Vincent Neiger
LIP6, Sorbonne Université, France

designing and **exploiting** **fast algorithms** for **univariate polynomial matrices**

disclaimer

- don't hesitate to interrupt me with questions or comments
- slides will be available, and SageMath notebook too
- many thanks to ISSAC organizers and committees

disclaimer

- don't hesitate to interrupt me with questions or comments
- slides will be available, and SageMath notebook too
- many thanks to ISSAC organizers and committees

Schönhage's rule 3: Do trust the truth!

disclaimer

- don't hesitate to interrupt me with questions or comments
- slides will be available, and SageMath notebook too
- many thanks to ISSAC organizers and committees

Schönhage's rule 3: Do trust the truth!

abstract:
Matrices whose entries are univariate polynomials over a field. . .

[. . .]

. . . Finally, we will conclude with a focus on the use of polynomial matrices to accelerate the modular composition of univariate polynomials and the multipoint evaluation of bivariate polynomials.

fake news! but, talk on Thursday

outline

- first guess, then prove
- polynomials and matrices
- guessing univariate relations
- software, algorithms, impacts

outline

- **first guess, then prove**
  - linearly recurrent sequences
  - guessing algebraicity or D-finiteness
  - Hermite-Padé approximants
- **polynomials and matrices**
- **guessing univariate relations**
- **software, algorithms, impacts**

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?

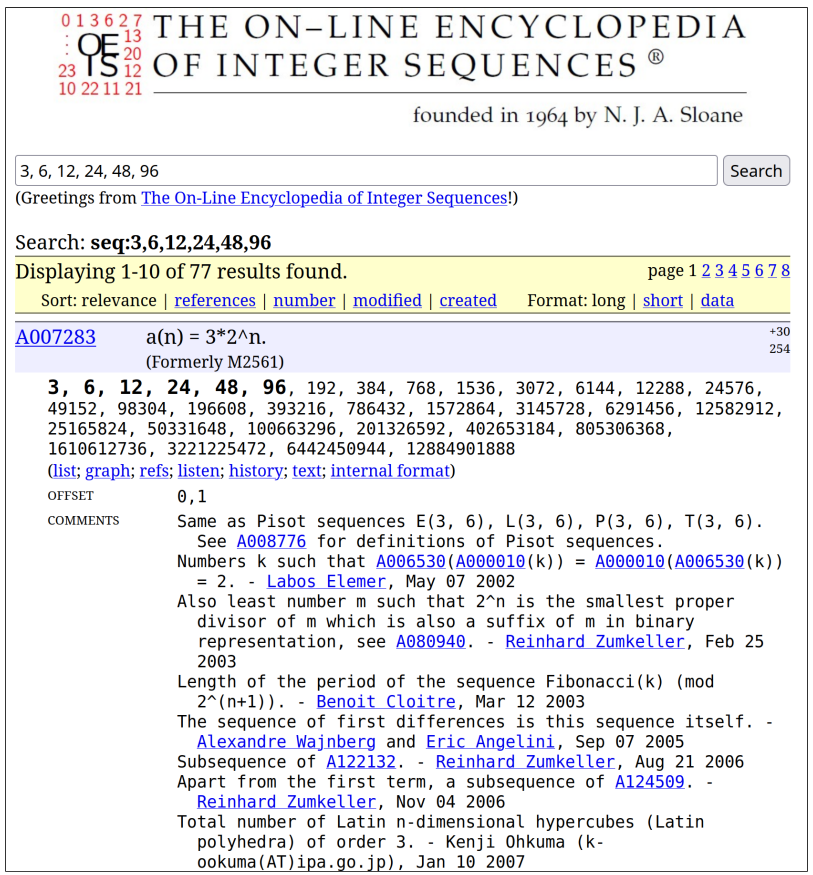
0, 1, 2, 3, 4, 5, **6**

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, **?**



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

$u_n = n$, integers

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

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**

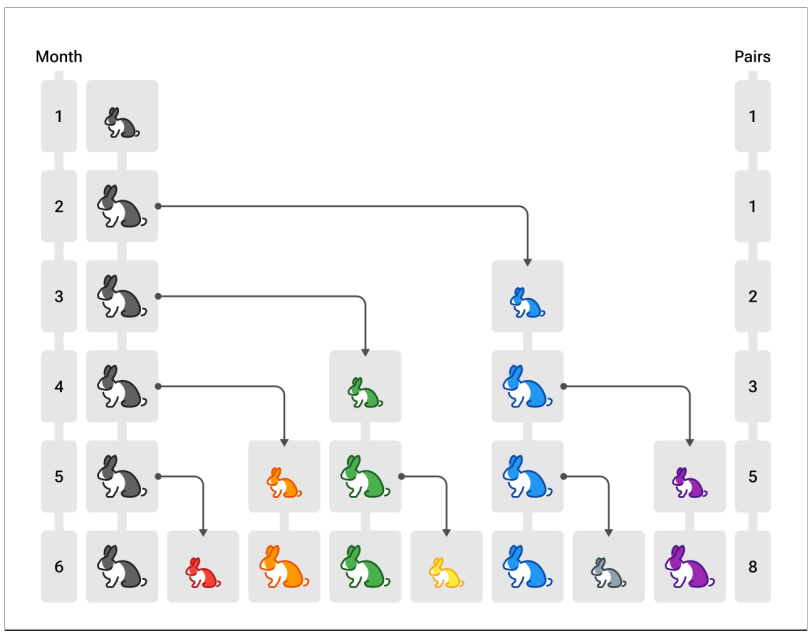
$u_n = 3 \cdot 2^n$, geometric

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?

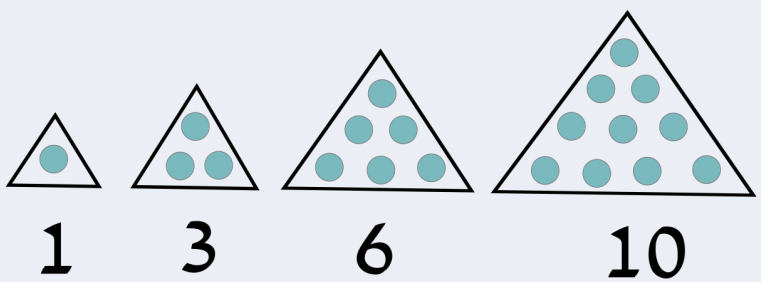
0, 1, 2, 3, 4, 5, **6** — $u_n = n$, integers

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?

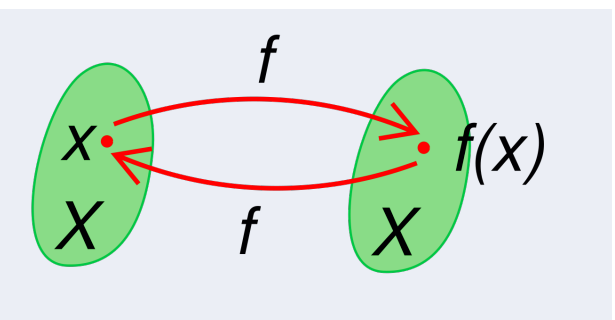
0, 1, 2, 3, 4, 5, **6** — $u_n = n$, integers

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 → **make conjectures** → **find proof**

algorithmic 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**

algorithmic 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**

algorithmic 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]. 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**

algorithmic 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].

proof?

[Kauers-Koutschan, 2022]

core algorithmic tool:
Hermite-Padé approximants
[Hermite 1873+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 → **make conjectures** → **find proof**

algorithmic guessing of algebraic relations

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

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**

algorithmic guessing of algebraic relations

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

consider $u_n = \lfloor n \cdot \tanh(\pi) \rfloor$ and $f(x) = \sum_n u_n x^n$ [Borwein & Borwein, 1992]

- numerically: $f(\frac{1}{10}) = \frac{1}{81}$ at precision 250 digits
- Padé approximation: $f(x) = \frac{x^2}{(x-1)^2} + O(x^{250})$

$\rightsquigarrow$ well, $f(x)$ must be 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**

algorithmic guessing of algebraic relations

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

consider $u_n = \lfloor n \cdot \tanh(\pi) \rfloor$ and $f(x) = \sum_n u_n x^n$ [Borwein & Borwein, 1992]

- numerically: $f(\frac{1}{10}) = \frac{1}{81}$ at precision 250 digits
- Padé approximation: $f(x) = \frac{x^2}{(x-1)^2} + O(x^{250})$

$\rightsquigarrow$ well, $f(x)$ must be 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**

algorithmic guessing of algebraic relations

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

consider $u_n = \lfloor n \cdot \tanh(\pi) \rfloor$ and $f(x) = \sum_n u_n x^n$ [Borwein & Borwein, 1992]

- numerically: $f(\frac{1}{10}) = \frac{1}{81}$ at precision 250 digits
- Padé approximation: $f(x) = \frac{x^2}{(x-1)^2} + O(x^{250})$

$\rightsquigarrow$ well, $f(x)$ must be rational!

- a guess is not a proof, especially if based on approximated data
- this talk focuses on **algorithmic techniques** used in guessing
- for many purposes, an approximation is useful in its own right

"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**

algorithmic 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

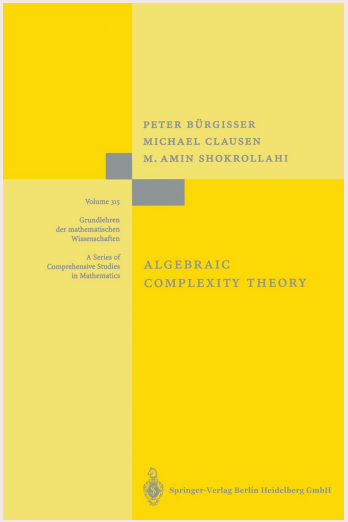
we needed careful implementations of sophisticated algorithms

outline

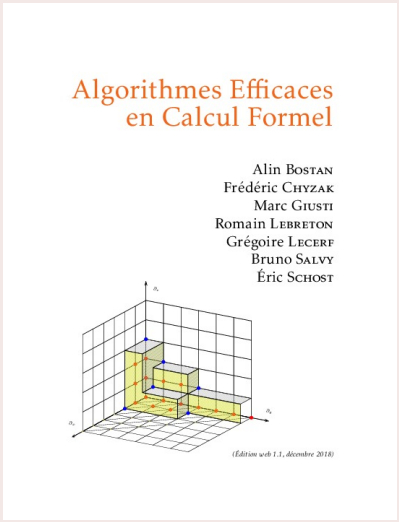
- **first guess, then prove**
  - linearly recurrent sequences
  - guessing algebraicity or D-finiteness
  - Hermite-Padé approximants
- **polynomials and matrices**
- **guessing univariate relations**
- **software, algorithms, impacts**

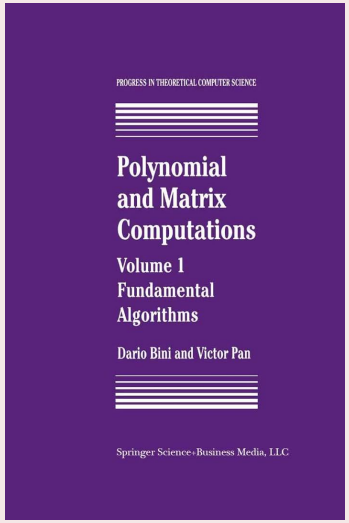
outline

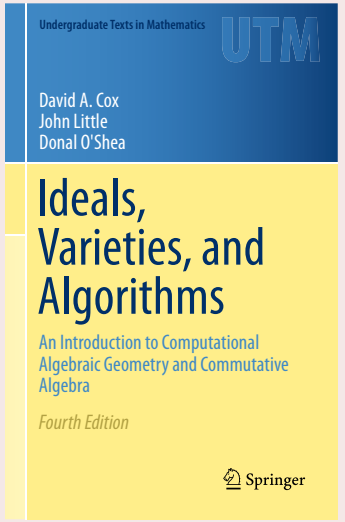
- **first guess, then prove**
  - linearly recurrent sequences
  - guessing algebraicity or D-finiteness
  - Hermite-Padé approximants
- **polynomials and matrices**
  - algebraic computations
  - asymptotic complexity bounds
  - basics of polynomial matrices
- **guessing univariate relations**
- **software, algorithms, impacts**

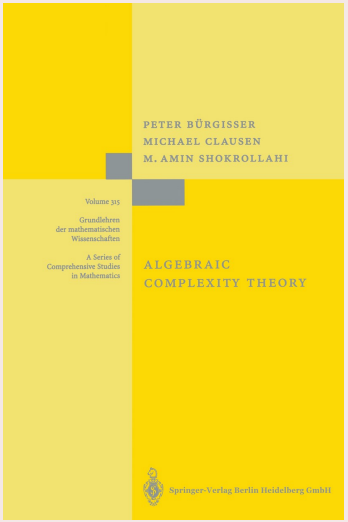

PETER BÜRGISSER
MICHAEL CLAUSEN
M. AMIN SHOKROLLAHI
Volume 315
Grundlehren der mathematischen Wissenschaften
A Series of Comprehensive Studies in Mathematics
ALGEBRAIC COMPLEXITY THEORY
Springer-Verlag Berlin Heidelberg GmbH

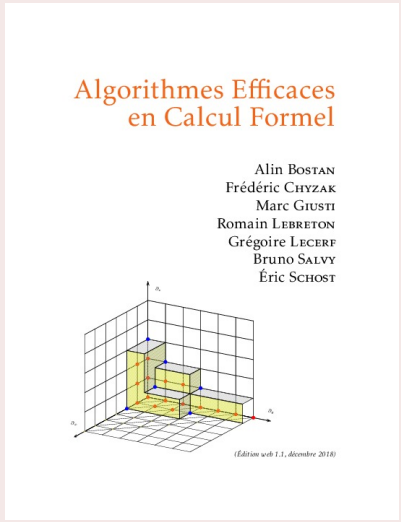

Modern Computer Algebra
Third Edition
Joachim von zur Gathen and Jürgen Gerhard
CAMBRIDGE
CAMBRIDGE
more information – www.cambridge.org/9781107039032

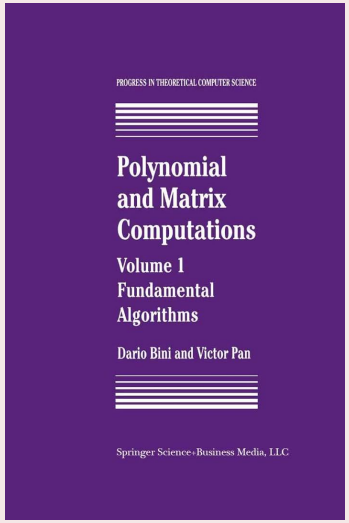

Algorithmes Efficaces en Calcul Formel
Alin Bostan
Frédéric Chyzak
Marc Giusti
Romain Lebreton
Grégoire Lecerf
Bruno Salvy
Éric Schost
(Édition web 1.1, décembre 2018)

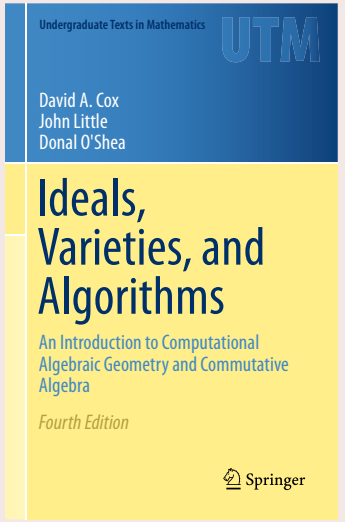

PROGRESS IN THEORETICAL COMPUTER SCIENCE
Polynomial and Matrix Computations
Volume 1
Fundamental Algorithms
Dario Bini and Victor Pan
Springer Science+Business Media, LLC


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 '65+'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 most 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 most 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

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

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

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

begin to excel enormous unless they are very sparse, when other techniques apply

"Yes, this is optimal. No, you should

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

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

polynomial matrices: introduction

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

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

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

- combination of matrix and polynomial computations
- addition in $O(m^2 d)$
- naive multiplication in $O(m^3 d^2)$

[Cantor-Kaltofen'91]

multiplication in $\mathsf{MM}(m, d) \in O(m^\omega \mathsf{M}(d))$

(more accurate bound in a few slides)

polynomial matrices: introduction

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

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

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

- combination of matrix and polynomial computations
- addition in $O(m^2 d)$
- naive multiplication in $O(m^3 d^2)$

[Cantor-Kaltofen'91]

multiplication in $MM(m, d) \in O(m^\omega M(d))$

(more accurate bound in a few slides)

- structure: matrices over $\mathbb{K}[x]$ $\longleftrightarrow$ "free" modules over $\mathbb{K}[x]$

similarly to: matrices over $\mathbb{K}$ $\longleftrightarrow$ vector spaces over $\mathbb{K}$

- $\mathbb{K}[x]$ is not a field, but $\mathbb{K}[x]^{m\times n} \subset \mathbb{K}(x)^{m\times n}$

$\rightsquigarrow$ usual definitions of determinant, rank, inverse*, linear systems*, etc.

$\rightsquigarrow$ running $\mathbb{K}$-matrix algorithms on $\mathbb{K}[x]$-matrices is often a bad idea

degree explosion: Gaussian elimination can be exponential-time

polynomial matrices: introduction

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

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

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

- combination of matrix and polynomial computations
- addition in $O(m^2 d)$
- naive multiplication in $O(m^3 d^2)$

[Cantor-Kaltofen'91]

multiplication in $\mathsf{MM}(m, d) \in O(m^\omega \mathsf{M}(d))$

(more accurate bound in a few slides)

familiar appearances of polynomial matrices?

polynomial matrices: introduction

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

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

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

- combination of matrix and polynomial computations
- addition in $O(m^2 d)$
- naive multiplication in $O(m^3 d^2)$

[Cantor-Kaltofen'91]

multiplication in $MM(m,d) \in O(m^\omega M(d))$

(more accurate bound in a few slides)

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 $\begin{bmatrix} u & v \\ \bar{b} & \bar{a} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} g \\ 0 \end{bmatrix}$

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

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

polynomial matrices: introduction

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

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

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

- combination of matrix and polynomial computations
- addition in $O(m^2 d)$
- naive multiplication in $O(m^3 d^2)$

[Cantor-Kaltofen'91]

multiplication in $\mathrm{MM}(m, d) \in O(m^\omega \mathrm{M}(d))$

(more accurate bound in a few slides)

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

- fastest known algorithm uses this viewpoint [N.-Pernet, 2021]
- gradually transforms $xI_m - A$ to smaller matrices with larger degrees

polynomial matrices: introduction

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

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

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

- combination of matrix and polynomial computations
- addition in $O(m^2 d)$
- naive multiplication in $O(m^3 d^2)$

[Cantor-Kaltofen'91]

multiplication in $MM(m,d) \in O(m^\omega M(d))$

(more accurate bound in a few slides)

complexity with 2 parameters (m,d) breaks some habits:

- Karatsuba's algorithm is already interesting for degree $d = 1$
- Strassen's algorithm is already interesting for size $m = 2$

polynomial matrices as polynomials

$\mathbb{K}[x]^{m\times n}$ is isomorphic to $\mathbb{K}^{m\times n}[x]$

$$\begin{bmatrix} 3x+4 & x^3+4x+1 & 4x^2+3 \\ 5 & 5x^2+3x+1 & 5x+3 \\ 3x^3+x^2+5x+3 & 6x+5 & 2x+1 \end{bmatrix}$$

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

polynomial matrices as polynomials

$\mathbb{K}[x]^{m\times n}$ **is isomorphic to** $\mathbb{K}^{m\times n}[x]$

$$\begin{bmatrix} 3x+4 & x^3+4x+1 & 4x^2+3 \\ 5 & 5x^2+3x+1 & 5x+3 \\ 3x^3+x^2+5x+3 & 6x+5 & 2x+1 \end{bmatrix}$$

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

- natural notion of **degree** of a polynomial matrix
- **addition** of $A, B \in \mathbb{K}[x]^{m\times n}_{<d}$ is in $O(mnd)$ operations in $\mathbb{K}$
- other "entry-wise" polynomial operations:
truncation A mod x^N, shift $x^d A$, evaluation $A(\alpha)$, ...
  - suggests **two data structures**:
array of mn polynomials in $\mathbb{K}[x]_{<d}$
versus
array of d matrices in $\mathbb{K}^{m\times n}$

critical choice
in implementations

polynomial matrices as polynomials

$\mathbb{K}[x]^{m\times n}$ **is isomorphic to** $\mathbb{K}^{m\times n}[x]$

$$\begin{bmatrix} 3x+4 & x^3+4x+1 & 4x^2+3 \\ 5 & 5x^2+3x+1 & 5x+3 \\ 3x^3+x^2+5x+3 & 6x+5 & 2x+1 \end{bmatrix}$$

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

when $m = n$, $\mathbb{K}^{m\times m}$ is a (non-commutative) ring

- **multiplication** in $\mathbb{K}[x]^{m\times m}$ seen as a product of polynomials

what algorithm, what complexity?

- **truncated inversion** via power series & Newton iteration

any issue due to non-commutativity?

- fast **Euclidean division with remainder**

under what conditions on the divisor?

On fast multiplication of polynomials over arbitrary algebras

David G. Cantor[1] **and Erich Kaltofen**[2] ⋆

[1] Department of Mathematics, University of California, Los Angeles, CA 90024-1555, USA
[2] Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, USA

Received January 22, 1988 / May 10, 1991

1 Introduction

In this paper we generalize the well-known Schönhage-Strassen algorithm for multiplying large integers to an algorithm for multiplying polynomials with coefficients from an arbitrary, not necessarily commutative, not necessarily associative, algebra $\mathcal{A}$. Our main result is an algorithm to multiply polynomials of degree $< n$ in $O(n \log n)$ algebra multiplications and $O(n \log n \operatorname{loglog} n)$ algebra additions/subtractions (we count a subtraction as an addition). The constant implied by the "O" does not depend upon the algebra $\mathcal{A}$. The parallel complexity of our algorithm, i.e., the depth of the corresponding arithmetic circuit, is

polynomial matrices as polynomials: fast multiplication

On fast multiplication of polynomials over arbitrary algebras

David G. Cantor[1] **and Erich Kaltofen**[2] ⋆

[1] Department of Mathematics, University of California, Los Angeles, CA 90024-1555, USA
[2] Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, USA

Received January 22, 1988 / May 10, 1991

multiplication in $\mathbb{K}^{m\times m}[x]$ with degree $< d$:
- $O(d\log(d))$ multiplications in $\mathbb{K}^{m\times m}$
- $O(d\log(d)\log\log(d))$ additions in $\mathbb{K}^{m\times m}$

$MM(m,d) \in O(m^\omega d\log(d)+m^2 d\log(d)\log\log(d))$

1 Introduction

In this paper we generalize the well-known Schönhage-Strassen algorithm for multiplying large integers to an algorithm for multiplying polynomials with coefficients from an arbitrary, not necessarily commutative, not necessarily associative, algebra $\mathscr{A}$. Our main result is an algorithm to multiply polynomials of degree $< n$ in $O(n \log n)$ algebra multiplications and $O(n \log n \operatorname{loglog} n)$ algebra additions/subtractions (we count a subtraction as an addition). The constant implied by the "O" does not depend upon the algebra $\mathscr{A}$. The parallel complexity of our algorithm, i.e., the depth of the corresponding arithmetic circuit, is

polynomial matrices as polynomials: fast multiplication

On fast multiplication of polynomials over arbitrary algebras

David G. Cantor[1] **and Erich Kaltofen**[2] ⋆

[1] Department of Mathematics, University of California, Los Angeles, CA 90024-1555, USA
[2] Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, USA

Received January 22, 1988 / May 10, 1991

multiplication in $\mathbb{K}^{m\times m}[x]$ with degree $< d$:
- $O(d\log(d))$ multiplications in $\mathbb{K}^{m\times m}$
- $O(d\log(d)\log\log(d))$ additions in $\mathbb{K}^{m\times m}$

$MM(m,d) \in O(m^\omega d\log(d)+m^2 d\log(d)\log\log(d))$

improvement for many fields $\mathbb{K}$ [Bostan-Schost '05]:
$MM(m,d) \in O(m^\omega d + m^2 M(d))$
(**evaluation-interpolation** at a geometric progression)

1 Introduction

In this paper we generalize the well known Schönhage-Strassen algorithm for multiplying large integers to an algorithm for multiplying polynomials with coefficients from an arbitrary, not necessarily commutative, associative, algebra $\mathscr{A}$. Our main result is that the product of two polynomials of degree $< n$ in $O(n \log n)$ algebra multiplications/divisions and $O(n \log n \log\log n)$ additions/subtractions (we count a subtraction as an addition). The constant implied by the "O" does not depend upon the algebra $\mathscr{A}$. The parallel complexity of our algorithm, i.e., the depth of the corresponding arithmetic circuit, is

polynomial matrices as polynomials: truncated inversion

[Kung, 1974]

2. Newton Iteration

We will use notation of Sieveking (1972) and Strassen (1973). Let k be an infinite field, a_i, b_i, $i = 0, 1, \ldots$, indeterminates over k, A an extension field of k, and t an indeterminate over A. Suppose that E and F are finite subsets of A and that we do computations in the field A. Let $L(E \bmod F)$ denote the number of multiplications or divisions by units which are necessary to compute E starting from $k \cup F$ not counting multiplications by scalars in k. In this paper, A will be taken to be one of $k(a_0, a_1, \ldots)$, $k(a_0, b_0, a_1, b_1, \ldots)$ or $k(a_0)$. One should note that all algorithms presented in this paper do not require the commutativity relation among $a_0, b_0, a_1, b_1, \ldots$. Therefore, a_i, b_i could be, for example, matrices. We shall prove the following theorem by using Newton iteration.

Theorem 2.1. *Suppose* $A = k(a_0, a_1, \ldots)$, a_0 *is a unit in* A *and*

polynomial matrices as polynomials: truncated inversion

[Kung, 1974]

2. Newton Iteration

We will use notation of Sieveking (1972) and Strassen (1973). Let k be an infinite field, $a_i, b_i, i = 0, 1, \ldots$, indeterminates over k, A an extension field of k, and t an indeterminate over A. Suppose that E and F are finite subsets of A and that we do computations in the field A. Let $L(E \bmod F)$ denote the number of multiplications or divisions by units which are necessary to compute E starting from $k \cup F$ not counting multiplications by scalars in k. In this paper, A will be taken to be one of $k(a_0, a_1, \ldots)$, $k(a_0, b_0, a_1, b_1, \ldots)$ or $k(a_0)$. One should note that all algorithms presented in this paper do not require the commutativity relation among $a_0, b_0, a_1, b_1, \ldots$. Therefore, a_i, b_i could be, for example, matrices. We shall prove the following theorem by using Newton iteration.

Theorem 2.1. *Suppose* $A = k(a_0, a_1, \ldots)$, a_0 *is a unit in* A *and*

- $A \in \mathbb{K}^{m \times m}[x]$ is invertible as a power series
  $\Leftrightarrow$ its constant term $A(0) \in \mathbb{K}^{m \times m}$ is invertible
- **Newton iteration:** $U_{k+1} \leftarrow U_k + (I_m - U_k A)U_k \bmod x^{2^{k+1}}$
  if $U_k = A^{-1} \bmod x^{2^k}$, then $U_{k+1} = A^{-1} \bmod x^{2^{k+1}}$
  $\rightsquigarrow$ computing $A^{-1} \bmod x^N$ costs $O(MM(m, N))$

polynomial matrices as polynomials: division with remainder

[in] $A \in \mathbb{K}^{k\times m}[x]$ and $B \in \mathbb{K}^{m\times m}[x]$ with $\mathrm{lc}(B)$ invertible
[out] the unique $Q, R \in \mathbb{K}^{k\times m}[x]$ s.t. $A = QB + R$ and $\deg(R) < \deg(B)$

notation: $\deg(A) = d \geqslant e = \deg(B)$,
$\mathrm{lc}(B) = B_e$ where $B = B_0 + B_1 x + \cdots + B_e x^e$

matrix version of the **classical fast division** algorithm [Strassen 1973]:

1. reverse the equation:
   $A = QB + R$ becomes $\bar{A} = \bar{Q}\bar{B} + x^{d-e+1}\bar{R}$ $\quad O(1)$
2. find quotient by truncated inversion ($\bar{B}_0 = \mathrm{lc}(B)$):
   $\bar{Q} \leftarrow \bar{A}\bar{B}^{-1} \bmod x^{d-e+1}$ $\quad O(\mathrm{MM}(m, d-e))$
3. deduce remainder by truncated multiplication:
   $R \leftarrow (A - QB) \bmod x^e$ $\quad O(\mathrm{MM}(m, e))$

these bounds assume $k \in O(m)$

polynomial matrices as polynomials: division with remainder

[in] $A \in \mathbb{K}^{k\times m}[x]$ and $B \in \mathbb{K}^{m\times m}[x]$ with $\mathrm{lc}(B)$ invertible
[out] the unique $Q, R \in \mathbb{K}^{k\times m}[x]$ s.t. $A = QB + R$ and $\deg(R) < \deg(B)$

notation: $\deg(A) = d \geqslant e = \deg(B)$,
$\mathrm{lc}(B) = B_e$ where $B = B_0 + B_1 x + \cdots + B_e x^e$

matrix version of the **classical fast division** algorithm [Strassen 1973]:

1. reverse the equation:
   $A = QB + R$ becomes $\bar{A} = \bar{Q}\bar{B} + x^{d-e+1}\bar{R}$ $\quad O(1)$
2. find quotient by truncated inversion ($\bar{B}_0 = \mathrm{lc}(B)$):
   $\bar{Q} \leftarrow \bar{A}\bar{B}^{-1} \bmod x^{d-e+1}$ $\quad O(\mathrm{MM}(m, d-e))$
3. deduce remainder by truncated multiplication:
   $R \leftarrow (A - QB) \bmod x^e$ $\quad O(\mathrm{MM}(m, e))$

these bounds assume $k \in O(m)$

consequences: operations on the row span $\langle B \rangle \subset \mathbb{K}[x]^{1\times m}$
- canonical representation in the quotient module $\mathbb{K}[x]^{1\times m}/\langle B \rangle$
- test module membership: is the vector $v \in \mathbb{K}[x]^{1\times m}$ in $\langle B \rangle$?

$\rightsquigarrow$ remove the assumption on $\mathrm{lc}(B)$ via Popov forms $=$ $\mathbb{K}[x]$-module Gröbner bases

outline

- **first guess, then prove**
  - linearly recurrent sequences
  - guessing algebraicity or D-finiteness
  - Hermite-Padé approximants
- **polynomials and matrices**
  - algebraic computations
  - asymptotic complexity bounds
  - basics of polynomial matrices
- **guessing univariate relations**
- **software, algorithms, impacts**

outline

first guess, then prove

- linearly recurrent sequences
- guessing algebraicity or D-finiteness
- Hermite-Padé approximants

polynomials and matrices

- algebraic computations
- asymptotic complexity bounds
- basics of polynomial matrices

guessing univariate relations

- vector approximation/interpolation
- fast divide and conquer scheme
- unbalanced degrees and Popov bases

software, algorithms, impacts

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 1873+93] [Padé 1892+94] $\alpha_1 = \cdots = \alpha_d = 0$
- 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

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

omitting degree constraints, the set of solutions is

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

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

basis of solutions:

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

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

polynomial matrices enter the arena

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

omitting degree constraints, the set of solutions is
$\mathcal{S} = \{(p_1, \ldots, p_m) \in \mathbb{K}[x]^m \mid p_1 f_1 + \cdots + p_m f_m = 0 \bmod M\}$

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

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

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

computing a basis of $\mathcal{S}$ with "minimal degrees"
- is more powerful than just a small-degree solution
- is the fastest known strategy anyway (!)

$\rightsquigarrow$ shifted minimal bases: minimal row degrees ← a.k.a. shifted reduced Gröbner basis of $\mathcal{S}$

$\rightsquigarrow$ shifted Popov bases: canonical form ←

shift: takes degree bounds into account + arises in algorithms
[van Barel-Bultheel 1992] [Beckermann-Labahn-Villard 1999]

polynomial matrices: canonical forms

consider some arbitrary polynomial matrix A:

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

by elementary row operations, A is transformed into...

Popov form [Popov 1972]

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

Hermite form [Hermite 1851]

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

- 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

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

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

a. recursive call at first part of the points

$P_1 \leftarrow$ recursive call on $F, M_1 = \prod_{1 \leqslant i \leqslant d_1}(x - \alpha_i), \mathbf{s}$

b. compute residual vector G and update degree shift

$G \leftarrow \frac{1}{M_1} P_1 F, \quad \mathbf{t} \leftarrow \mathbf{s}$-degree of P_1

c. recursive call at second part of the points

$P_2 \leftarrow$ recursive call on $G, M_2 = \prod_{d_1 < i \leqslant d}(x - \alpha_i), \mathbf{t}$

d. multiply bases: return the product $P_2 P_1$

[Beckermann-Labahn 1994,1997] [Giorgi-Jeannerod-Villard '03] [Storjohann '06]
[Zhou-Labahn '09,'12] [Jeannerod-Neiger-Schost-Villard '16,'17,'20]
when $m = 2$, related to XGCD / half-GCD algorithms [Knuth 1970] [Schönhage 1971] [Moenck 1973]

M-Padé: divide and conquer reduction to multiplication

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

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

a. recursive call at first part of the points

$P_1 \leftarrow$ recursive call on $F, M_1 = \prod_{1 \leqslant i \leqslant d_1}(x - \alpha_i), \mathbf{s}$

b. compute residual vector G and update degree shift

$G \leftarrow \frac{1}{M_1} P_1 F, \quad \mathbf{t} \leftarrow \mathbf{s}$-degree of P_1

c. recursive call at second part of the points

$P_2 \leftarrow$ recursive call on $G, M_2 = \prod_{d_1 < i \leqslant d}(x - \alpha_i), \mathbf{t}$

d. multiply bases: return the product $P_2 P_1$

[Beckermann-Labahn 1994,1997] [Giorgi-Jeannerod-Villard '03] [Storjohann '06]
[Zhou-Labahn '09,'12] [Jeannerod-Neiger-Schost-Villard '16,'17,'20]
when $m = 2$, related to XGCD / half-GCD algorithms [Knuth 1970] [Schönhage 1971] [Moenck 1973]

correctness:

- correctness of base case (for small d)
- basis: $pF = 0 \bmod M \Leftrightarrow p = uP_1$ and $uP_1F = 0 \bmod M$
  $\Leftrightarrow p = uP_1$ and $uG = 0 \bmod M_2 \Leftrightarrow p = vP_2P_1$
- P_2P_1 is $\mathbf{s}$-minimal $\Leftarrow$ P_1 is $\mathbf{s}$-minimal and P_2 is $\mathbf{t}$-minimal

M-Padé: divide and conquer reduction to multiplication

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

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

a. recursive call at first part of the points

b. compute residual vector G and update degree shift

c. recursive call at second part of the points

d. multiply bases: return the product $P_2 P_1$

M-Padé: divide and conquer reduction to multiplication

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

[out] $\mathbf{s}$-minimal basis $P \in \mathbb{K}[x]^{m \times m}$ of solutions with $d_1 = \lfloor d/2 \rfloor$

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

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

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

d. multiply bases: return the product $P_2 P_1$ $\tilde{O}(m^\omega d)$

$\rightsquigarrow$ output basis is $\mathbf{s}$-minimal, but not $\mathbf{s}$-Popov (not preserved by multiplication)

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

- with $d_1 = \lfloor d/2 \rfloor$, complexity equation $\mathcal{C}(m, d) = 2\mathcal{C}(m, \lfloor d/2 \rceil) + \tilde{O}(m^\omega d)$
- good: close to worst-case output size $\Theta(m^2 d)$, reached for unbalanced $\mathbf{s}$
- but: output size is $O(md)$ for balanced $\mathbf{s}$ or for $\mathbf{s}$-Popov basis

M-Padé: divide and conquer reduction to multiplication

[in] vector F, polynomial $M = \prod_{1 \leqslant i \leqslant d}(x - \alpha_i)$, degree shift $\mathbf{s}$
[out] $\mathbf{s}$-minimal basis $P \in \mathbb{K}[x]^{m \times m}$ of solutions with $d_1 = \lfloor d/2 \rfloor$

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

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

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

d. multiply bases: return the product $P_2 P_1$ $\tilde{O}(m^\omega d)$

$\rightsquigarrow$ output basis is $\mathbf{s}$-minimal, but not $\mathbf{s}$-Popov (not preserved by multiplication)

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

- with $d_1 = \lfloor d/2 \rfloor$, complexity equation $\mathcal{C}(m, d) = 2\mathcal{C}(m, \lfloor d/2 \rceil) + \tilde{O}(m^\omega d)$
- good: close to worst-case output size $\Theta(m^2 d)$, reached for unbalanced $\mathbf{s}$
- but: output size is $O(md)$ for balanced $\mathbf{s}$ or for $\mathbf{s}$-Popov basis

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

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

→ $\tilde{O}(m^{\omega-1} d)$

M-Padé: computing the Popov basis

observations:

- P_2P_1 is not $\mathbf{s}$-Popov
- computing P_2P_1 is beyond target cost (arbitrary $\mathbf{s} \Rightarrow$ unbalanced degrees)

towards the $\mathbf{s}$-Popov basis: [Jeannerod-Neiger-Schost-Villard 2016]

- pivot degrees $\boldsymbol{\delta}$ of $\mathrm{Popov}_{\mathbf{s}}(P_2P_1)$ are the sum of those of P_2 and P_1
- knowing $\boldsymbol{\delta} \Rightarrow$ fast computation of a $-\boldsymbol{\delta}$-minimal basis R
- for $L \in \mathbb{K}^{m \times m}$ its leading matrix, $L^{-1}R$ is the $\mathbf{s}$-Popov basis

M-Padé: computing the Popov basis

observations:

- P_2P_1 is not $\mathbf{s}$-Popov
- computing P_2P_1 is beyond target cost (arbitrary $\mathbf{s}$ $\Rightarrow$ unbalanced degrees)

towards the $\mathbf{s}$-Popov basis: [Jeannerod-Neiger-Schost-Villard 2016]

- pivot degrees δ of $\mathrm{Popov}_{\mathbf{s}}(P_2P_1)$ are the sum of those of P_2 and P_1
- knowing δ $\Rightarrow$ fast computation of a $-\delta$-minimal basis R
- for $L \in \mathbb{K}^{m\times m}$ its leading matrix, $L^{-1}R$ is the $\mathbf{s}$-Popov basis

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

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

a. $\mathbf{s}$-Popov basis $P_1 \leftarrow$ recursive call at first d_1 points

b. compute residual G and updated degree shift $\mathbf{t}$

c. $\mathbf{t}$-Popov basis $P_2 \leftarrow$ recursive call at remaining points

d. learn degrees: deduce pivot degrees δ of P_2P_1

e. $-\delta$-minimal basis $R \leftarrow$ previous algorithm on $F, M, -\delta$

f. $\mathbf{s}$-Popov basis: return $L^{-1}R$ where $L = \mathrm{lmat}_{-\delta}(R)$

⚠ P_2P_1 is never computed!

M-Padé: computing the Popov basis

observations:

- P_2P_1 is not $\mathbf{s}$-Popov
- computing P_2P_1 is beyond target cost (arbitrary $\mathbf{s} \Rightarrow$ unbalanced degrees)

towards the $\mathbf{s}$-Popov basis: [Jeannerod-Neiger-Schost-Villard 2016]

- pivot degrees δ of $\mathrm{Popov}_{\mathbf{s}}(P_2P_1)$ are the sum of those of P_2 and P_1
- knowing $\delta \Rightarrow$ fast computation of a $-\delta$-minimal basis R
- for $L \in \mathbb{K}^{m\times m}$ its leading matrix, $L^{-1}R$ is the $\mathbf{s}$-Popov basis

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

[out] $\mathbf{s}$-**Popov** basis $P \in \mathbb{K}[x]^{m\times m}$ of solutions with $d_1 = \lfloor d/2 \rfloor$

a. $\mathbf{s}$-Popov basis $P_1 \leftarrow$ recursive call at first d_1 points $\mathcal{C}(m, \lfloor d/2 \rfloor)$

b. compute residual G and updated degree shift $\mathbf{t}$ $\tilde{O}(m^{\omega-1}d)$

c. $\mathbf{t}$-Popov basis $P_2 \leftarrow$ recursive call at remaining points $\mathcal{C}(m, \lceil d/2 \rceil)$

d. learn degrees: deduce pivot degrees δ of P_2P_1 $O(1)$

e. $-\delta$-minimal basis $R \leftarrow$ previous algorithm on $F, M, -\delta$ $\tilde{O}(m^{\omega-1}d)$

f. $\mathbf{s}$-Popov basis: return $L^{-1}R$ where $L = \mathrm{lmat}_{-\delta}(R)$ $O(m^{\omega-1}d)$

⚠ P_2P_1 is never computed! overall $\tilde{O}(m^{\omega-1}d)$

handling unbalanced degrees

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

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

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

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

examples:
- product AB easily deduced from $\bar{A}\bar{B}$
- preservation of the determinant $\det(A) = \det(\bar{A})$
- inverse $\bar{A}^{-1}$ contains A^{-1} as a submatrix
- etc...

⚠ not all types of "average degrees" are supported; this is situation-specific

handling unbalanced degrees

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

[Storjohann 2006] [Zhou-Labahn 2012]

most basic illustration:

- let $u \in \mathbb{K}[x]^{m \times 1}$ of degree $< d$,
- let $A \in \mathbb{K}[x]^{m \times m}$ of degree $< \delta = d/m$,

then the matrix-vector product Au can be computed in

$$MM(m, \delta) + O(m^2 \delta) \subset \tilde{O}(m^{\omega-1} d)$$

cost of "naive" multiplication: $O(m^2 M(d)) \subset \tilde{O}(m^2 d)$

handling unbalanced degrees

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

[Storjohann 2006] [Zhou-Labahn 2012]

most basic illustration:

- let $u \in \mathbb{K}[x]^{m \times 1}$ of degree $< d$,
- let $A \in \mathbb{K}[x]^{m \times m}$ of degree $< \delta = d/m$,

then the matrix-vector product Au can be computed in

$$\mathsf{MM}(m, \delta) + O(m^2 \delta) \ \subset \ \tilde{O}(m^{\omega-1} d)$$

cost of "naive" multiplication: $O(m^2 \mathsf{M}(d)) \subset \tilde{O}(m^2 d)$

algorithm:

$$\begin{bmatrix} & A & \end{bmatrix} \begin{bmatrix} u \end{bmatrix} = \begin{bmatrix} & A & \end{bmatrix} \begin{bmatrix} & \bar{u} & \end{bmatrix} \begin{bmatrix} 1 \\ x^\delta \\ x^{2\delta} \\ \vdots \end{bmatrix}$$

here $\bar{u} \in \mathbb{K}[x]^{m \times m}$ of degree $< \delta$ is the x^δ-expansion:
$u = \bar{u}_{*,0} + \bar{u}_{*,1} x^\delta + \bar{u}_{*,2} x^{2\delta} + \cdots + \bar{u}_{*,m-1} x^{(m-1)\delta}$

handling unbalanced degrees

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

[Storjohann 2006] [Zhou-Labahn 2012]

most basic illustration:

- let $u \in \mathbb{K}[x]^{m \times 1}$ of degree $< d$,
- let $A \in \mathbb{K}[x]^{m \times m}$ of degree $< \delta = d/m$,

then the matrix-vector product Au can be computed in

$$\mathsf{MM}(m, \delta) + O(m^2 \delta) \subset \tilde{O}(m^{\omega-1} d)$$

cost of "naive" multiplication: $O(m^2 \mathsf{M}(d)) \subset \tilde{O}(m^2 d)$

running times of implementation

Polynomial Matrix Library: optimized, low-level C/C++

			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: state of the art

Hermite-Padé: $p \cdot F = 0 \bmod x^d$

- divide and conquer algorithm: from [Beckermann-Labahn 1994]
  supports matrices $F \in \mathbb{K}[x]^{m \times n}$ with $n \geqslant 1$
- [Giorgi-Jeannerod-Villard 2003] achieved $O(m^\omega M(d) \log(d))$
  for $n \in O(m)$, with better incorporation of fast $\mathbb{K}$-linear algebra
- for $s = \mathbf{0}$ and generic F: $\tilde{O}(m^\omega \lceil \frac{nd}{m} \rceil)$ [observation by Lecerf, ca 2001]
- "quasi-optimal" $\tilde{O}(m^{\omega-1} nd)$ [Storjohann 2006] [Zhou-Labahn 2012]
  $\rightsquigarrow$ arbitrary s and F, returns the s-Popov basis [Jeannerod-Neiger-Villard 2020]

univariate relations: state of the art

Hermite-Padé: $p \cdot F = 0 \bmod x^d$

- divide and conquer algorithm: from [Beckermann-Labahn 1994]
  supports matrices $F \in \mathbb{K}[x]^{m \times n}$ with $n \geqslant 1$
- [Giorgi-Jeannerod-Villard 2003] achieved $O(m^\omega M(d) \log(d))$
  for $n \in O(m)$, with better incorporation of fast $\mathbb{K}$-linear algebra
- for $s = \mathbf{0}$ and generic F: $\tilde{O}(m^\omega \lceil \frac{nd}{m} \rceil)$ [observation by Lecerf, ca 2001]
- "quasi-optimal" $\tilde{O}(m^{\omega-1} nd)$ [Storjohann 2006] [Zhou-Labahn 2012]
  $\rightsquigarrow$ arbitrary s and F, returns the s-Popov basis [Jeannerod-Neiger-Villard 2020]

M-Padé: $p \cdot F = 0 \bmod \prod_{1 \leqslant i \leqslant d} (x - \alpha_i)$

- quasi-optimal, arbitrary input, returns s-Popov

[Beckermann-Labahn 1997] [Jeannerod-Neiger-Schost-Villard 2016+2017]

univariate relations: state of the art

Hermite-Padé: $p \cdot F = 0 \bmod x^d$

- divide and conquer algorithm: from [Beckermann-Labahn 1994]
  supports matrices $F \in \mathbb{K}[x]^{m \times n}$ with $n \geqslant 1$
- [Giorgi-Jeannerod-Villard 2003] achieved $O(m^\omega M(d) \log(d))$
  for $n \in O(m)$, with better incorporation of fast $\mathbb{K}$-linear algebra
- for $s = \mathbf{0}$ and generic F: $\tilde{O}(m^\omega \lceil \frac{nd}{m} \rceil)$ [observation by Lecerf, ca 2001]
- "quasi-optimal" $\tilde{O}(m^{\omega-1} nd)$ [Storjohann 2006] [Zhou-Labahn 2012]
  $\rightsquigarrow$ arbitrary s and F, returns the s-Popov basis [Jeannerod-Neiger-Villard 2020]

M-Padé: $p \cdot F = 0 \bmod \prod_{1 \leqslant i \leqslant d} (x - \alpha_i)$

- quasi-optimal, arbitrary input, returns s-Popov

[Beckermann-Labahn 1997] [Jeannerod-Neiger-Schost-Villard 2016+2017]

general relations: $p \cdot F = 0 \bmod B$, for nonsingular $B \in \mathbb{K}[x]^{n \times n}$

- quasi-optimal, arbitrary input, returns s-Popov [Neiger 2016][Neiger-Vu 2017]
- breakthrough: dual problem in similar cost [Rosenkilde-Storjohann 2021]

bi-level structures: an open problem

structured relations

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

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

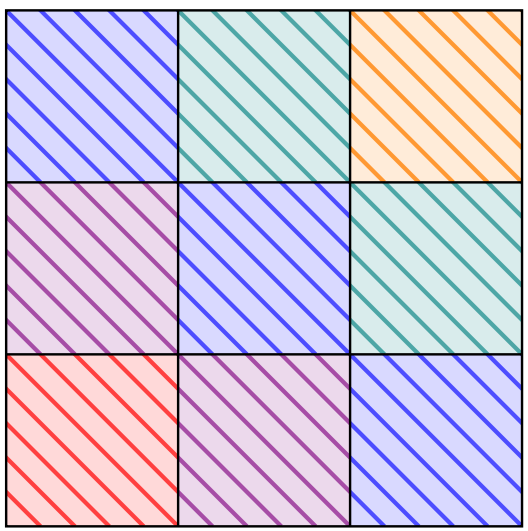
$\downarrow$ structured f_i's

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

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

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

how to leverage this structure?



BTTB: block Toeplitz with Toeplitz blocks

$$\left[\begin{array}{cccc} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_8 \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_8^2 \\ \alpha_1^3 & \alpha_2^3 & \cdots & \alpha_8^3 \\ \alpha_1^4 & \alpha_2^4 & \cdots & \alpha_8^4 \\ \hline \beta_1 & \beta_2 & \cdots & \beta_8 \\ \alpha_1\beta_1 & \alpha_2\beta_2 & \cdots & \alpha_8\beta_8 \\ \alpha_1^2\beta_1 & \alpha_2^2\beta_2 & \cdots & \alpha_8^2\beta_8 \\ \hline \beta_1^2 & \beta_2^2 & \cdots & \beta_8^2 \end{array}\right]$$

bi-level Vandermonde

outline

first guess, then prove

- linearly recurrent sequences
- guessing algebraicity or D-finiteness
- Hermite-Padé approximants

polynomials and matrices

- algebraic computations
- asymptotic complexity bounds
- basics of polynomial matrices

guessing univariate relations

- vector approximation/interpolation
- fast divide and conquer scheme
- unbalanced degrees and Popov bases

software, algorithms, impacts

outline

first guess, then prove

- linearly recurrent sequences
- guessing algebraicity or D-finiteness
- Hermite-Padé approximants

polynomials and matrices

- algebraic computations
- asymptotic complexity bounds
- basics of polynomial matrices

guessing univariate relations

- vector approximation/interpolation
- fast divide and conquer scheme
- unbalanced degrees and Popov bases

software, algorithms, impacts

- software implementation
- kernel bases and Hermite forms
- some uses, from Krylov to Gröbner

open-source mathematics software system

Python/Cython

goals: **complete, robust, available**

and, ultimately, **performance**
via interfaces to FLINT, LinBox, NTL, . . .

high-performance matrix/polynomial libraries

FLINT – LinBox – NTL *C/C++*

goal: **fully optimized core operations**

careful implementations of the best algorithms

software development for polynomial matrices

```
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

goals: **complete, robust, available**

and, ultimately, **performance**
via interfaces to FLINT, LinBox, NTL, . . .

high-performance matrix/polynomial libraries

FLINT – LinBox – NTL *C/C++*

goal: **fully optimized core operations**

careful implementations of the best algorithms

software development for polynomial matrices

Polynomial Matrix Library *C/C++*

561 files, 95k lines of code, including 25k lines of comments
`https://github.com/vneiger/pml`
[Hyun-Neiger-Schost '19]

- NTL-based version: stable, but no new features
- current development in synergy with FLINT
- **welcome** comments, suggestions, contributions

"hey, this doesn't work!"
"yo, plans for implementing this?"
"how to compute bivariate resultants with PML?"

wide range of algorithms

efficiency = state of the art

kernel, high-order lifting,
system solving, reduced form. . .

polynomial matrix multiplication

most fundamental operation ⇒ must be thoroughly optimized

Schönhage's rule 2: Do not waste a factor of two!

various algorithms + use of thresholds:

- small size or small degree: Karatsuba, Strassen, Waksman, ...
- medium degrees: Vandermonde evaluation-interpolation
- medium size and large degrees: FFT evaluation-interpolation

(multimodular computations with several FFT primes)

- large size and large degrees: special evaluation-interpolation

(using points in geometric progression)

$\mathfrak{m}$	$\mathfrak{d}$	PML	FLINT	Waksman	geometric	Vandermonde	Vandermonde2
2	20000	8.09e-03	1.05e-02	**8.34e-03**	3.97e-02	∞	∞
8	20000	3.62e-01	5.80e-01	**3.50e-01**	6.23e-01	∞	∞
32	20000	1.03e+01	3.72e+01	2.01e+01	**1.05e+01**	∞	∞
32	1000	3.91e-01	1.49e+00	7.85e-01	**4.04e-01**	∞	∞
64	1000	1.71e+00	1.20e+01	6.28e+00	**1.66e+00**	∞	∞
128	1000	8.21e+00	4.04e+01	5.17e+01	**8.00e+00**	∞	∞
256	10	2.06e-01	3.32e-01	2.67e+00	2.40e-01	2.29e-01	**2.00e-01**
512	10	1.27e+00	1.73e+00	2.16e+01	1.26e+00	1.35e+00	**1.08e+00**
1024	10	7.34e+00	8.94e+00	1.78e+02	6.54e+00	6.95e+00	**6.90e+00**

+ middle product versions + precomputation structures for repeated products

fraction reconstruction and basis reduction

[Giorgi-Jeannerod-Villard 2003, Algo. PM-Basis]

reconstruct a matrix of degree $< d$ as a fraction with degrees $\leqslant d/2$

- i.e. Padé approximation for $m \times m$ matrices: $F = P^{-1}Q \bmod x^d$
- interpolation variant also implemented, modulo $\prod_{1 \leqslant i \leqslant d}(x - \alpha_i)$
- interpolation often slightly faster and can often serve same purpose

$\rightsquigarrow$ block Wiedemann, matrix Berlekamp-Massey, modular composition, bivariate resultants, ...

m	n	d	approx	interp	d	approx	interp	int-geom
8	4	32	4.31e-4	**3.54e-4**	32768	**4.36**	20.7	**4.38**
32	16	32	9.41e-3	**6.47e-3**	4096	**6.91**	17.0	**6.18**
128	64	32	0.333	**0.229**	1024	31.9	41.7	**25.7**
256	128	32	2.49	**1.46**	256	33.3	28.1	**24.2**

fraction reconstruction and basis reduction

[Giorgi-Jeannerod-Villard 2003, Algo. PM-Basis]

reconstruct a matrix of degree $< d$ as a fraction with degrees $\leqslant d/2$

- i.e. Padé approximation for $m \times m$ matrices: $F = P^{-1}Q \bmod x^d$
- interpolation variant also implemented, modulo $\prod_{1 \leqslant i \leqslant d}(x - \alpha_i)$
- interpolation often slightly faster and can often serve same purpose

$\rightsquigarrow$ block Wiedemann, matrix Berlekamp-Massey, modular composition, bivariate resultants, ...

m	n	d	approx	interp	d	approx	interp	int-geom
8	4	32	4.31e-4	**3.54e-4**	32768	**4.36**	20.7	**4.38**
32	16	32	9.41e-3	**6.47e-3**	4096	**6.91**	17.0	**6.18**
128	64	32	0.333	**0.229**	1024	31.9	41.7	**25.7**
256	128	32	2.49	**1.46**	256	33.3	28.1	**24.2**

basis reduction: find a minimal form R of a matrix $A \in \mathbb{K}[x]^{m \times m}$

expansion of A^{-1} satisfies, ultimately, a matrix recurrence of order $\leqslant d$

$$A^{-1} = F_0 + F_1 x + \cdots + F_{md} x^{md} + \cdots + F_{md+2d-1} x^{md+2d-1} \bmod x^{md+2d}$$

find F by high-order lifting [Storjohann 2003] and reconstruct $F = R^{-1}Q \bmod x^{2d}$

$\rightsquigarrow$ core ingredient in transformation to Popov form [Gupta-Sarkar-Storjohann-Valeriote '11+'12]

m	d	Newton	high-order	reconstruct	total
4	24574	**1.251**	1.688	8.772	10.02
16	1534	4.457	**3.044**	8.506	11.55
64	94	30.62	**5.509**	5.833	11.34
128	94	371.1	**29.17**	37.23	66.41

linear system solving over $\mathbb{F}_p[x]$

- Dixon's solver is often the most efficient [Dixon 1982] $\tilde{O}(m^3 d)$
- solver by minimal kernel basis is not far behind, and more general $\tilde{O}(m^\omega d)$
- solver by high-order lifting seems slower $\tilde{O}(m^\omega d)$

m	d	Dixon	kernel	high-order lifting
16	1024	**0.695**	1.96	2.39
32	1024	**2.88**	8.06	13.8
128	512	**37.2**	84.2	266

linear system solving over $\mathbb{F}_p[x]$

- Dixon's solver is often the most efficient [Dixon 1982] $\tilde{O}(m^3 d)$
- solver by minimal kernel basis is not far behind, and more general $\tilde{O}(m^\omega d)$
- solver by high-order lifting seems slower $\tilde{O}(m^\omega d)$

m	d	Dixon	kernel	high-order lifting
16	1024	**0.695**	1.96	2.39
32	1024	**2.88**	8.06	13.8
128	512	**37.2**	84.2	266

determinant

- expansion by minors for small dimensions ($m \leqslant 4$ or 5) $\tilde{O}(d)$
- evaluation/interpolation at md points in geometric progression $\tilde{O}(m^{\omega+1} d)$
- solving a linear system with random right-hand side [Pan, 1988] $\tilde{O}(m^\omega d)$
- triangularization via minimal kernel basis [Labahn-Neiger-Zhou, 2017] $\tilde{O}(m^\omega d)$

m	d	minors	evaluation	linsolve	triangular
4	65536	**0.673**	1.90	5.78	**0.686**
16	4096	∞	**3.75**	**3.52**	6.12
32	4096	∞	26.5	**15.3**	32.4
64	2048	∞	109	**35.9**	71.0
128	512	∞	∞	**40.7**	71.8

various problems and reductions

reductions of most problems to polynomial matrix multiplication

matrix $m \times m$ of degree d $\rightarrow$ $\tilde{O}(m^\omega d)$

of "average" degree $\frac{D}{m}$ $\rightarrow$ $\tilde{O}(m^\omega \frac{D}{m})$

classical matrix operations

- multiplication
- kernel, system solving
- rank, determinant
- inversion $\tilde{O}(m^3 d)$

univariate specific operations

- truncated inverse, QuoRem
- vector M-Padé
- simultaneous M-Padé
- syzygies / univariate relations

transformation to **normal forms**

- triangularization: Hermite form
- row reduction: Popov form
- diagonalization: Smith form

various problems and reductions

reductions of most problems to polynomial matrix multiplication

matrix $m \times m$ of degree d $\rightarrow$ $\tilde{O}(m^\omega d)$

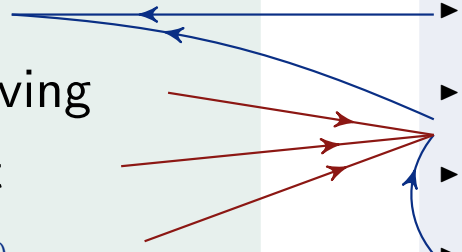
of "average" degree $\frac{D}{m}$ $\rightarrow$ $\tilde{O}(m^\omega \frac{D}{m})$

classical matrix operations

- multiplication
- kernel, system solving
- rank, determinant
- inversion $\tilde{O}(m^3 d)$

univariate specific operations

- truncated inverse, QuoRem
- vector M-Padé
- simultaneous M-Padé
- syzygies / univariate relations



transformation to **normal forms**

- triangularization: Hermite form
- row reduction: Popov form
- diagonalization: Smith form

various problems and reductions

reductions of most problems to polynomial matrix multiplication

matrix $m \times m$ of degree d $\rightarrow$ $\tilde{O}(m^\omega d)$

of "average" degree $\frac{D}{m}$ $\rightarrow$ $\tilde{O}(m^\omega \frac{D}{m})$

classical matrix operations

- multiplication
- kernel, system solving
- rank, determinant
- inversion $\tilde{O}(m^3 d)$

univariate specific operations

- truncated inverse, QuoRem
- vector M-Padé
- simultaneous M-Padé
- syzygies / univariate relations

transformation to **normal forms**

- triangularization: Hermite form
- row reduction: Popov form
- diagonalization: Smith form

(polynomial) kernel bases

for $F \in \mathbb{K}[x]^{m \times n}$, its left kernel is

$$\mathcal{K}(F) = \{p \in \mathbb{K}[x]^{1 \times m} \mid pF = 0\}$$

- $\mathcal{K}(F)$ is a $\mathbb{K}[x]$-module
- it has rank $m - r$, where r is the rank of F

$\Rightarrow$ basis $K \in \mathbb{K}[x]^{(m-r) \times m}$

(polynomial) kernel bases

for $F \in \mathbb{K}[x]^{m \times n}$, its left kernel is

$$\mathcal{K}(F) = \{p \in \mathbb{K}[x]^{1 \times m} \mid pF = 0\}$$

- $\mathcal{K}(F)$ is a $\mathbb{K}[x]$-module
- it has rank $m - r$, where r is the rank of F

$\Rightarrow$ basis $K \in \mathbb{K}[x]^{(m-r) \times m}$

kernel basis for a constant matrix $\rightarrow$ usual nullspace

kernel basis K

$$\begin{bmatrix} 5 & 6 & 1 & 0 & 0 \\ 0 & 5 & 0 & 1 & 0 \\ 0 & 0 & 3 & 2 & 1 \end{bmatrix}$$

input matrix F

$$\begin{bmatrix} 5 & 6 \\ 6 & 1 \\ 2 & 6 \\ 5 & 2 \\ 5 & 6 \end{bmatrix}$$

(polynomial) kernel bases

for $F \in \mathbb{K}[x]^{m \times n}$, its left kernel is

$$\mathcal{K}(F) = \{p \in \mathbb{K}[x]^{1 \times m} \mid pF = 0\}$$

- $\mathcal{K}(F)$ is a $\mathbb{K}[x]$-module
- it has rank $m - r$, where r is the rank of F

$\Rightarrow$ basis $K \in \mathbb{K}[x]^{(m-r) \times m}$

a small-degree kernel basis, for any $R \in \mathbb{K}[x] \setminus \{0\}$

kernel basis K

$$\begin{bmatrix} 1+x & -1 & & \\ 0 & x & -1 & \\ 0 & 0 & x & -1 \end{bmatrix}$$

input matrix F

$$\begin{bmatrix} R \\ R + xR \\ xR + x^2R \\ x^2R + x^3R \end{bmatrix}$$

(polynomial) kernel bases

for $F \in \mathbb{K}[x]^{m \times n}$, its left kernel is

$$\mathcal{K}(F) = \{p \in \mathbb{K}[x]^{1 \times m} \mid pF = 0\}$$

- $\mathcal{K}(F)$ is a $\mathbb{K}[x]$-module
- it has rank $m - r$, where r is the rank of F

$\Rightarrow$ basis $K \in \mathbb{K}[x]^{(m-r) \times m}$

a simple case of kernel basis computation

kernel basis K

$$\begin{bmatrix} 1 & 0 & 0 & -f_{00} & -f_{01} \\ 0 & 1 & 0 & -f_{10} & -f_{11} \\ 0 & 0 & 1 & -f_{20} & -f_{21} \end{bmatrix}$$

input matrix F

$$\begin{bmatrix} f_{00} & f_{01} \\ f_{10} & f_{11} \\ f_{20} & f_{21} \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

(polynomial) kernel bases

for $F \in \mathbb{K}[x]^{m \times n}$, its left kernel is

$$\mathcal{K}(F) = \{p \in \mathbb{K}[x]^{1 \times m} \mid pF = 0\}$$

- $\mathcal{K}(F)$ is a $\mathbb{K}[x]$-module
- it has rank $m - r$, where r is the rank of F

$\Rightarrow$ basis $K \in \mathbb{K}[x]^{(m-r) \times m}$

simple case made less simple, with $\det(H) \neq 0$

kernel basis K	input matrix F
$\ldots$ is left multiple of $\begin{bmatrix} I_m & -GH^{-1} \end{bmatrix}$	$\begin{bmatrix} G \\ H \end{bmatrix} \in \mathbb{K}[x]^{(m+n) \times n}$
$\ldots$ $\det(H) \begin{bmatrix} I_m & -GH^{-1} \end{bmatrix}$ is left multiple of it	

(polynomial) kernel bases

for $F \in \mathbb{K}[x]^{m \times n}$, its left kernel is

$$\mathcal{K}(F) = \{p \in \mathbb{K}[x]^{1 \times m} \mid pF = 0\}$$

- $\mathcal{K}(F)$ is a $\mathbb{K}[x]$-module
- it has rank $m - r$, where r is the rank of F

$\Rightarrow$ basis $K \in \mathbb{K}[x]^{(m-r) \times m}$

inclusion $\mathcal{K}(F) \subset \mathcal{I}(M, F) = \{p \in \mathbb{K}[x]^{1 \times m} \mid pF = 0 \bmod M\}$

$\Rightarrow$ recover kernel via M-Padé for a suitable choice of modulus M

kernel algorithm: M-Padé at large order

[in] matrix $F \in \mathbb{K}[x]^{m \times n}$ of degree $\leqslant d$
[in] $\delta \in \mathbb{Z}_{>0}$ such that there exists a basis of $\mathcal{K}(F)$ of degree $\leqslant \delta$
[out] minimal basis K of $\mathcal{K}(F)$

1. $\Delta \leftarrow \delta + d + 1$
2. $\boldsymbol{\alpha} \leftarrow$ choose some $(\alpha_1, \ldots, \alpha_\Delta)$ in $\mathbb{K}^\Delta$ (not necessarily distinct)
3. $P \in \mathbb{K}[x]^{m \times m} \leftarrow$ minimal basis of $\mathcal{I}(\boldsymbol{\alpha}, F)$
4. $K \in \mathbb{K}[x]^{k \times m} \leftarrow$ rows of P which have degree $\leqslant \delta$

kernel algorithm: M-Padé at large order

[in] matrix $F \in \mathbb{K}[x]^{m \times n}$ of degree $\leqslant d$
[in] $\delta \in \mathbb{Z}_{>0}$ such that there exists a basis of $\mathcal{K}(F)$ of degree $\leqslant \delta$
[out] minimal basis K of $\mathcal{K}(F)$

1. $\Delta \leftarrow \delta + d + 1$
2. $\boldsymbol{\alpha} \leftarrow$ choose some $(\alpha_1, \ldots, \alpha_\Delta)$ in $\mathbb{K}^\Delta$ (not necessarily distinct)
3. $P \in \mathbb{K}[x]^{m \times m} \leftarrow$ minimal basis of $\mathcal{I}(\boldsymbol{\alpha}, F)$
4. $K \in \mathbb{K}[x]^{k \times m} \leftarrow$ rows of P which have degree $\leqslant \delta$

$\Rightarrow$ complexity $O(m^\omega M(\lceil \frac{n\Delta}{m} \rceil) \log(\lceil \frac{n\Delta}{m} \rceil))$

. . . but what order Δ or degree bound δ may we use?

kernel algorithm: M-Padé at large order

[in] matrix $F \in \mathbb{K}[x]^{m \times n}$ of degree $\leqslant d$
[in] $\delta \in \mathbb{Z}_{>0}$ such that there exists a basis of $\mathcal{K}(F)$ of degree $\leqslant \delta$
[out] minimal basis K of $\mathcal{K}(F)$

1. $\Delta \leftarrow \delta + d + 1$
2. $\boldsymbol{\alpha} \leftarrow$ choose some $(\alpha_1, \ldots, \alpha_\Delta)$ in $\mathbb{K}^\Delta$ (not necessarily distinct)
3. $P \in \mathbb{K}[x]^{m \times m} \leftarrow$ minimal basis of $\mathcal{I}(\boldsymbol{\alpha}, F)$
4. $K \in \mathbb{K}[x]^{k \times m} \leftarrow$ rows of P which have degree $\leqslant \delta$

$\Rightarrow$ complexity $O(m^\omega M(\lceil \frac{n\Delta}{m} \rceil) \log(\lceil \frac{n\Delta}{m} \rceil))$

. . . but what order Δ or degree bound δ may we use?

a specific bound may be known from the context (e.g. gcd or "row bases")

- general bound: $\delta = nd$
- yields complexity $\tilde{O}(m^{\omega-1} n^2 d)$

factor n from "quasi-optimal"; generalizes to s-minimal

bonus: proof of minimal basis

knowing $\delta \in \mathbb{Z}_{>0}$ such that there exists a basis of $\mathcal{K}(\mathrm{F})$ of degree $\leqslant \delta$

1. $\Delta \leftarrow \delta + \mathrm{d} + 1$
2. $\boldsymbol{\alpha} \leftarrow$ choose some $(\alpha_1, \ldots, \alpha_\Delta)$ in $\mathbb{K}^\Delta$ (not necessarily distinct)
3. $\mathrm{P} \in \mathbb{K}[\chi]^{\mathrm{m} \times \mathrm{m}} \leftarrow$ minimal basis of $\mathcal{I}(\boldsymbol{\alpha}, \mathrm{F})$
4. $\mathrm{K} \in \mathbb{K}[\chi]^{\mathrm{k} \times \mathrm{m}} \leftarrow$ rows of P which have degree $\leqslant \delta$

$\Rightarrow$ K is a minimal basis of $\mathcal{K}(\mathrm{F})$

bonus: proof of minimal basis

knowing $\delta \in \mathbb{Z}_{>0}$ such that there exists a basis of $\mathcal{K}(F)$ of degree $\leqslant \delta$

1. $\Delta \leftarrow \delta + d + 1$
2. $\boldsymbol{\alpha} \leftarrow$ choose some $(\alpha_1, \ldots, \alpha_\Delta)$ in $\mathbb{K}^\Delta$ (not necessarily distinct)
3. $P \in \mathbb{K}[x]^{m \times m} \leftarrow$ minimal basis of $\mathcal{I}(\boldsymbol{\alpha}, F)$
4. $K \in \mathbb{K}[x]^{k \times m} \leftarrow$ rows of P which have degree $\leqslant \delta$

$\Rightarrow$ K is a minimal basis of $\mathcal{K}(F)$

proof:

$\Rightarrow$ K is minimal by construction

. K satisfies $KF = 0 \bmod (x - \alpha_1) \cdots (x - \alpha_\Delta)$
. and $\deg(K) \leqslant \delta$, hence $\deg(KF) \leqslant \delta + d < \Delta$
$\Rightarrow KF = 0$, i.e. the rows of K are in $\mathcal{K}(F)$

. let $B \in \mathbb{K}[x]^{(m-r) \times m}$ be a basis of $\mathcal{K}(F)$ of degree $\leqslant \delta$
. then $B = UP$ for some U
. by the predictable degree property, in fact $B = VK$
$\Rightarrow$ any vector in $\mathcal{K}(F)$ is generated by K

bonus: more details on degree bounds

knowing $\delta \in \mathbb{Z}_{>0}$ such that there exists a basis of $\mathcal{K}(F)$ of degree $\leqslant \delta$

what degree bound δ may we use?

a specific bound may be known from the context — e.g. gcd, "row bases"

- a general bound is $\delta = nd$
- yields complexity $\tilde{O}(m^{\omega\lceil \frac{n^2 d}{m} \rceil})$

bonus: more details on degree bounds

knowing $\delta \in \mathbb{Z}_{>0}$ such that there exists a basis of $\mathcal{K}(F)$ of degree $\leqslant \delta$

what degree bound δ may we use?

a specific bound may be known from the context — e.g. gcd, "row bases"

- a general bound is $\delta = nd$
- yields complexity $\tilde{O}(m^\omega \lceil \frac{n^2 d}{m} \rceil)$

proof:
complexity $\tilde{O}(m^\omega \lceil \frac{n\Delta}{m} \rceil)$
with $\Delta = \delta + d + 1 = (n+1)d + 1$

bonus: more details on degree bounds

knowing $\delta \in \mathbb{Z}_{>0}$ such that there exists a basis of $\mathcal{K}(F)$ of degree $\leqslant \delta$

what degree bound δ may we use?

a specific bound may be known from the context — e.g. gcd, "row bases"

- a general bound is $\delta = nd$
- yields complexity $\tilde{O}(m^{\omega}\lceil \frac{n^2 d}{m} \rceil)$

proof:

up to row and column permutation, $F = \left[\begin{smallmatrix} G & * \\ H & * \end{smallmatrix}\right]$
with $G \in \mathbb{K}[x]^{r \times r}$ nonsingular
then, $\mathcal{K}(F) = \mathcal{K}(\left[\begin{smallmatrix} G \\ H \end{smallmatrix}\right])$

the matrix $[-H(\det(G)G^{-1}) \quad \det(G)I_{m-r}]$ has polynomial entries,
it has rank $m - r$ and its rows are in $\mathcal{K}(F)$,
it has degree $\leqslant \max(\deg(\det(G)), \deg(H) + (r-1)\deg(G)) \leqslant rd$

by degree minimality of reduced matrices,
any minimal basis of $\mathcal{K}(F)$ must have degree $\leqslant rd$

kernel algorithm: Zhou-Labahn-Storjohann

breakthrough [Zhou-Labahn-Storjohann 2012]

- complexity $\tilde{O}(m^\omega \lceil \frac{n d}{m} \rceil)$ without assumption
- computes s-minimal basis of $\mathcal{K}(F)$ for $s = 0$ or $s = \mathrm{rdeg}(F)$

- n large: divide and conquer on n, via residual + basis multiplication

$\rightsquigarrow$ partial linearization for multiplication with weakly unbalanced degrees

- n small: use fast M-Padé approximation/interpolation

$\rightsquigarrow$ well-chosen d yields at least half the kernel efficiently

kernel algorithm: Zhou-Labahn-Storjohann

breakthrough [Zhou-Labahn-Storjohann 2012]

- complexity $\tilde{O}(m^\omega \lceil \frac{n\,d}{m} \rceil)$ without assumption
- computes $\mathbf{s}$-minimal basis of $\mathcal{K}(F)$ for $\mathbf{s} = 0$ or $\mathbf{s} = \mathrm{rdeg}(F)$

- n large: divide and conquer on n, via residual + basis multiplication

$\rightsquigarrow$ partial linearization for multiplication with weakly unbalanced degrees

- n small: use fast M-Padé approximation/interpolation

$\rightsquigarrow$ well-chosen d yields at least half the kernel efficiently

if $n > \frac{m}{2}$:

$K_1 \leftarrow$ recursive call on first $\frac{n}{2}$ columns of F, and shift $\mathbf{s}$

$G \leftarrow$ multiply $K_1 \cdot F_{*,\frac{n}{2}..n}$ (last $\frac{n}{2}$ columns of F)

$K_2 \leftarrow$ recursive call on G, and shift $\mathbf{t} = \mathrm{rdeg}_{\mathbf{s}}(K_1)$

return $K_2 K_1$

kernel algorithm: Zhou-Labahn-Storjohann

breakthrough [Zhou-Labahn-Storjohann 2012]

- complexity $\tilde{O}(m^\omega \lceil \frac{n\,d}{m} \rceil)$ without assumption
- computes s-minimal basis of $\mathcal{K}(F)$ for $s = 0$ or $s = \mathrm{rdeg}(F)$

- n large: divide and conquer on n, via residual + basis multiplication

$\rightsquigarrow$ partial linearization for multiplication with weakly unbalanced degrees

- n small: use fast M-Padé approximation/interpolation

$\rightsquigarrow$ well-chosen d yields at least half the kernel efficiently

if $n \leqslant \frac{m}{2}$:

$\delta \leftarrow 2d$ ($\approx$ degree of kernel basis expected for generic F)

$\Delta \leftarrow \delta + d + 1$ and take some $\boldsymbol{\alpha} \leftarrow (\alpha_1, \ldots, \alpha_\Delta)$ in $\mathbb{K}^\Delta$

$P \in \mathbb{K}[x]^{m \times m} \leftarrow s$-reduced basis of $\mathcal{I}(\boldsymbol{\alpha}, F)$

$K_1, Q \leftarrow$ rows of P which are in $\mathcal{K}(F)$ / which are not in $\mathcal{K}(F)$

$K_2 \leftarrow$ recursive call on $\frac{QF}{(x-\alpha_1)\cdots(x-\alpha_\Delta)}$, return $\left[\begin{smallmatrix} K_1 \\ K_2 Q \end{smallmatrix}\right]$

kernel algorithm: Zhou-Labahn-Storjohann

breakthrough [Zhou-Labahn-Storjohann 2012]

- complexity $\tilde{O}(m^\omega \lceil \frac{n\,d}{m} \rceil)$ without assumption
- computes s-minimal basis of $\mathcal{K}(F)$ for $s = 0$ or $s = \mathrm{rdeg}(F)$

- n large: divide and conquer on n, via residual + basis multiplication

$\rightsquigarrow$ partial linearization for multiplication with weakly unbalanced degrees

- n small: use fast M-Padé approximation/interpolation

$\rightsquigarrow$ well-chosen d yields at least half the kernel efficiently

rules of thumb, generically:
"quantity of information is preserved"
+
"degrees in minimal basis are uniform"

$\rightsquigarrow$ $(m-r)m\deg(K) \approx mnd$, which means $\deg(K) \approx \frac{n}{m-r}d$

example: if F is $m \times \frac{m}{2}$, generically $\deg(K) = d$

$\Rightarrow \Delta = 2d+1$ and quasi-optimal complexity $\tilde{O}(m^\omega d)$

kernel algorithm: Zhou-Labahn-Storjohann

breakthrough [Zhou-Labahn-Storjohann 2012]

- complexity $\tilde{O}(m^\omega \lceil \frac{n\,d}{m} \rceil)$ without assumption
- computes s-minimal basis of $\mathcal{K}(F)$ for $s = 0$ or $s = \mathrm{rdeg}(F)$

- n large: divide and conquer on n, via residual + basis multiplication

$\rightsquigarrow$ partial linearization for multiplication with weakly unbalanced degrees

- n small: use fast M-Padé approximation/interpolation

$\rightsquigarrow$ well-chosen d yields at least half the kernel efficiently

running times of an optimized implementation (PML)

			M-Padé at large order		[ZLS 2012]	
m	n	d	approx.	interp.	approx.	interp.
8	4	8192	7.22	6.60	**2.16**	**2.49**
8	7	8192	14.1	14.4	**4.64**	5.63
32	16	1024	86.3	63.1	**3.75**	**3.51**
32	31	1024	142	118	**8.27**	**8.09**
128	64	256	2720	1827	16.8	**11.8**
128	127	256	>1h	>1h	43.8	**35.6**
16	1	512	**5.68**	**5.31**	11.5	11.2

kernel algorithm: Zhou-Labahn-Storjohann

breakthrough [Zhou-Labahn-Storjohann 2012]

- complexity $\tilde{O}(m^\omega \lceil \frac{n d}{m} \rceil)$ without assumption
- computes s-minimal basis of $\mathcal{K}(F)$ for $s = 0$ or $s = \mathrm{rdeg}(F)$

- n large: divide and conquer on n, via residual + basis multiplication

$\rightsquigarrow$ partial linearization for multiplication with weakly unbalanced degrees

- n small: use fast M-Padé approximation/interpolation

$\rightsquigarrow$ well-chosen d yields at least half the kernel efficiently

Schönhage's rule 4: Do not raise the overall cost for speeding up rare cases—avoid lotteries!

. . . but do seek features of the most common cases that might help you lower the overall cost

consequence: linear system solving

linear system solving:

[in] $A \in \mathbb{K}[x]^{m \times m}$ nonsingular and $v \in \mathbb{K}[x]^{1 \times m}$

[out] $u \in \mathbb{K}[x]^{1 \times m}$ and $g \in \mathbb{K}[x]$ such that:

$$uA = gv \quad \text{and} \quad g \text{ has minimal degree}$$

- the equation has a solution: $u = gvA^{-1}$ with $g = \det(A)$
- but there is often no polynomial solution with $g = 1$
- $\det(A)A^{-1}$, and therefore u, can have degree $\approx m \deg(A)$

consequence: linear system solving

linear system solving:

[in] $A \in \mathbb{K}[x]^{m \times m}$ nonsingular and $v \in \mathbb{K}[x]^{1 \times m}$

[out] $u \in \mathbb{K}[x]^{1 \times m}$ and $g \in \mathbb{K}[x]$ such that:

$$uA = gv \quad \text{and} \quad g \text{ has minimal degree}$$

- the equation has a solution: $u = gvA^{-1}$ with $g = \det(A)$
- but there is often no polynomial solution with $g = 1$
- $\det(A)A^{-1}$, and therefore u, can have degree $\approx m \deg(A)$

$$[u \quad g] \in \mathbb{K}[x]^{1 \times (m+1)} \leftarrow \text{minimal kernel basis of } F = \begin{bmatrix} A \\ -v \end{bmatrix} \in \mathbb{K}[x]^{(m+1) \times m}$$

- using the shift $s = \text{rdeg}(F) = (\text{rdeg}(A), \deg(v))$
- complexity $\tilde{O}(m^\omega \max(\deg(A), \deg(v)))$
- $uA = gv$ with $\deg(g)$ minimal

→ in fact: $\max(\deg(A), \frac{\deg(v)}{m})$

matrix GCDs / row bases

a row basis of a matrix $A \in \mathbb{K}[x]^{m \times n}$
is a basis of its $\mathbb{K}[x]$-row span $\longrightarrow \{pA \mid p \in \mathbb{K}[x]^{1 \times m}\}$

$\rightsquigarrow$ represented as $R \in \mathbb{K}[x]^{r \times n}$, where r is the rank of A

$\rightsquigarrow$ $A = UR$ for some $U \in \mathbb{K}[x]^{m \times r}$

matrix GCDs / row bases

a row basis of a matrix $A \in \mathbb{K}[x]^{m \times n}$
is a basis of its $\mathbb{K}[x]$-row span $\longrightarrow \{pA \mid p \in \mathbb{K}[x]^{1 \times m}\}$

$\rightsquigarrow$ represented as $R \in \mathbb{K}[x]^{r \times n}$, where r is the rank of A
$\rightsquigarrow$ $A = UR$ for some $U \in \mathbb{K}[x]^{m \times r}$

examples:

- row basis for $A \in \mathbb{K}[x]^{m \times m}$ nonsingular?
- row basis of $\begin{bmatrix} a \\ b \end{bmatrix}$ for a, b two polynomials?
- $A \in \mathbb{K}[x]^{(m-r) \times m}$ a kernel basis of $F \in \mathbb{K}[x]^{m \times n}$

row basis of A? column basis of A?

matrix GCDs / row bases

a row basis of a matrix $A \in \mathbb{K}[x]^{m \times n}$
is a basis of its $\mathbb{K}[x]$-row span $\longrightarrow \{pA \mid p \in \mathbb{K}[x]^{1 \times m}\}$

$\rightsquigarrow$ represented as $R \in \mathbb{K}[x]^{r \times n}$, where r is the rank of A
$\rightsquigarrow$ $A = UR$ for some $U \in \mathbb{K}[x]^{m \times r}$

examples:

- row basis for $A \in \mathbb{K}[x]^{m \times m}$ nonsingular? $R = A$
- row basis of $\begin{bmatrix} a \\ b \end{bmatrix}$ for a, b two polynomials?
- $A \in \mathbb{K}[x]^{(m-r) \times m}$ a kernel basis of $F \in \mathbb{K}[x]^{m \times n}$

row basis of A? column basis of A?

matrix GCDs / row bases

a row basis of a matrix $A \in \mathbb{K}[x]^{m \times n}$
is a basis of its $\mathbb{K}[x]$-row span $\longrightarrow \{pA \mid p \in \mathbb{K}[x]^{1 \times m}\}$

$\rightsquigarrow$ represented as $R \in \mathbb{K}[x]^{r \times n}$, where r is the rank of A
$\rightsquigarrow$ $A = UR$ for some $U \in \mathbb{K}[x]^{m \times r}$

examples:

- row basis for $A \in \mathbb{K}[x]^{m \times m}$ nonsingular? $R = A$
- row basis of $\begin{bmatrix} a \\ b \end{bmatrix}$ for a, b two polynomials? $R = [\gcd(a, b)]$
- $A \in \mathbb{K}[x]^{(m-r) \times m}$ a kernel basis of $F \in \mathbb{K}[x]^{m \times n}$

row basis of A? column basis of A?

matrix GCDs / row bases

a row basis of a matrix $A \in \mathbb{K}[x]^{m \times n}$
is a basis of its $\mathbb{K}[x]$-row span $\longrightarrow \{pA \mid p \in \mathbb{K}[x]^{1 \times m}\}$

$\rightsquigarrow$ represented as $R \in \mathbb{K}[x]^{r \times n}$, where r is the rank of A
$\rightsquigarrow$ $A = UR$ for some $U \in \mathbb{K}[x]^{m \times r}$

examples:

- row basis for $A \in \mathbb{K}[x]^{m \times m}$ nonsingular? $R = A$
- row basis of $\begin{bmatrix} a \\ b \end{bmatrix}$ for a, b two polynomials? $R = [\gcd(a, b)]$
- $A \in \mathbb{K}[x]^{(m-r) \times m}$ a kernel basis of $F \in \mathbb{K}[x]^{m \times n}$
row basis of A? column basis of A? $R = A$ and $C = I_{m-r}$

proof:
A has full rank so C is $(m-r) \times (m-r)$ nonsingular
and by definition $A = C\bar{A}$ for some $\bar{A}$
so $AF = 0 \Rightarrow \bar{A}F = 0$, hence $\bar{A} = VA$
from $A = CVA$, with A having full row rank, we deduce $CV = I_{m-r}$

matrix GCDs / row bases

a row basis of a matrix $A \in \mathbb{K}[x]^{m \times n}$
is a basis of its $\mathbb{K}[x]$-row span $\longrightarrow \{pA \mid p \in \mathbb{K}[x]^{1 \times m}\}$

$\rightsquigarrow$ represented as $R \in \mathbb{K}[x]^{r \times n}$, where r is the rank of A
$\rightsquigarrow$ $A = UR$ for some $U \in \mathbb{K}[x]^{m \times r}$

applications:

- find the s-Popov form of a rectangular / rank-deficient matrix A
- check whether a matrix is saturated / unimodularly completable
- triangularization: Hermite normal form and determinant

matrix GCDs / row bases

a row basis of a matrix $A \in \mathbb{K}[x]^{m \times n}$ is a basis of its $\mathbb{K}[x]$-row span $\longrightarrow \{pA \mid p \in \mathbb{K}[x]^{1 \times m}\}$

$\rightsquigarrow$ represented as $R \in \mathbb{K}[x]^{r \times n}$, where r is the rank of A

$\rightsquigarrow$ $A = UR$ for some $U \in \mathbb{K}[x]^{m \times r}$

applications:

- find the s-Popov form of a rectangular / rank-deficient matrix A
- check whether a matrix is saturated / unimodularly completable
- triangularization: Hermite normal form and determinant

algorithm: [Zhou-Labahn, 2013]

- $K \leftarrow$ left kernel basis for A
- $G \leftarrow$ right kernel basis for K
- $R \leftarrow$ matrix such that $A = GR$

complexity $\tilde{O}(mn^{\omega-1}d)$, assuming $m \geqslant n$, where $d = \deg(A)$

fast, deterministic triangularization

[Mulders-Storjohann 2003, Giorgi-Jeannerod-Villard 2003, Zhou 2012]

triangularization of $m \times m$ matrix A using $\frac{m}{2} \times \frac{m}{2}$ blocks

not computed

$$\begin{bmatrix} * & * \\ K_1 & K_2 \end{bmatrix} \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} = \begin{bmatrix} R & * \\ 0 & B \end{bmatrix}$$

kernel basis of $\begin{bmatrix} A_1 \\ A_3 \end{bmatrix}$ $\quad K_1A_2 + K_2A_4$ $\quad$ row basis of $\begin{bmatrix} A_1 \\ A_3 \end{bmatrix}$

main property: $\begin{bmatrix} * & * \\ K_1 & K_2 \end{bmatrix}$ is unimodular

- Hermite form of A = Hermite form of $\begin{bmatrix} R & * \\ 0 & B \end{bmatrix}$
- $\det(A) = \det(R)\det(B)$

Hermite normal form and determinant in $\tilde{O}(m^\omega \deg(A))$

[Zhou, 2012] [Labahn-Neiger-Zhou, 2017]

use 1: faster XGCD and rational reconstruction

gcd
[in] a and b univariate polynomials in $\mathbb{K}[x]_{\leqslant d}$
[out] $g = \gcd(a, b)$

xgcd
[in] a and b univariate polynomials in $\mathbb{K}[x]_{\leqslant d}$
[out] u, v, g with $g = \gcd(a, b) = ua + vb$

ratrecon
[in] a, M univariate polynomials in $\mathbb{K}[x]_{\leqslant d}$
[out] p, q of degree $\leqslant d/2$ with $a = \frac{q}{p} \bmod M$

solved in $\gamma \cdot M(d)\log(d) + O(M(d))$ with small γ, via M-Padé interpolation

[Kevin Tran's PhD thesis, available soon]

$\rightsquigarrow$ gain of a constant factor over half-gcd [Mateer '08] [van der Hoeven '25]
$\rightsquigarrow$ prototype in FLINT confirms improved software performance

Schönhage's rule 2: Do not waste a factor of two!

note: more generally, M-Padé finds $p_1, \ldots, p_m$ of minimal degree such that
$$p_1 a_1 + \cdots + p_m a_m = \gcd(a_1, \ldots, a_m)$$

use 1: faster XGCD and rational reconstruction

gcd

[in] a and b univariate polynomials in $\mathbb{K}[x]_{\leqslant d}$
[out] $g = \gcd(a, b)$

solved in $\gamma M(d)\log(d) + O(M(d))$ with small γ, via M-Padé interpolation

[Kevin Tran's PhD thesis, available soon]

use 1: faster XGCD and rational reconstruction

gcd

[in] a and b univariate polynomials in $\mathbb{K}[x]_{\leqslant d}$
[out] $g = \gcd(a, b)$

solved in $\gamma M(d)\log(d) + O(M(d))$ with small γ, via M-Padé interpolation

[Kevin Tran's PhD thesis, available soon]

notation: $\bar{a} = a/g$ and $\bar{b} = b/g$ $\quad$ $\bar{a}$ and $\bar{b}$ are coprime

lemma: $\begin{bmatrix} -\bar{b} & \bar{a} \end{bmatrix}$ is a minimal kernel basis of $F = \begin{bmatrix} a \\ b \end{bmatrix}$

algorithm: M-Padé at distinct points

- the input F has degree $\leqslant d$
- the sought kernel basis has degree $\leqslant \delta = d$ $\quad$ (or $\delta = d - \deg(h)\ldots$)
- for efficiency: use geometric progression (or FFT points if available)

$$\Rightarrow \begin{cases} 1.\ \text{pick } \Delta = \delta + d + 1 = 2d + 1 \text{ points } \boldsymbol{\alpha} \in \mathbb{K}^{2d+1} & O(1) \\ 2.\ \text{find } \begin{bmatrix} -\bar{b} & \bar{a} \end{bmatrix} \text{ via a minimal basis of } \mathcal{I}(\boldsymbol{\alpha}, F) & \gamma M(\delta)\log(\delta) + \cdots \\ 3.\ \text{deduce } g = a/\bar{a} & O(M(\delta)) \end{cases}$$

use 2: faster linear algebra (Krylov iterates)

[in] $A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{bmatrix} \in \mathbb{K}^{n \times n}$

[in] $u = \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix} \in \mathbb{K}^{1 \times n}$

[in] integer $d > 0$

[out] $\begin{bmatrix} u \\ uA \\ uA^2 \\ \vdots \\ uA^{d-1} \end{bmatrix} \in \mathbb{K}^{d \times n}$

naive algorithm: d matrix-vector products $O(n^2 d)$

Keller-Gehrig [1985]: $\log(d)$ matrix-matrix products $O(n^\omega \log(d))$

via kernel basis: $\mathcal{K}(\begin{bmatrix} I - xA \\ -u \end{bmatrix})$ + power series expansions $O(n^\omega + nM(d))$

some use cases (typically $d = O(n)$):

- minimal polynomial of a vector / of a matrix
- invariant factors / Frobenius normal form
- test for matrix similarity "is there U s.t. $A = U^{-1}BU$?"
- Wiedemann algorithm, Gröbner basis change of ordering

use 2: faster linear algebra (Krylov iterates)

$$(I - xA)^{-1} = \sum_{k \geqslant 0} A^k x^k$$

- series expansion of the inverse of $I - xA$

use 2: faster linear algebra (Krylov iterates)

$$u(I - xA)^{-1} = \sum_{k \geqslant 0} uA^k x^k$$

- series expansion of the inverse of $I - xA$
- projection by u

use 2: faster linear algebra (Krylov iterates)

$$t(x)^{-1}s(x) = u(I - xA)^{-1} = \sum_{k \geqslant 0} uA^k x^k$$

- series expansion of the inverse of $I - xA$
- projection by u
- switch between left/right matrix fraction descriptions
  $\rightsquigarrow$ denominator $n \times n$ of degree 1 becomes 1×1 of degree n
  $\rightsquigarrow$ numerator $1 \times n$ of degree < 1 becomes $1 \times n$ of degree $< n$

use 2: faster linear algebra (Krylov iterates)

$$t(x)^{-1}s(x) = u(I - xA)^{-1} = \sum_{k \geqslant 0} uA^k x^k$$

$$\Updownarrow$$

$$s(x)(I - xA) = t(x)\ u$$

- series expansion of the inverse of $I - xA$
- projection by u
- switch between left/right matrix fraction descriptions
  $\rightsquigarrow$ denominator $n \times n$ of degree 1 becomes 1×1 of degree n
  $\rightsquigarrow$ numerator $1 \times n$ of degree < 1 becomes $1 \times n$ of degree $< n$

use 2: faster linear algebra (Krylov iterates)

$$t(x)^{-1}s(x) = u(I - xA)^{-1} = \sum_{k \geqslant 0} uA^k x^k$$

$$\Updownarrow$$

$$s(x)(I - xA) = t(x)\,u \;\Leftrightarrow\; \begin{bmatrix} s(x) & t(x) \end{bmatrix} \begin{bmatrix} I - xA \\ -u \end{bmatrix} = 0$$

- series expansion of the inverse of $I - xA$
- projection by u
- switch between left/right matrix fraction descriptions
  $\rightsquigarrow$ denominator $n \times n$ of degree 1 becomes 1×1 of degree n
  $\rightsquigarrow$ numerator $1 \times n$ of degree < 1 becomes $1 \times n$ of degree $< n$

use 2: faster linear algebra (Krylov iterates)

$$t(x)^{-1}s(x) = u(I - xA)^{-1} = \sum_{k \geqslant 0} uA^k x^k$$

$$\Updownarrow$$

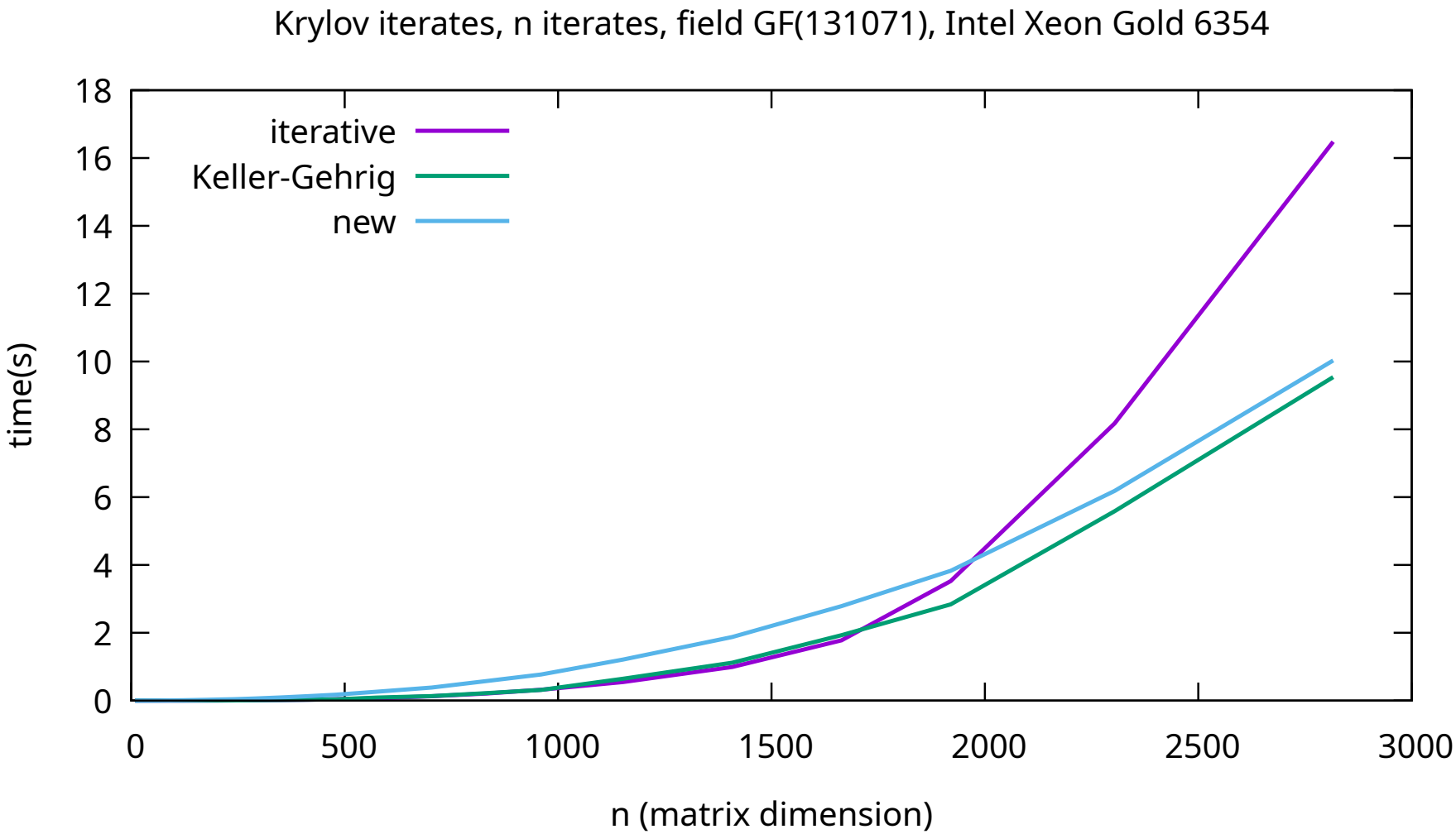
$$s(x)(I - xA) = t(x)\,u \;\Leftrightarrow\; \begin{bmatrix} s(x) & t(x) \end{bmatrix} \begin{bmatrix} I - xA \\ -u \end{bmatrix} = 0$$

- series expansion of the inverse of $I - xA$
- projection by u
- switch between left/right matrix fraction descriptions
  $\rightsquigarrow$ denominator $n \times n$ of degree 1 becomes 1×1 of degree n
  $\rightsquigarrow$ numerator $1 \times n$ of degree < 1 becomes $1 \times n$ of degree $< n$

1. find s, t via a **minimal kernel basis**
   - algorithm [Zhou-Labahn-Storjohann'12]
   - analysis of log factors $\rightsquigarrow O(n^\omega)$
2. deduce d Krylov iterates by power series operations $\rightsquigarrow O(nM(d))$

(they are given by $t(x)^{-1}s(x) \bmod x^d$)

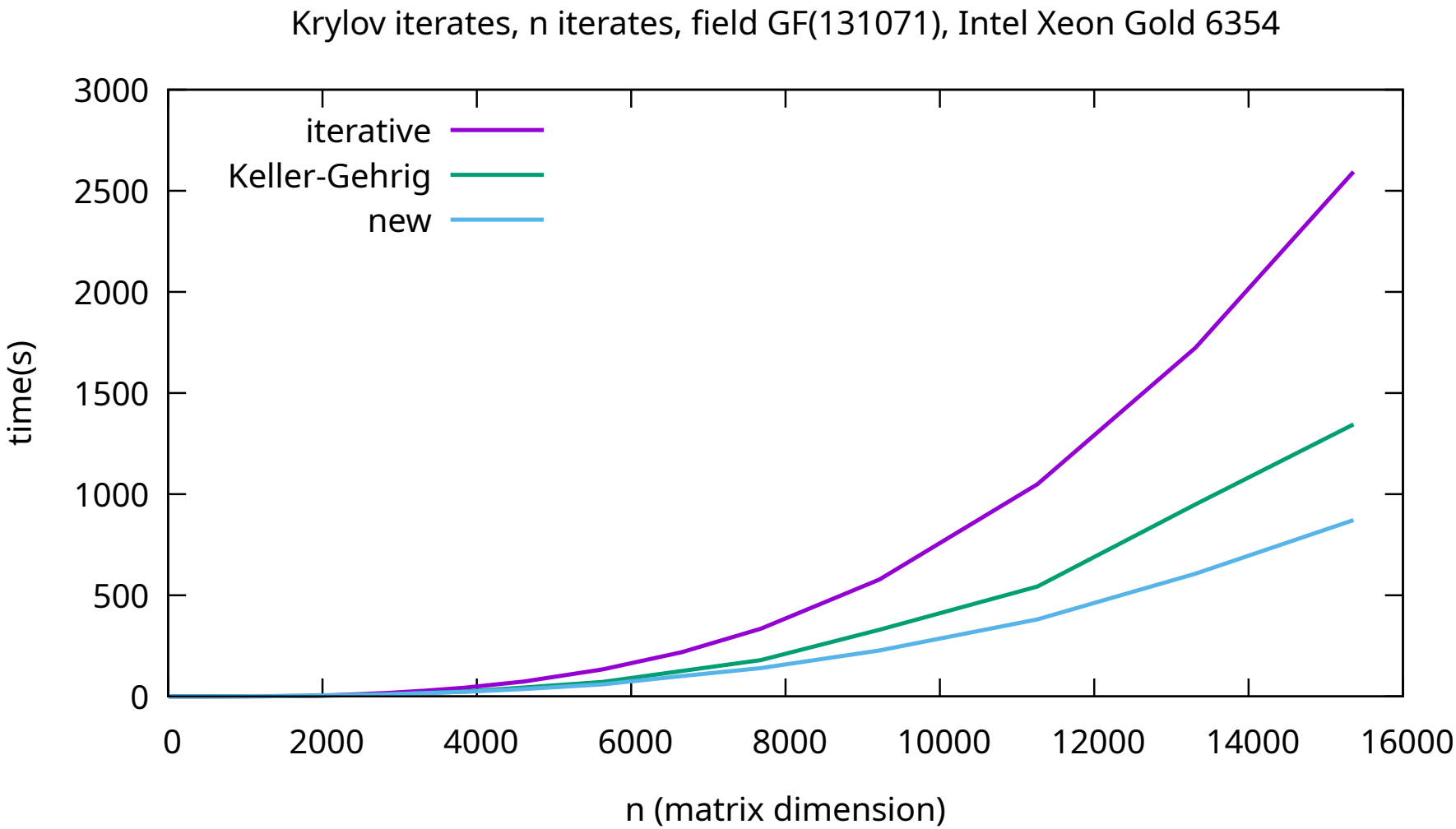
use 2: faster linear algebra (Krylov iterates)



prototype implementation in NTL / Polynomial Matrix Library

`https://github.com/libntl/ntl` `https://github.com/vneiger/pml`

use 2: faster linear algebra (Krylov iterates)



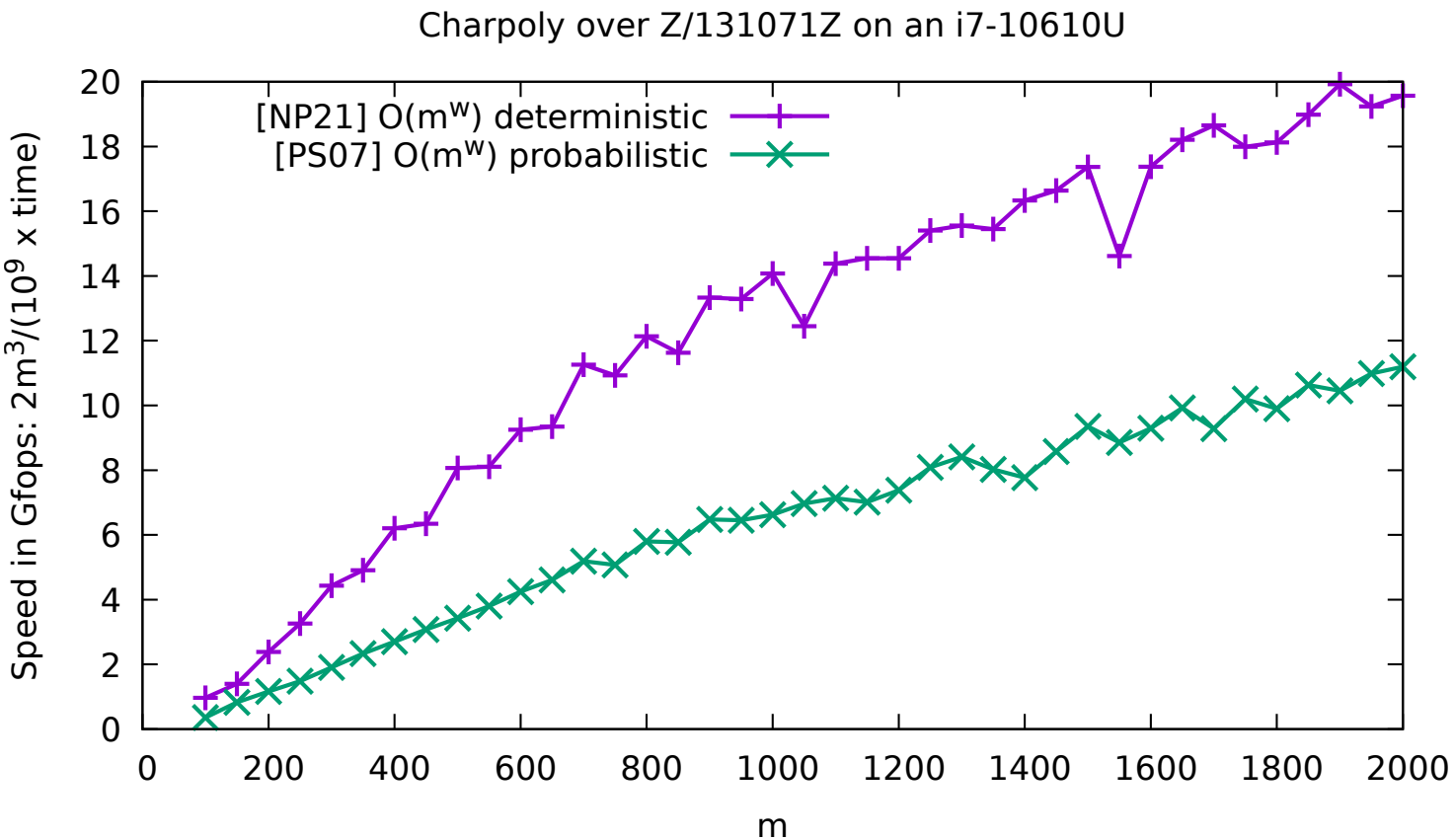
prototype implementation in NTL / Polynomial Matrix Library

`https://github.com/libntl/ntl` `https://github.com/vneiger/pml`

use 2: faster linear algebra (characteristic polynomial)

further in this direction:

- deterministic characteristic polynomial in $O(n^\omega)$ [N.-Pernet '21]
- faster Krylov matrix, Krylov basis, and Frobenius form [N.-Pernet-Villard '24]



prototyped in LinBox and fflas-ffpack `https://github.com/linbox-team/`

use 3: bivariate evaluation and interpolation

problems: given points $(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)$ in $\mathbb{K}^2$,
- given $p(x, y)$, compute $p(\alpha_i, \beta_i)$ for $1 \leqslant i \leqslant n$
- find $p(x, y)$ of small degree such that $p(\alpha_i, \beta_i) = 0$

[Nüsken-Ziegler 2004] [Möller-Buchberger 1982] [Marinari-Möller-Mora 1993]

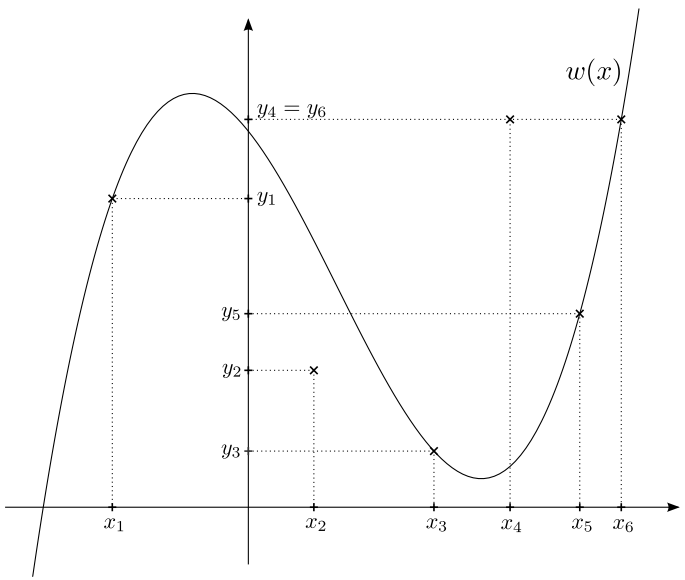
bivariate multipoint evaluation is related to modular composition $p(x, L(x)) \bmod M(x)$

bivariate interpolation = main step in Reed-Solomon list-decoding (univariate interpolation with errors)

[Guruswami-Sudan 1999] [Kötter-Vardy 2003]

extensions to AG codes via good bases of Riemann-Roch spaces

[Beelen-Rosenkilde-Solomatov 2022]



algebraic approximants and **bi-level structures**

use 3: bivariate evaluation and interpolation

example:
$(\alpha_i, \beta_i)_{1 \leqslant i \leqslant 8} = \{(24, 80), (31, 73), (15, 73), (32, 35), (83, 66), (27, 46), (20, 91), (59, 64)\}$
find a bivariate $p(x, y) \in \mathbb{K}[x, y]$ such that $p(\alpha_i, \beta_i) = 0$ for $1 \leqslant i \leqslant 8$

$$\left.\begin{array}{l} M(x) \leftarrow (x - 24) \cdots (x - 59) \\ f(x) \leftarrow \text{interpolant } f(\alpha_i) = \beta_i \end{array}\right\} \longrightarrow \text{all solutions} = \text{ideal } \langle M(x), y - f(x) \rangle$$

solutions of **smaller x-degree, larger y-degree**:

find $p(x, y) = p_0(x) + p_1(x)y + p_2(x)y^2$
such that $p(x, f(x)) = 0 \bmod M(x)$

$$\iff \begin{bmatrix} p_0 & p_1 & p_2 \end{bmatrix} \begin{bmatrix} 1 \\ f \\ f^2 \end{bmatrix} = 0 \bmod M$$

- bivariate ideal $\rightarrow$ univariate module
- this is a structured M-Padé equation (powers of f mod M)
- computing an s-minimal basis provides solution satisfying degree constraints

use 3: bivariate evaluation and interpolation

example:

$(\alpha_i, \beta_i)_{1 \leqslant i \leqslant 8} = \{(24, 80), (31, 73), (15, 73), (32, 35), (83, 66), (27, 46), (20, 91), (59, 64)\}$

find a bivariate $p(x, y) \in \mathbb{K}[x, y]$ such that $p(\alpha_i, \beta_i) = 0$ for $1 \leqslant i \leqslant 8$

structured linear algebra viewpoint

add specific degree constraints:

find $p(x, y) = p_{00} + p_{01}x + p_{02}x^2 + p_{03}x^3 + p_{04}x^4 + (p_{10} + p_{11}x + p_{12}x^2)y + p_{20}y^2$

$$\left[\begin{array}{ccccc:ccc:c} p_{00} & p_{01} & p_{02} & p_{03} & p_{04} & p_{10} & p_{11} & p_{12} & p_{20} \end{array}\right] \left[\begin{array}{cccc} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_8 \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_8^2 \\ \alpha_1^3 & \alpha_2^3 & \cdots & \alpha_8^3 \\ \alpha_1^4 & \alpha_2^4 & \cdots & \alpha_8^4 \\ \hdashline \beta_1 & \beta_2 & \cdots & \beta_8 \\ \alpha_1\beta_1 & \alpha_2\beta_2 & \cdots & \alpha_8\beta_8 \\ \alpha_1^2\beta_1 & \alpha_2^2\beta_2 & \cdots & \alpha_8^2\beta_8 \\ \hdashline \beta_1^2 & \beta_2^2 & \cdots & \beta_8^2 \end{array}\right] = 0$$

- bivariate ideal $\rightarrow$ univariate module $\rightarrow$ $\mathbb{K}$-linear system
- two levels of Vandermonde structure

use 4: multivariate Gröbner basis computations

⚠ **conjecture:** *this should be a moment in the tutorial when the audience is dreaming about the welcome reception* @18h15
⇝ *let's just give a very quick overview*

Gröbner basis computations

$$f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_r] \;\to\; \mathcal{G}_{\text{drl}}$$

typically rely on $\mathbb{K}$-linear algebra [Macaulay 1902+1916] [Lazard 1983]
⇝ can benefit from using univariate polynomial matrices

(situation-specific, and tricky too analyze due to degree growth, sparsity, . . .)

favorable case: **change of monomial order** (finite-dimensional space)

- polynomial system solving: $\mathcal{G}_{\text{drl}} \;\to\; \mathcal{G}_{\text{lex}}$
- basis of vanishing ideal: $\alpha_1, \ldots, \alpha_D \in \mathbb{K}^r \;\to\; \mathcal{G}_{\text{drl}}$

matrices are smaller, but univariate ⇝ exploits one level of structure

[FGLM 1993, MMM 1993, Faugère-Mou 2011, Berthomieu-Neiger-Safey 2022]
[Wiedemann 1986, Coppersmith 1994, Pernet-Storjohann 2007]

rule of thumb: fewer variables ⇒ greater benefit

bonus: overview of block-Wiedemann

given a sparse matrix $A \in \mathbb{K}^{n \times n}$:
- solve a linear system $Au = v$
- compute the minimal polynomial of A

. sparse means that A has a large proportion of zero entries

. goal: exploit sparsity to do better than $O(n^\omega)$

[Wiedemann 1986, Coppersmith 1994, Kaltofen 1995, Villard 1997]

block Wiedemann approach, for block dimension m:

1. choose random blocking matrices $U, V \in \mathbb{K}^{n \times m}$
2. compute Krylov matrix: $[V \;\; AV \;\; \cdots \;\; A^{d-1}V]$
3. deduce linearly recurrent sequence of matrices in $\mathbb{K}^{m \times m}$
$$U^T V, U^T A V, \ldots, U^T A^{d-1} V$$
4. find polynomial matrix generator $P \in \mathbb{K}[x]^{m \times m}$ of this sequence

bonus: overview of block-Wiedemann

given a sparse matrix $A \in \mathbb{K}^{n \times n}$:
- solve a linear system $Au = v$
- compute the minimal polynomial of A

. sparse means that A has a large proportion of zero entries

. goal: exploit sparsity to do better than $O(n^\omega)$

[Wiedemann 1986, Coppersmith 1994, Kaltofen 1995, Villard 1997]
block Wiedemann approach, for block dimension m:

1. choose random blocking matrices $U, V \in \mathbb{K}^{n \times m}$
2. compute Krylov matrix: $[V \;\; AV \;\; \cdots \;\; A^{d-1}V]$
3. deduce linearly recurrent sequence of matrices in $\mathbb{K}^{m \times m}$
$$U^T V, U^T AV, \ldots, U^T A^{d-1} V$$
4. find polynomial matrix generator $P \in \mathbb{K}[x]^{m \times m}$ of this sequence

- generically, taking $d = 2n/m$ terms is sufficient
- step 4 is M-Padé, in $\tilde{O}(m^\omega d) = \tilde{O}(m^{\omega-1} n)$
- often, m is related to the number of threads available for parallel computation of the matrix sequence

open questions: Frobenius and Smith forms

deterministic, log-free Frobenius form

$$\begin{bmatrix} \quad A \quad \end{bmatrix} \longrightarrow \begin{bmatrix} C_{\mu_1} & & & \\ & C_{\mu_2} & & \\ & & \ddots & \\ & & & C_{\mu_m} \end{bmatrix}$$

μ_{i+1} divides μ_i

- complexity $O(n^\omega)$ [Pernet-Storjohann'07]
- Las Vegas, requires large field
- new techniques $\rightsquigarrow O(n^\omega (\log\log n)^2)$

deterministic algo in $O(n^\omega)$?

open questions: Frobenius and Smith forms

deterministic, log-free Frobenius form

$$\begin{bmatrix} & A & \end{bmatrix} \longrightarrow \begin{bmatrix} C_{\mu_1} & & & \\ & C_{\mu_2} & & \\ & & \ddots & \\ & & & C_{\mu_m} \end{bmatrix}$$

μ_{i+1} divides μ_i

- complexity $O(n^\omega)$ [Pernet-Storjohann'07]
- Las Vegas, requires large field
- new techniques $\rightsquigarrow O(n^\omega (\log\log n)^2)$

deterministic algo in $O(n^\omega)$?

deterministic Smith form

$$\begin{bmatrix} & A(x) & \end{bmatrix} \longrightarrow \begin{bmatrix} \mu_1 & & & \\ & \mu_2 & & \\ & & \ddots & \\ & & & \mu_m \end{bmatrix}$$

μ_{i+1} divides μ_i

- complexity $\tilde{O}(m^\omega \frac{D}{m})$ [Storjohann'03]
- Las Vegas, requires large field
- exploit progress on $\mathbb{K}[x]$-matrices?

deterministic algo in $\tilde{O}(m^\omega \frac{D}{m})$?

Schönhage's rule 8: the development of fast algorithms is slow!

first guess, then prove

- linearly recurrent sequences
- guessing algebraicity or D-finiteness
- Hermite-Padé approximants

polynomials and matrices

- algebraic computations
- asymptotic complexity bounds
- basics of polynomial matrices

guessing univariate relations

- vector approximation/interpolation
- fast divide and conquer scheme
- unbalanced degrees and Popov bases

software, algorithms, impacts

- software implementation
- kernel bases and Hermite forms
- some uses, from Krylov to Gröbner