf{s}$-Popov form for degree shift $\mathbf{s} = (s_1, \ldots, s_m)$
- canonical basis with minimal $\mathbf{s}$-degree
- strong properties + small size $\Rightarrow$ fast computations

M-Padé: divide and conquer reduction to multiplication

[Beckermann-Labahn 1994,1997] [Giorgi-Jeannerod-Villard '03] [Storjohann '06]
[Zhou-Labahn '09,'12] [Jeannerod-Neiger-Schost-Villard '16,'17,'20]

[in] vector $\mathbf{F}$, polynomial $M = \prod_{1 \leqslant i \leqslant d}(x - \alpha_i)$, degree bounds $\mathbf{s}$

[out] $\mathbf{s}$-minimal basis $\mathbf{P} \in \mathbb{K}[x]^{m \times m}$ of solutions

a. recursive call at first half of the points — $\mathcal{C}(m, \lfloor d/2 \rfloor)$

$\mathbf{P}_1 \leftarrow$ recursive call on $\mathbf{F}, M_1 = \prod_{1 \leqslant i \leqslant \lfloor d/2 \rfloor}(x - \alpha_i), \mathbf{s}$

b. compute residual vector $\mathbf{G}$ and update degree bounds — $\tilde{O}(m^2 d)$

$\mathbf{G} \leftarrow \frac{1}{M_1}\mathbf{P}_1\mathbf{F}, \quad \mathbf{t} \leftarrow \mathbf{s}$-degree of $\mathbf{P}_1$

c. recursive call at second half of the points — $\mathcal{C}(m, \lceil d/2 \rceil)$

$\mathbf{P}_2 \leftarrow$ recursive call on $\mathbf{G}, M_2 = \prod_{\lfloor d/2 \rfloor < i \leqslant d}(x - \alpha_i), \mathbf{t}$

d. multiply bases: return the product $\mathbf{P}_2\mathbf{P}_1$ — $\tilde{O}(m^\omega d)$

M-Padé: divide and conquer reduction to multiplication

[Beckermann-Labahn 1994,1997] [Giorgi-Jeannerod-Villard '03] [Storjohann '06]
[Zhou-Labahn '09,'12] [Jeannerod-Neiger-Schost-Villard '16,'17,'20]

[in] vector $\mathbf{F}$, polynomial $M = \prod_{1 \leqslant i \leqslant d}(x - \alpha_i)$, degree bounds $\mathbf{s}$

[out] $\mathbf{s}$-minimal basis $\mathbf{P} \in \mathbb{K}[x]^{m \times m}$ of solutions

a. recursive call at first half of the points $\mathcal{C}(m, \lfloor d/2 \rfloor)$

b. compute residual vector $\mathbf{G}$ and update degree bounds $\tilde{O}(m^2 d)$

c. recursive call at second half of the points $\mathcal{C}(m, \lceil d/2 \rceil)$

d. multiply bases: return the product $\mathbf{P}_2\mathbf{P}_1$ $\tilde{O}(m^\omega d)$

M-Padé: divide and conquer reduction to multiplication

[Beckermann-Labahn 1994,1997] [Giorgi-Jeannerod-Villard '03] [Storjohann '06]
[Zhou-Labahn '09,'12] [Jeannerod-Neiger-Schost-Villard '16,'17,'20]

[in] vector $\mathbf{F}$, polynomial $M = \prod_{1 \leqslant i \leqslant d}(x - \alpha_i)$, degree bounds $\mathbf{s}$
[out] $\mathbf{s}$-minimal basis $\mathbf{P} \in \mathbb{K}[x]^{m \times m}$ of solutions

a. recursive call at first half of the points $\mathcal{C}(m, \lfloor d/2 \rfloor)$

b. compute residual vector $\mathbf{G}$ and update degree bounds $\tilde{O}(m^2 d)$

c. recursive call at second half of the points $\mathcal{C}(m, \lceil d/2 \rceil)$

d. multiply bases: return the product $\mathbf{P}_2\mathbf{P}_1$ $\tilde{O}(m^\omega d)$

complexity equation $\mathcal{C}(m, d) = 2\mathcal{C}(m, \lfloor d/2 \rceil) + \tilde{O}(m^\omega d)$

complexity = that of multiplication + 1 log factor $\tilde{O}(m^\omega d)$

- good: close to worst-case output size $\Theta(m^2 d)$, reached for unbalanced $\mathbf{s}$
- but: output size is $O(md)$ for uniform $\mathbf{s}$ or for $\mathbf{s}$-Popov basis

M-Padé: divide and conquer reduction to multiplication

[Beckermann-Labahn 1994,1997] [Giorgi-Jeannerod-Villard '03] [Storjohann '06]
[Zhou-Labahn '09,'12] [Jeannerod-Neiger-Schost-Villard '16,'17,'20]

[in] vector $\mathbf{F}$, polynomial $M = \prod_{1 \leqslant i \leqslant d}(x - \alpha_i)$, degree bounds $\mathbf{s}$
[out] $\mathbf{s}$-minimal basis $\mathbf{P} \in \mathbb{K}[x]^{m \times m}$ of solutions

a. recursive call at first half of the points $\mathcal{C}(m, \lfloor d/2 \rfloor)$

b. compute residual vector $\mathbf{G}$ and update degree bounds $\tilde{O}(m^2 d)$

c. recursive call at second half of the points $\mathcal{C}(m, \lceil d/2 \rceil)$

d. multiply bases: return the product $\mathbf{P}_2\mathbf{P}_1$ $\tilde{O}(m^\omega d)$

complexity equation $\mathcal{C}(m, d) = 2\mathcal{C}(m, \lfloor d/2 \rceil) + \tilde{O}(m^\omega d)$

complexity = that of multiplication + 1 log factor $\tilde{O}(m^\omega d)$

- good: close to worst-case output size $\Theta(m^2 d)$, reached for unbalanced $\mathbf{s}$
- but: output size is $O(md)$ for uniform $\mathbf{s}$ or for $\mathbf{s}$-Popov basis

further improvements towards the optimal cost $O(md)$:

- output the $\mathbf{s}$-**Popov basis**
- deal with **unbalanced degrees**

→ $\tilde{O}(m^{\omega-1} d)$

handling unbalanced degrees

key to complexity speed-up $\tilde{O}(m^\omega d) \to \tilde{O}(m^{\omega-1} d)$:
reduce **unbalanced** degrees to **average** degree

[Storjohann 2006] [Zhou-Labahn 2012] [Jeannerod-Neiger-Villard 2020]

from a matrix $\mathbf{A} \in \mathbb{K}[x]^{m \times m}$ with average degree $\delta \ll \deg(\mathbf{A})$
construct a matrix $\bar{\mathbf{A}} \in \mathbb{K}[x]^{m' \times m'}$ with

- a slight increase of matrix dimension: $m \leqslant m' \leqslant 2m$
- a strong decrease of matrix degree: $\deg(\bar{\mathbf{A}}) \leqslant 2\delta$
- preservation of the features targeted by our computations

examples: $\mathbf{AB}$ easily deduced from $\bar{\mathbf{A}}\bar{\mathbf{B}}$; $\det(\mathbf{A}) = \det(\bar{\mathbf{A}})$; $\bar{\mathbf{A}}^{-1}$ contains $\mathbf{A}^{-1}$; etc.

handling unbalanced degrees

key to complexity speed-up $\tilde{O}(m^\omega d) \to \tilde{O}(m^{\omega-1} d)$:
reduce **unbalanced** degrees to **average** degree

[Storjohann 2006] [Zhou-Labahn 2012] [Jeannerod-Neiger-Villard 2020]

from a matrix $\mathbf{A} \in \mathbb{K}[x]^{m \times m}$ with average degree $\delta \ll \deg(\mathbf{A})$
construct a matrix $\bar{\mathbf{A}} \in \mathbb{K}[x]^{m' \times m'}$ with

- a slight increase of matrix dimension: $m \leqslant m' \leqslant 2m$
- a strong decrease of matrix degree: $\deg(\bar{\mathbf{A}}) \leqslant 2\delta$
- preservation of the features targeted by our computations

examples: $\mathbf{AB}$ easily deduced from $\bar{\mathbf{A}}\bar{\mathbf{B}}$; $\det(\mathbf{A}) = \det(\bar{\mathbf{A}})$; $\bar{\mathbf{A}}^{-1}$ contains $\mathbf{A}^{-1}$; etc.

			matrix-vector product			Hermite-Padé		
m	d	$\delta = \frac{d}{m}$	$\tilde{O}(m^2 d)$	$\tilde{O}(m^{\omega-1} d)$	ratio	$\tilde{O}(m^\omega d)$	$\tilde{O}(m^{\omega-1} d)$	ratio
50	4000	80	0.365	0.085	4.29	1.01	0.169	5.98
100	4000	40	1.10	0.136	8.09	5.48	0.396	13.8
200	4000	20	2.78	0.286	9.72	35.5	1.01	35.1
400	4000	10	7.94	0.684	11.6	279	2.90	96.2

univariate relations: an open problem

structured relations

$\rightsquigarrow$ P-recursive recurrences, algebraic/D-finite series, modular composition, bivariate interpolation, ...

$$p_1 f_1 + p_2 f_2 + \cdots + p_m f_m = 0 \bmod M$$

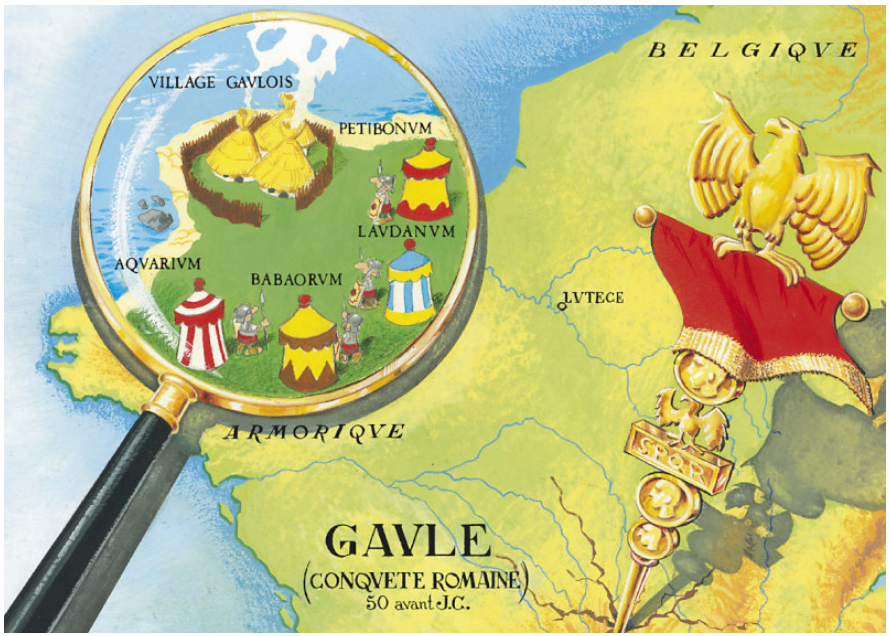
$$\downarrow \text{ structured } f_i\text{'s}$$

$$p_1 1 + p_2 f + \cdots + p_m f^{m-1} = 0 \bmod M$$

$$p_1 f + p_2 f' + \cdots + p_m f^{(m-1)} = 0 \bmod M$$

- here, input/output size is $O(d)$
- most algorithms ignore this structure
- some recent progress [Villard 2018]

how to leverage this structure?



The year is 2026 A.D.

Algebraic Relations Guessing is entirely occupied by Computer Algebraists.

Well not entirely!

One small village of indomitable open problems still holds out against the invaders. And life is not easy for the scientists who garrison the fortified camps of Universities, CNRS, Inria...

outline

first guess, then prove

- linearly recurrent sequences
- differentially finite power series
- guessing algebraicity or D-finiteness

algorithms & complexity

- algebraic computations
- asymptotic complexity bounds
- software performance

finding univariate relations

- Hermite-Padé approximation
- fast divide and conquer scheme
- unbalanced degrees and Popov bases

finding multivariate relations

outline

first guess, then prove

- linearly recurrent sequences
- differentially finite power series
- guessing algebraicity or D-finiteness

algorithms & complexity

- algebraic computations
- asymptotic complexity bounds
- software performance

finding univariate relations

- Hermite-Padé approximation
- fast divide and conquer scheme
- unbalanced degrees and Popov bases

finding multivariate relations

- r-dimensional recurrent sequences
- Gröbner bases via linear algebra
- fast divide and conquer scheme?

Schönhage's rule: the development of fast algorithms is slow!

first guess, then prove

- linearly recurrent sequences
- differentially finite power series
- guessing algebraicity or D-finiteness

algorithms & complexity

- algebraic computations
- asymptotic complexity bounds
- software performance

finding univariate relations

- Hermite-Padé approximation
- fast divide and conquer scheme
- unbalanced degrees and Popov bases

finding multivariate relations

- $\mathfrak{r}$-dimensional recurrent sequences
- Gröbner bases via linear algebra
- fast divide and conquer scheme?