Symbolic computation versus numeric computation

George Labahn
Symbolic Computation Group
University of Waterloo

Claude-Pierre Jeannerod
Laboratoire LIP
INRIA - ENS Lyon

SCA 2002

- Lecture 1: Approximate polynomial algebra
- Lecture 2: The approximate GCD problem

Example

$$
\begin{array}{lcl}
a(z) = (z - \frac{1}{3}) \cdot (z - \frac{5}{3}) & \longrightarrow & z^2 - 2z + .56 \\
b(z) = z - \frac{1}{3} & \longrightarrow & z - .33
\end{array}
$$

$$
\begin{array}{lcl}
a(z) = 50 - 7z & \longrightarrow & 50 - 7z \\
b(z) = z - \frac{1}{7} & \longrightarrow & z - .14
\end{array}
$$

GCD$(a(z), b(z)) = ?$ **NORMAL**$(\frac{a(z)}{b(z)}) = ?$

Answer: Error message? convert/exact? Euclid Algorithm?

Example: Euclid's Algorithm

$$a(z) = z^4 + z^3 + (1+\epsilon)\,z^2 + \epsilon z + 1, \qquad b(z) = z^3 - z^2 + 3\,z - 2$$

$$c(z) = \mathsf{rem}(a, b) = \epsilon z^2 + (\epsilon - 4)z + 5.$$

$$d(z) = \mathsf{rem}(b, c) = (5 - \tfrac{17}{\epsilon} + \tfrac{16}{\epsilon^2})z + (-2 + \tfrac{10}{\epsilon} - \tfrac{20}{\epsilon^2})$$

$$e(z) = \mathsf{rem}(c, d) = \frac{\epsilon^2\left(-39\,\epsilon^2 - 32\,\epsilon + 137 + 14\,\epsilon^3\right)}{\left(5\,\epsilon^2 - 17\,\epsilon + 16\right)^2}$$

ϵ is small numeric $\begin{cases} \epsilon \text{ less than tolerance - good} \\ \epsilon \text{ a little over tolerance - horrible errors} \end{cases}$

Why look at symbolic-numeric problems?

- Linear systems theory
- Control theory
- Computer aided design
- ...

[See lectures by L. Gonzalez-Vega]

Outline for lecture 1

- Approximate input data
- Approximate algebraic operations
  - Ill-posed problems
  - Nearness problems
- Models of computation
- Algorithmic goals (complexity, stability)
- Related available software

Approximate input data

The data to handle are polynomials over $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ with coefficients known up to "small errors": uncertainty bounded by a given ϵ.

Approximate input data

The data to handle are polynomials over $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ with coefficients known up to "small errors": uncertainty bounded by a given ϵ.

Rather than the polynomial $a(z) = \sum_{i=0}^{d} a_i z^i$, one will thus consider the set of nearby polynomials

$$\{a^*(z) \,:\, \|a - a^*\| \leq \epsilon,\, \deg a^* \leq d\}.$$

Approximate input data (cont'd)

- Examples: $\epsilon = 2^{-32} \sim 10^{-10}$ for $a(z)$ computed in single-prec.;
  ϵ = accuracy of physical measurement or storage.

- Intrinsic properties are preserved (degree, monic,...).

- Perturbations can be assumed sparse, structured, real,....

Another representation: approximate roots [Pan - 96]

- Write $a(z) = \prod_{i=1}^{d}(z - a_i)$ instead of $a(z) = \sum_{i=0}^{d} a_i z^i$.

Another representation: approximate roots [Pan - 96]

- Write $a(z) = \prod_{i=1}^{d}(z - a_i)$ instead of $a(z) = \sum_{i=0}^{d} a_i z^i$.

- Then, for a given error bound ϵ, we represent $a(z)$ by the set

$$\{a^*(z) = \prod_{i=1}^{d}(z - a_i^*) \ : \ |a_i - a_i^*| \leq \epsilon\}.$$

Another representation: approximate roots [Pan - 96]

- Write $a(z) = \prod_{i=1}^{d}(z - a_i)$ instead of $a(z) = \sum_{i=0}^{d} a_i z^i$.

- Then, for a given error bound ϵ, we represent $a(z)$ by the set

$$\{a^*(z) = \prod_{i=1}^{d}(z - a_i^*) \ : \ |a_i - a_i^*| \leq \epsilon\}.$$

- This implies that $\|a - a^*\| \leq O(\epsilon)\|a\|$ and $\deg a^* = d$:

$$\text{Nearby roots} \quad \Longrightarrow \quad \text{Nearby polynomials}$$

The converse of course is not true in general.

Ill-posed problems

• For *error-free input* ($\epsilon = 0$), polynomial operations such as $+$, $-$, $\times$, division, testing coprimeness, GCDs, LCMs and factorization can be performed in a provably correct way by $exact$ computations.

Ill-posed problems

- For *error-free input* ($\epsilon = 0$), polynomial operations such as $+$, $-$, $\times$, division, testing coprimeness, GCDs, LCMs and factorization can be performed in a provably correct way by $exact$ computations.

- Such operations may correspond to ill-posed problems (continuous perturbations of the input lead to discontinuities in the output):

$$\deg \gcd(z^2 - 9, z + 3) = 1 \qquad \deg \gcd(z^2 - 9, z + 3 + \epsilon) = 0$$

- Asking for a "GCD of two approximate polynomials" using the $classic$ definition of a GCD is therefore asking the wrong question.

- Simply converting floating-point data to exact rational data is also the wrong thing to do.

- Asking for a "GCD of two approximate polynomials" using the $classic$ definition of a GCD is therefore asking the wrong question.

- Simply converting floating-point data to exact rational data is also the wrong thing to do.

- What should the classic definitions of these algebraic operations by replaced with in the presence of approximate data?

Approximate counterparts of algebraic ops

- Ill-posed problems are converted into a so-called approximate (or epsilon or numerical) associated problem.

- Example: approximate coprimeness test:
  "among all the polynomial pairs defined by two approximate polynomials, is there at least one pair that has a nontrivial exact GCD?"

More formally, given $a(z)$, $b(z)$ with degrees m, n and given an error bound $\epsilon > 0$,

1. Let d be the distance between the pair (a, b) and the set

$$\{(a^*, b^*) \,:\, \deg(a^*, b^*) \leq (m, n),\ \deg \gcd(a^*, b^*) > 0\}$$

2. Then a and b are ϵ-coprime if and only if $\epsilon < d$.

Examples of approximate problems

- Coprimeness test
- GCD, LCM
- Factorization
- Polynomial systems
- Matrix normal forms
- Multivariate problems

Nearness problems

In the presence of approximate input data, usual computer algebra questions are typically replaced with so-called nearness problems.

Nearness problems

In the presence of approximate input data, usual computer algebra questions are typically replaced with so-called nearness problems.

Examples for the approximate coprimeness test:

- Are the given polynomials $near$ a noncoprime pair? (distance)
- Return such a $near$ noncoprime pair.
- Return a $nearest$ noncoprime pair.

Nearness problems

In the presence of approximate input data, usual computer algebra questions are typically replaced with so-called nearness problems.

Examples for the approximate coprimeness test:

- Are the given polynomials $near$ a noncoprime pair? (distance)
- Return such a $near$ noncoprime pair.
- Return a $nearest$ noncoprime pair.

What $near$ means depends on the value of ϵ and the distance used (usually defined by a suitable norm $\| \cdot \|$).

Additional nearness problems

Other nearness problems include:

- Nearest polynomial with a $real$ root
- Nearest polynomial with a root of multiplicity r
- Nearest polynomial with given roots

[Hitz & Kaltofen - 98, 99], [Stetter - 99]

Example: Karmarkar & Lakshman

Nearest noncoprime pair via optimization:

Given monic polynomials $a, b \in \mathbb{C}[z]$ of the same degree n,
find monic polynomials $p, q \in \mathbb{C}[z]$ of degree n and such that

- p, q have a common root,
- $\|a - p\|^2 + \|b - q\|^2$ is minimized. (Here, $\| \ \| = \| \ \|_2$.)

[Karmarkar & Lakshman - 98] Polynomial algorithm in n and a bound of the bit size of the coefficients of a, b.

Steps of Algorithm

1. The minimum of $\|a-p\|^2+\|b-q\|^2$ as a function of x and under the constraints $p(x)=q(x)=0$ is

$$M(x):=\frac{a(x)a^*(x)+b(x)b^*(x)}{\sum_{i=0}^{n-1}\left(xx^*\right)^i},\quad x=u+iv\in\mathbb{C}.$$

Steps of Algorithm

1. The minimum of $\|a-p\|^2+\|b-q\|^2$ as a function of x and under the constraints $p(x)=q(x)=0$ is

$$M(x) := \frac{a(x)a^*(x)+b(x)b^*(x)}{\sum_{i=0}^{n-1}\left(xx^*\right)^i}, \quad x=u+iv\in\mathbb{C}.$$

2. Let x_0 s.t. $\frac{\partial M}{\partial u}(x_0)=0$, $\frac{\partial M}{\partial v}(x_0)=0$ and $M(x_0)$ is minimum.

Steps of Algorithm

1. The minimum of $\|a-p\|^2+\|b-q\|^2$ as a function of x and under the constraints $p(x)=q(x)=0$ is

$$M(x) := \frac{a(x)a^*(x)+b(x)b^*(x)}{\sum_{i=0}^{n-1}(xx^*)^i}, \quad x=u+iv\in\mathbb{C}.$$

2. Let x_0 s.t. $\frac{\partial M}{\partial u}(x_0)=0$, $\frac{\partial M}{\partial v}(x_0)=0$ and $M(x_0)$ is minimum.

3. With $C=\sum_{j=0}^{n-1}(x_0x_0^*)^j$, one has

$$p_i := a_i - \frac{a(x_0)}{C}x_0^{*i}, \quad q_i := b_i - \frac{b(x_0)}{C}x_0^{*i}.$$

- The most expensive part is step 2:

Compute the intersections of $\frac{\partial M}{\partial u} = 0$ and $\frac{\partial M}{\partial v} = 0$ and take as x_0 the one that minimizes $M(x)$.

- Two simplifications:

  1) Search inside the box $u, v \in [-B, B]$, $B = 5 \max(\|a\|^2 + \|b\|^2)$.

  2) $o(\log \epsilon + n \log B)$ bits of precision are enough to ensure that

$$M(x_0) - (\text{absolute minimum of } M(x)) \leq \epsilon^2.$$

Example of $M(x)$ (Karmarkar & Lakshman)

$$a(z) = z^2 - 6z + 5 = (z-1)(z-5), \qquad b(z) = z^2 - 6.3z + 5.72 = (z-1.1)(z-5.2).$$

$$M(u+iv) = \frac{2u^4 + 2v^4 + 4u^2v^2 - \frac{123}{5}uv^2 + \frac{9713}{100}u^2 + \frac{217}{4}v^2 - \frac{16509}{125}u + \frac{36074}{625}}{1 + u^2 + v^2}$$

Minima to within 10^{-6}: $(1.054336548, 0.0)$ and $(5.096939087, 0.0)$.

Summary

- Input: monic polynomials $a, b \in \mathbb{C}[z]$ of degree n,
  an error bound ϵ.

- Output: monic polynomials $p, q \in \mathbb{C}[z]$ of degree n such that
  $p(x_0) = q(x_0) = 0$,
  $M(x_0) -$ (absolute minimum of $M(x)) \leq \epsilon^2$.

- Cost: polynomial in n, $\log B$, $\log \epsilon$ using exact methods.

Extensions

- Different degrees
- Representation in other bases
- Sparse perturbations
- Real perturbations
- Nearest singular polynomial

Nearness problems and exact linear algebra

Sylvester matrix for $a(z) = \sum_{i=0}^{3} a_i z^i$ and $b(z) = \sum_{i=0}^{2} b_i z^i$:

$$S = \left[\begin{array}{cc|ccc} a_0 & & b_0 & & \\ a_1 & a_0 & b_1 & b_0 & \\ a_2 & a_1 & b_2 & b_1 & b_0 \\ a_3 & a_2 & & b_2 & b_1 \\ & a_3 & & & b_2 \end{array}\right].$$

S is singular $\iff$ a and b have a nontrivial GCD

Karmarkar & Lakshman's algo: nearest singular Sylvester matrix.

Nearness problems and exact linear algebra

Given a Toeplitz matrix $T = \begin{bmatrix} t_0 & t_1 & t_2 & t_3 \\ t_1 & t_0 & t_1 & t_2 \\ t_2 & t_1 & t_0 & t_1 \\ t_3 & t_2 & t_1 & t_0 \end{bmatrix}$,

determine a nearest singular Toeplitz matrix [Kaltofen - 98].

This and other structured matrix cases are still open.

Computational models

• **Adaptive precision:**

- round-off errors "small enough" (compared to accuracy of input): [Corless - 95] [Emiris, Galligo, Lombardi - 97] [Pan - 96] [Rupprecht - 00]

- guaranteed to be correct provided the precision is more than a known lower bound. [Karmarkar & Lakshman - 96]

Computational models

• **Adaptive precision:**

- round-off errors "small enough" (compared to accuracy of input): [Corless - 95] [Emiris, Galligo, Lombardi - 97] [Pan - 96] [Rupprecht - 00]

- guaranteed to be correct provided the precision is more than a known lower bound. [Karmarkar & Lakshman - 96]

• **Fixed precision:** IEEE Floating-point [Beckermann & Labahn - 98]

Computational models

- **Adaptive precision:**

- round-off errors "small enough" (compared to accuracy of input):
  [Corless - 95] [Emiris, Galligo, Lombardi - 97] [Pan - 96] [Rupprecht - 00]
- guaranteed to be correct provided the precision is more than a known lower bound. [Karmarkar & Lakshman - 96]

- **Fixed precision:** IEEE Floating-point [Beckermann & Labahn - 98]

- **Hybrid:** combination [Hitz & Kaltofen - 98,99], [Noda & Sasaki - 91]

Rough algorithmic goals

- Adaptive precision:
  - Running time: poly(n) rather than exp(n) ops

Rough algorithmic goals

- Adaptive precision:
  - Running time: poly(n) rather than exp(n) ops
- Fixed precision:
  - Running time: $O(n^2)$ rather than $O(n^3)$ flops,
  - Forward / backward stability analysis.

Reminders about numerical stability [Bunch - 87]

Problem: Solve $Ax = b$. Computed solution $\hat{x}$.

- **Forward (or weak) stability:** $\|x - \hat{x}\|/\|x\|$ small for A well-conditioned.

- **Backward stability:**
  - Unstructured: $\hat{A}\hat{x} = \hat{b}$ with $\hat{A}$ close to A and $\hat{b}$ close to b.
  - Structured: $\hat{A}\hat{x} = \hat{b}$ with $\hat{A}$ close to A and of the same structure as A, and $\hat{b}$ close to b.

Available related software

- Frameworks for symbolic-numerics
  - SYNAPS (SYmbolic Numeric APplicationS):
    C++ library integrating GMP, Lapack, SuperLU, ...
- Approximate polynomial algebra
  - Matlab (GCD, matrix polynomials, ...)
  - Maple 8: SNAP (Symbolic-Numeric Algorithms for Polynomials)

The approximate GCD problem

Outline

- Some approximate GCDs
- Certification issues
- Example in fixed precision:
  Determine when two polynomials are relatively prime
  - How to decide? (use linear algebra)
  - How to compute? (fast and stable Euclidean-like algorithm)

Approximate GCDs

- Given $\epsilon > 0$, an ϵ-GCD for $a(z)$, $b(z)$ is $g = \gcd(a^*, b^*)$ where

$$\deg a^* \leq \deg a, \quad \deg b^* \leq \deg b, \tag{1}$$

$$\|a - a^*\| < \epsilon, \quad \|b - b^*\| < \epsilon, \tag{2}$$

and where $g(z)$ has maximal degree.

Approximate GCDs

- Given $\epsilon > 0$, an ϵ-GCD for $a(z)$, $b(z)$ is $g = \gcd(a^*, b^*)$ where

$$\deg a^* \leq \deg a, \quad \deg b^* \leq \deg b, \tag{1}$$

$$\|a - a^*\| < \epsilon, \quad \|b - b^*\| < \epsilon, \tag{2}$$

and where $g(z)$ has maximal degree.

- Variant: replace "maximal degree" with "positive degree".

- See [Corless *et al* - 95], [Beckermann & Labahn - 98].

Approximate GCDs (cont'd)

• Given $k \in \mathbb{N}$, a pseudo-GCD of order k for $a(z)$, $b(z)$ is $g = \gcd(a^*, b^*)$ where

$$\deg a^* \leq \deg a, \quad \deg b^* \leq \deg b, \quad \deg g = k,$$

and where the distance $\|(a, b) - (a^*, b^*)\|$ is minimal.

Approximate GCDs (cont'd)

- Given $k \in \mathbb{N}$, a pseudo-GCD of order k for $a(z)$, $b(z)$ is $g = \gcd(a^*, b^*)$ where

$$\deg a^* \leq \deg a, \quad \deg b^* \leq \deg b, \quad \deg g = k,$$

and where the distance $\|(a, b) - (a^*, b^*)\|$ is minimal.

- Variant: replace "minimal" with "$\leq \epsilon$".
- See [Karmarkar & Lakshman - 96].

Approximate GCDs (cont'd)

- For $\epsilon > 0$ and polynomials given by their roots a_i, b_j, consider the bipartite graph $G = \{(a_i, b_j)\}$ with $|a_i - b_j| \leq 2\epsilon$.
- Let $M = \{(a_{i_k}, b_{j_k})\}$ be a maximal subgraph of G.
- Pan's ϵ-GCD $g(z) := \prod_k (z - (a_{i_k} - b_{j_k})/2)$.
- See [Pan - 96].

Approximate GCDs (cont'd)

• Given $a(z)$, $b(z)$ and $\epsilon > 0$, a quasi-GCD $g(z)$ with precision ϵ is defined by the existence of $u(z), v(z), a^*(z)$ and $b^*(z)$ such that

$$\|av + bu - g\| < \epsilon\|g\| \quad \text{with} \quad \deg u < \deg a, \quad \deg v < \deg b,$$

and

$$\|a - ga^*\| < \epsilon, \quad \|b - gb^*\| < \epsilon.$$

• See [Schönhage - 85].

A remark on Schönhage's model

$\neq$ the adaptive precision model:

the input is assumed to be arbitrarily precise

$\Longrightarrow$ at any step in the algorithm,
one can get as many digits of the input coefficients as needed.

Certification issues

- SVD + extended Euclidean algorithm [Corless $et\ al$ - 95]:

lower bound on the degree of an approximate GCD.
$\Longrightarrow$ Approximate divisors only.

Certification issues

- SVD + extended Euclidean algorithm [Corless $et\ al$ - 95]:

lower bound on the degree of an approximate GCD.
$\Longrightarrow$ Approximate divisors only.

- Certification results [Emiris $et\ al$ - 95, 97], [Rupprecht - 00]:

$\Longrightarrow$ Degree guaranteed to be maximum within the given ϵ.

When are two polynomials numerically relatively prime?

Given the polynomials $a(z) = \sum_{i=0}^{m} a_i z^i$ and $b(z) = \sum_{i=0}^{n} b_i z^i$, given the set

$$P = \{(p, q) \,:\, \deg p \leq m,\, \deg q \leq n,\, \deg \gcd(p, q) > 0\}$$

and given a suitable norm $\|\cdot\|$, how to estimate the distance

$$\text{dist} = \inf_{P} \|(a, b) - (p, q)\| \quad ?$$

When are two polynomials numerically relatively prime?

Given the polynomials $a(z) = \sum_{i=0}^{m} a_i z^i$ and $b(z) = \sum_{i=0}^{n} b_i z^i$, given the set

$$P = \{(p, q) \,:\, \deg p \leq m,\, \deg q \leq n,\, \deg \gcd(p, q) > 0\}$$

and given a suitable norm $\|\cdot\|$, how to estimate the distance

$$\text{dist} = \inf_{P} \|(a, b) - (p, q)\| \quad ?$$

Answer: make use of linear algebra.

Norms

View $c \in \mathbb{C}[z]$, $c(z) = c_0 + .. + c_n z^n$ as the vector $\vec{c} = (c_0, .., c_n)^T$.

- Norm for $\mathbb{C}[z]$: $\quad ||c|| = ||\vec{c}||_1 = \sum_j |c_j|$
- Norm for $\mathbb{C}[z]^{r \times s}$: $\quad ||(c_{j,k})|| = ||(||c_{j,k}||)||_1 = \max_k \sum_j ||c_{j,k}||$.

Norms

View $c \in \mathbb{C}[z]$, $c(z) = c_0 + .. + c_n z^n$ as the vector $\vec{c} = (c_0, .., c_n)^T$.

- Norm for $\mathbb{C}[z]$: $\quad ||c|| = ||\vec{c}||_1 = \sum_j |c_j|$
- Norm for $\mathbb{C}[z]^{r \times s}$: $\quad ||(c_{j,k})|| = ||(||c_{j,k}||)||_1 = \max_k \sum_j ||c_{j,k}||$.

Note: $||c \cdot d|| \leq ||c|| \cdot ||d||$

Linear algebra formulation

- Sylvester matrix for $a(z)$ and $b(z)$:

$$S = \left[\begin{array}{cc|ccc} a_0 & & b_0 & & \\ a_1 & a_0 & b_1 & b_0 & \\ a_2 & a_1 & b_2 & b_1 & b_0 \\ a_3 & a_2 & & b_2 & b_1 \\ & a_3 & & & b_2 \end{array}\right]$$

Linear algebra formulation

- Sylvester matrix for $a(z)$ and $b(z)$:

$$S = \left[\begin{array}{ccc|ccc} a_0 & & & b_0 & & \\ a_1 & a_0 & & b_1 & b_0 & \\ a_2 & a_1 & & b_2 & b_1 & b_0 \\ a_3 & a_2 & & & b_2 & b_1 \\ & a_3 & & & & b_2 \end{array}\right]$$

- Well known: $\det S \neq 0 \iff \gcd(a, b) = 1$

$$\iff \text{can solve } a(z)s(z) + b(z)t(z) = 1.$$

Linear algebra formulation (cont'd)

Also well known: classical lower bound $\frac{1}{||S^{-1}||_1} \leq \text{dist}$.

Proof: $||S||_1 = ||(a,b)||$ where $||B||_1 = \max_{x \neq 0} ||Bx||_1/||x||$

$$\implies \quad \text{dist} = \inf\{||S - S(p,q)||_1 : S(p,q) \text{ singular}\}$$

$$\geq \min\{||S - B||_1 : \ B \text{ singular}\} = \frac{1}{||S^{-1}||_1}$$

□

- Note: In the case of the 2-norm, we have

$$\sigma_j = \min\{||S - B||_2 : \text{ rank}(B) = j - 1\},$$

with $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_{m+n}$ being the singular values of S. Allows one to define the ϵ–rank and ϵ–GCD [Corless $et\ al$ - 95].

- Note: In the case of the 2-norm, we have

$$\sigma_j = \min\{||S - B||_2 : \text{ rank}(B) = j - 1\},$$

  with $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_{m+n}$ being the singular values of S. Allows one to define the ϵ–rank and ϵ–GCD [Corless et al - 95].

- Problem: SVD is expensive.

- Also would like to improve on lower bound.

Formula for the inverse of a Sylvester matrix

- $\det S \neq 0 \Leftrightarrow$ can solve $a(z)s(z) + b(z)t(z) = 1$
- $\det S \neq 0 \Leftrightarrow$ can solve

$$S \cdot [s_0, \cdots, s_{n-1}, t_0, \cdots, t_{m-1}]^T = [1, 0, \cdots, 0]^T.$$

- Only need to know that a single column of inverse exists!

Formula for the inverse of a Sylvester matrix

- $\det S \neq 0 \Leftrightarrow$ can solve $a(z)s(z) + b(z)t(z) = 1$
- $\det S \neq 0 \Leftrightarrow$ can solve

$$S \cdot [s_0, \cdots, s_{n-1}, t_0, \cdots, t_{m-1}]^T = [1, 0, \cdots, 0]^T.$$

- Only need to know that a single column of inverse exists!
- Entire inverse given by $a(z)$, $b(z)$, $s(z)$, $t(z)$, $t(z)/a(z)$!

- The inverse S^{-1} is

$$\begin{bmatrix} s_0 & 0 & \cdots & \cdots & \cdots & 0 & b_0 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & & & \vdots & \vdots & \ddots & \ddots & & & \vdots \\ s_{n-1} & \cdots & s_0 & 0 & \cdots & 0 & b_{n-1} & \cdots & b_0 & 0 & \cdots & 0 \\ t_0 & 0 & \cdots & \cdots & \cdots & 0 & -a_0 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & & & \vdots & \vdots & \ddots & \ddots & & & \vdots \\ t_{m-1} & \cdots & t_0 & 0 & \cdots & 0 & -a_{m-1} & \cdots & -a_0 & 0 & \cdots & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \\ 0 & f_{-1} & \cdots & f_{1-m-n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & f_{-1} \end{bmatrix}$$

where $f_{-1}z^{-1} + \ldots + f_{1-m-n}z^{1-m-n} = \frac{t(z)}{a(z)} + \mathcal{O}(z^{-m-n})_{z\to\infty}$.

- The inverse S^{-1} is

$$\begin{bmatrix} s_0 & 0 & \cdots & \cdots & \cdots & 0 & b_0 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & & & \vdots & \vdots & \ddots & \ddots & & & \vdots \\ s_{n-1} & \cdots & s_0 & 0 & \cdots & 0 & b_{n-1} & \cdots & b_0 & 0 & \cdots & 0 \\ t_0 & 0 & \cdots & \cdots & \cdots & 0 & -a_0 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & & & \vdots & \vdots & \ddots & \ddots & & & \vdots \\ t_{m-1} & \cdots & t_0 & 0 & \cdots & 0 & -a_{m-1} & \cdots & -a_0 & 0 & \cdots & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \\ 0 & f_{-1} & \cdots & f_{1-m-n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & f_{-1} \end{bmatrix}$$

where $f_{-1}z^{-1} + \ldots + f_{1-m-n}z^{1-m-n} = \frac{t(z)}{a(z)} + \mathcal{O}(z^{-m-n})_{z\to\infty}$.

- From inverse formula we get

$$\left|\left|\begin{pmatrix} s \\ t \end{pmatrix}\right|\right| \leq ||S^{-1}||_1 \leq \left|\left|\begin{pmatrix} s \\ t \end{pmatrix}\right|\right| + 2 \cdot ||f|| \cdot ||(a,b)||.$$

- The inverse S^{-1} is

$$\begin{bmatrix} s_0 & 0 & \cdots & \cdots & \cdots & 0 & b_0 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & & & \vdots & \vdots & \ddots & \ddots & & & \vdots \\ s_{n-1} & \cdots & s_0 & 0 & \cdots & 0 & b_{n-1} & \cdots & b_0 & 0 & \cdots & 0 \\ t_0 & 0 & \cdots & \cdots & \cdots & 0 & -a_0 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & & & \vdots & \vdots & \ddots & \ddots & & & \vdots \\ t_{m-1} & \cdots & t_0 & 0 & \cdots & 0 & -a_{m-1} & \cdots & -a_0 & 0 & \cdots & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \\ 0 & f_{-1} & \cdots & f_{1-m-n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & f_{-1} \end{bmatrix}$$

where $f_{-1}z^{-1} + \ldots + f_{1-m-n}z^{1-m-n} = \frac{t(z)}{a(z)} + \mathcal{O}(z^{-m-n})_{z\to\infty}$.

- From inverse formula we get

$$\left|\left|\begin{pmatrix} s \\ t \end{pmatrix}\right|\right| \leq ||S^{-1}||_1 \leq \left|\left|\begin{pmatrix} s \\ t \end{pmatrix}\right|\right| + 2 \cdot ||f|| \cdot ||(a,b)||.$$

Problem: how large is $||f||$?

- Solve $a(z)u(z)+b(z)v(z) = z^{m+n-1}$ i.e. $S \cdot \begin{pmatrix} \vec{u} \\ \vec{v} \end{pmatrix} = e_{m+n}^T$.

- Let

$$\kappa := \left\| \begin{pmatrix} s & u \\ t & v \end{pmatrix} \right\| = \max\{ \left\| \begin{pmatrix} s \\ t \end{pmatrix} \right\|, \left\| \begin{pmatrix} u \\ v \end{pmatrix} \right\| \}.$$

- Solve $a(z)u(z)+b(z)v(z) = z^{m+n-1}$ i.e. $S \cdot \begin{pmatrix} \vec{u} \\ \vec{v} \end{pmatrix} = e_{m+n}^T$.

- Let

$$\kappa := \left|\left|\begin{pmatrix} s & u \\ t & v \end{pmatrix}\right|\right| = \max\{\left|\left|\begin{pmatrix} s \\ t \end{pmatrix}\right|\right|, \left|\left|\begin{pmatrix} u \\ v \end{pmatrix}\right|\right|\}.$$

Answer: $\kappa \leq ||S^{-1}||_1 \leq \kappa + 2 \cdot ||f|| \cdot ||(a,b)||$

$$||f|| = ||u \cdot t - v \cdot s|| \leq \kappa^2$$

A better lower bound

Sylvester matrix for $a(z)$ and $b(z)$ and its inverse:

$$S = \left[\begin{array}{cc|ccc} a_0 & & b_0 & & \\ a_1 & a_0 & b_1 & b_0 & \\ a_2 & a_1 & b_2 & b_1 & b_0 \\ a_3 & a_2 & & b_2 & b_1 \\ & a_3 & & & b_2 \end{array}\right] \qquad S^{-1} = \left[\begin{array}{ccccc} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{array}\right]$$

$$\frac{1}{\|S^{-1}\|_1} \leq \frac{1}{\kappa} \leq \text{dist} \quad \text{defined by } \kappa = \max(\textstyle\sum_i |*_i|, \sum_i |*_i|).$$

Compute the first and last columns of S^{-1}

Either by solving a structured linear system

$$\left[\begin{array}{cc|ccc} a_0 & & b_0 & & \\ a_1 & a_0 & b_1 & b_0 & \\ a_2 & a_1 & b_2 & b_1 & b_0 \\ a_3 & a_2 & & b_2 & b_1 \\ & a_3 & & & b_2 \end{array}\right] \times \begin{bmatrix} s_0 & u_0 \\ s_1 & u_1 \\ t_0 & v_0 \\ t_1 & v_1 \\ t_2 & v_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

or by solving a pair of polynomial equations

$$\begin{bmatrix} a(z) & b(z) \end{bmatrix} \times \begin{bmatrix} s(z) & u(z) \\ t(z) & v(z) \end{bmatrix} = \begin{bmatrix} 1 & z^{m+n-1} \end{bmatrix}.$$

Model and algorithmic goals

Under the standard floating point model (IEEE 754), we want provably both fast and stable algorithms for computing the first and the last column of the inverse of the Sylvester matrix defined by given polynomials $a(z)$ and $b(z)$ of degrees m and n.

- Here, fast $= O((m+n)^2)$ and slow $= O((m+n)^3)$ flops.
- Proven numerical stability: either forward or (better) backward.

Two algos, depending on the version

- **Matrix version** [Chandrasekaran & Sayed - 1998]

$$\left[\begin{array}{cc|ccc} a_0 & & b_0 & & \\ a_1 & a_0 & b_1 & b_0 & \\ a_2 & a_1 & b_2 & b_1 & b_0 \\ a_3 & a_2 & & b_2 & b_1 \\ & a_3 & & & b_2 \end{array}\right] \times \begin{bmatrix} s_0 & u_0 \\ s_1 & u_1 \\ t_0 & v_0 \\ t_1 & v_1 \\ t_2 & v_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

- **Polynomial version** [Beckermann & Labahn - 1998]

$$\begin{bmatrix} a(z) & b(z) \end{bmatrix} \times \begin{bmatrix} s(z) & u(z) \\ t(z) & v(z) \end{bmatrix} = \begin{bmatrix} 1 & z^{m+n-1} \end{bmatrix}$$

Two algos, depending on the version

- **Matrix version** [Chandrasekaran & Sayed - 1998]

$$\left[\begin{array}{cc|ccc} a_0 & & b_0 & & \\ a_1 & a_0 & b_1 & b_0 & \\ a_2 & a_1 & b_2 & b_1 & b_0 \\ a_3 & a_2 & & b_2 & b_1 \\ & a_3 & & & b_2 \end{array}\right] \times \begin{bmatrix} s_0 & u_0 \\ s_1 & u_1 \\ t_0 & v_0 \\ t_1 & v_1 \\ t_2 & v_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

- **Polynomial version** [Beckermann & Labahn - 1998]

$$\begin{bmatrix} a(z) & b(z) \end{bmatrix} \times \begin{bmatrix} s(z) & u(z) \\ t(z) & v(z) \end{bmatrix} = \begin{bmatrix} 1 & z^{m+n-1} \end{bmatrix}$$

Polynomial version

- Typical polynomial remainder sequence (Euclidean algorithm):

$$\begin{bmatrix} a & b \end{bmatrix} \overset{\times U^{(1)}}{\longrightarrow} \begin{bmatrix} a^{(1)} & b^{(1)} \end{bmatrix} \overset{\times U^{(2)}}{\longrightarrow} \begin{bmatrix} a^{(2)} & b^{(2)} \end{bmatrix} \overset{\times U^{(3)}}{\longrightarrow} \begin{bmatrix} a^{(3)} & b^{(3)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$\begin{matrix} 3 & 2 \end{matrix} \qquad \begin{matrix} 2 & 1 \end{matrix} \qquad \begin{matrix} 1 & 0 \end{matrix} \qquad \begin{matrix} 0 & -\infty \end{matrix}$$

Polynomial version

- Typical polynomial remainder sequence (Euclidean algorithm):

$$\begin{bmatrix} a & b \end{bmatrix} \xrightarrow{\times U^{(1)}} \begin{bmatrix} a^{(1)} & b^{(1)} \end{bmatrix} \xrightarrow{\times U^{(2)}} \begin{bmatrix} a^{(2)} & b^{(2)} \end{bmatrix} \xrightarrow{\times U^{(3)}} \begin{bmatrix} a^{(3)} & b^{(3)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$\begin{array}{cccccccc} 3 & 2 & & 2 & 1 & & 1 & 0 & & 0 & -\infty \end{array}$$

The first column of $U^{(1)}U^{(2)}U^{(3)}$ is $\begin{bmatrix} s(z) \\ t(z) \end{bmatrix}$ s.t. $as + bt = 1$.

- Fast but unstable!

Linear algebra interpretation

$$a(z) = 1 + \epsilon z + (1+\epsilon)\, z^2 + z^3 + z^4, \qquad b(z) = -2 + 3\, z - z^2 + z^3$$

$$S = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & -2 & 0 & 0 & 0 \\ \epsilon & 1 & 0 & 3 & -2 & 0 & 0 \\ 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 & 0 \\ 1 & 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 \\ 1 & 1 & 1+\epsilon & 0 & 1 & -1 & 3 \\ 0 & 1 & 1 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Euclid's algorithm solves linear subsystems.

Linear algebra interpretation

$$a(z) = 1 + \epsilon z + (1+\epsilon)\, z^2 + z^3 + z^4, \qquad b(z) = -2 + 3\, z - z^2 + z^3$$

$$S = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & -2 & 0 & 0 & 0 \\ \epsilon & 1 & 0 & 3 & -2 & 0 & 0 \\ 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 & 0 \\ 1 & 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 \\ 1 & 1 & 1+\epsilon & 0 & 1 & -1 & 3 \\ 0 & 1 & 1 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Euclid's algorithm solves linear subsystems.

Linear algebra interpretation

$$a(z) = 1 + \epsilon z + (1+\epsilon)\, z^2 + z^3 + z^4, \qquad b(z) = -2 + 3\, z - z^2 + z^3$$

$$S = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & -2 & 0 & 0 & 0 \\ \epsilon & 1 & 0 & 3 & -2 & 0 & 0 \\ 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 & 0 \\ 1 & 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 \\ 1 & 1 & 1+\epsilon & 0 & 1 & -1 & 3 \\ 0 & 1 & 1 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Euclid's algorithm solves linear subsystems.

Linear algebra interpretation

$$a(z) = 1 + \epsilon z + (1+\epsilon)\, z^2 + z^3 + z^4, \qquad b(z) = -2 + 3\, z - z^2 + z^3$$

$$S = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & -2 & 0 & 0 & 0 \\ \epsilon & 1 & 0 & 3 & -2 & 0 & 0 \\ 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 & 0 \\ 1 & 1+\epsilon & \epsilon & 1 & -1 & 3 & -2 \\ 1 & 1 & 1+\epsilon & 0 & 1 & -1 & 3 \\ 0 & 1 & 1 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Euclid's algorithm solves linear subsystems.

Problem: what to do with ill-conditioned subsystems?

Stabilized polynomial version [Beckermann & Labahn - 1998]

Look-ahead process: if $|U^{(2)}(0)|$ too small, jump over $\begin{bmatrix} a^{(2)} & b^{(2)} \end{bmatrix}$:

$$\begin{bmatrix} a & b \end{bmatrix} \overset{\times U^{(1)}}{\longrightarrow} \begin{bmatrix} a^{(1)} & b^{(1)} \end{bmatrix} \overset{\times U^{(2)}}{\longrightarrow} \begin{bmatrix} a^{(2)} & b^{(2)} \end{bmatrix} \overset{\times U^{(3)}}{\longrightarrow} \begin{bmatrix} a^{(3)} & b^{(3)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Stabilized polynomial version [Beckermann & Labahn - 1998]

Look-ahead process: if $|U^{(2)}(0)|$ too small, jump over $\begin{bmatrix} a^{(2)} & b^{(2)} \end{bmatrix}$:

$$\begin{bmatrix} a & b \end{bmatrix} \overset{\times U^{(1)}}{\longrightarrow} \begin{bmatrix} a^{(1)} & b^{(1)} \end{bmatrix} \overset{\times U^{(2)}}{\longrightarrow} \begin{bmatrix} a^{(2)} & b^{(2)} \end{bmatrix} \overset{\times U^{(3)}}{\longrightarrow} \begin{bmatrix} a^{(3)} & b^{(3)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

- Unstability related to the ill-conditioning of some $U^{(i)}$ at $z = 0$.
- Effective way of deducing $\begin{bmatrix} a^{(3)} & b^{(3)} \end{bmatrix}$ from $\begin{bmatrix} a^{(1)} & b^{(1)} \end{bmatrix}$.
- Skipping all the small $|U^{(i)}(0)|$ yields a small residual error.
- Provably stable (weak stability).

What about speed?

- At most $\max(m, n)$ jumps where $m = \deg a$ and $n = \deg b$.
- One jump of size s costs $O(s^3)$ flops.

$\hookrightarrow$ In most cases, $s \leq 3 \ll m+n$ and we have a $O((m+n)^2)$ algo.

$\hookrightarrow$ Worst-case: one large jump (i.e. $s \sim m+n$) costs $O((m+n)^3)$.

Small and large jumps

$$\left[\begin{array}{ccc} a^{(k)} & b^{(k)} & c^{(k)} \end{array}\right] \quad \Longrightarrow \quad \text{For } r \text{ from } 1 \text{ to } s \text{ solve a } 2r \times 2r \text{ linear system}$$

$$\vdots \qquad\qquad \Downarrow$$

$$\left[\begin{array}{ccc} a^{(k+s)} & b^{(k+s)} & c^{(k+s)} \end{array}\right] := \left[\begin{array}{cc} a^{(k)} & b^{(k)} \end{array}\right] \cdot \left[\begin{array}{ccc} \times & \times & \times \\ \times & \times & \times \end{array}\right]$$

Small and large jumps

$$\left[\begin{array}{ccc} a^{(k)} & b^{(k)} & c^{(k)} \end{array}\right] \quad \Longrightarrow$$

For r from 1 to s solve a $2r \times 2r$ linear system

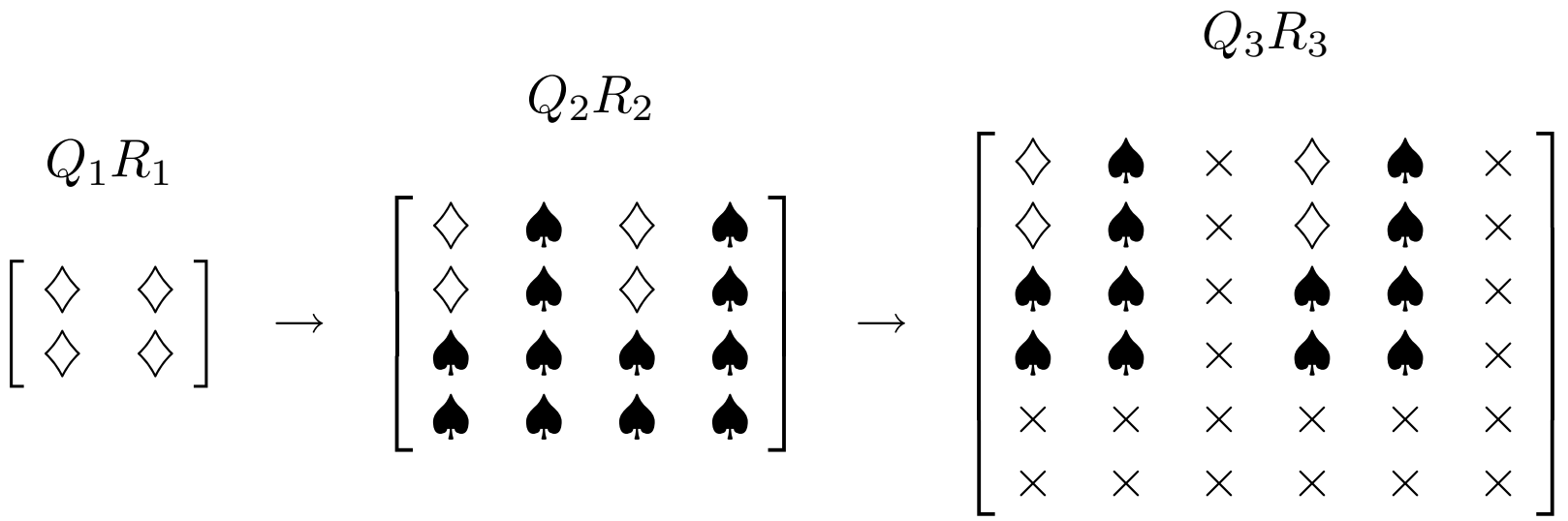
$$\vdots \qquad \Downarrow$$

$$\left[\begin{array}{ccc} a^{(k+s)} & b^{(k+s)} & c^{(k+s)} \end{array}\right] := \left[\begin{array}{cc} a^{(k)} & b^{(k)} \end{array}\right] \cdot \left[\begin{array}{ccc} \times & \times & \times \\ \times & \times & \times \end{array}\right]$$

Extra care needed: $\sum_{r=1}^{s} (2r)^3 = O(s^4)$.

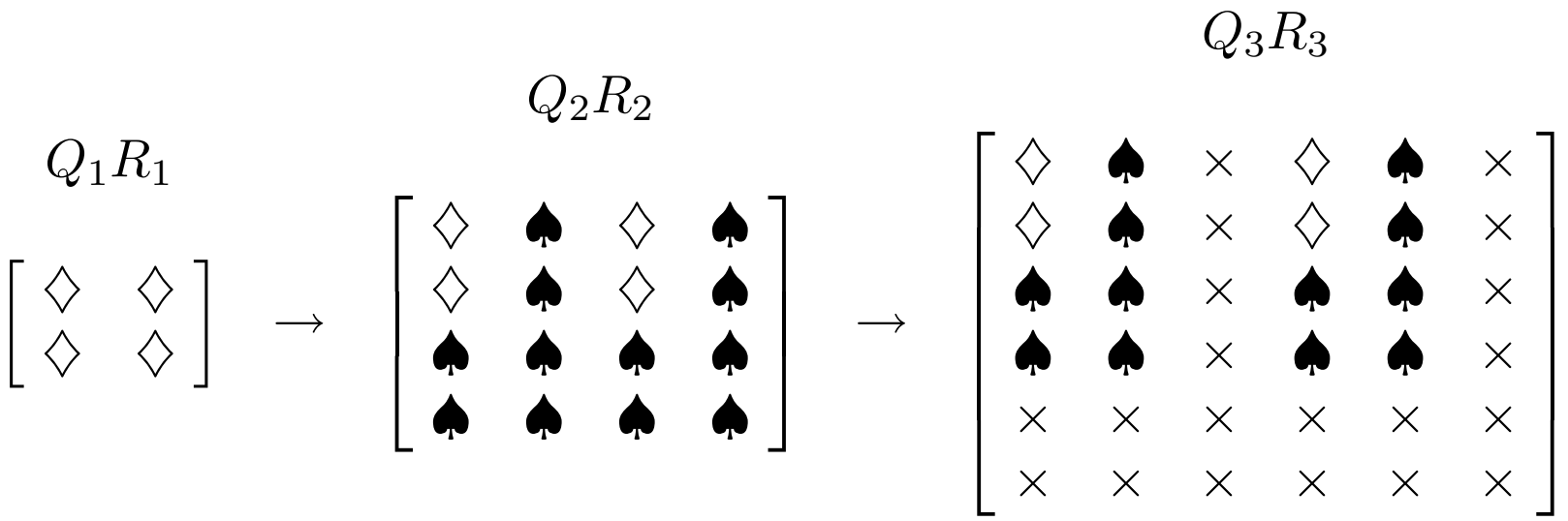
Updating successive QR factorizations

- Jump of size $s = 3$: solve 3 linear systems of order $2r \in \{2, 4, 6\}$.



Updating successive QR factorizations

- Jump of size $s = 3$: solve 3 linear systems of order $2r \in \{2, 4, 6\}$.



- Updating from Q_rR_r to $Q_{r+1}R_{r+1}$ by Givens rotations requires $O((2r)^2)$ flops and $\sum_{r=1}^{s}(2r)^2 = O(s^3)$.

Summary

- Euclidean-like algorithm:
  - Fast in most cases
  - Proven weak (or forward) stability.

Summary

- Euclidean-like algorithm:
  - Fast in most cases
  - Proven weak (or forward) stability.

- Not only a numerical coprimeness test:
  - ϵ-GCD
  - quasi-GCD

  can be guaranteed from the last accepted basis $[a^{(k)}\ b^{(k)}]$.

Two algos, depending on the version

- **Matrix version** [Chandrasekaran & Sayed - 1998]

$$\left[\begin{array}{cc|ccc} a_0 & & b_0 & & \\ a_1 & a_0 & b_1 & b_0 & \\ a_2 & a_1 & b_2 & b_1 & b_0 \\ a_3 & a_2 & & b_2 & b_1 \\ & a_3 & & & b_2 \end{array}\right] \times \begin{bmatrix} s_0 & u_0 \\ s_1 & u_1 \\ t_0 & v_0 \\ t_1 & v_1 \\ t_2 & v_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

- **Polynomial version** [Beckermann & Labahn - 1998]

$$\begin{bmatrix} a(z) & b(z) \end{bmatrix} \times \begin{bmatrix} s(z) & u(z) \\ t(z) & v(z) \end{bmatrix} = \begin{bmatrix} 1 & z^{m+n-1} \end{bmatrix}$$

Two algos, depending on the version

- **Matrix version** [Chandrasekaran & Sayed - 1998]

$$\left[\begin{array}{cc|ccc} a_0 & & b_0 & & \\ a_1 & a_0 & b_1 & b_0 & \\ a_2 & a_1 & b_2 & b_1 & b_0 \\ a_3 & a_2 & & b_2 & b_1 \\ & a_3 & & & b_2 \end{array}\right] \times \begin{bmatrix} s_0 & u_0 \\ s_1 & u_1 \\ t_0 & v_0 \\ t_1 & v_1 \\ t_2 & v_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

- **Polynomial version** [Beckermann & Labahn - 1998]

$$\begin{bmatrix} a(z) & b(z) \end{bmatrix} \times \begin{bmatrix} s(z) & u(z) \\ t(z) & v(z) \end{bmatrix} = \begin{bmatrix} 1 & z^{m+n-1} \end{bmatrix}$$

Fast (direct) solution to structured linear systems

Illustration with $n \times n$ symmetric positive definite Toeplitz matrices:

$$T = \begin{bmatrix} 1 & t_1 & t_2 & t_3 \\ t_1 & 1 & t_1 & t_2 \\ t_2 & t_1 & 1 & t_1 \\ t_3 & t_2 & t_1 & 1 \end{bmatrix} \quad \longrightarrow \quad T = LL^t, \quad L = \begin{bmatrix} * & & & \\ * & * & & \\ * & * & * & \\ * & * & * & * \end{bmatrix}$$

"Compress, operate, decompress" [V. Pan - 2001]

Compress, operate, decompress

Let $Z = \begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ & 1 & 0 & \\ & & 1 & 0 \end{bmatrix}$. Then $T - ZTZ^t$ is equal to

$$\begin{bmatrix} 1 & t_1 & t_2 & t_3 \\ t_1 & 1 & t_1 & t_2 \\ t_2 & t_1 & 1 & t_1 \\ t_3 & t_2 & t_1 & 1 \end{bmatrix} - \begin{bmatrix} & & & \\ & 1 & t_1 & t_2 \\ & t_1 & 1 & t_1 \\ & t_2 & t_1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & t_1 & t_2 & t_3 \\ t_1 & & & \\ t_2 & & & \\ t_3 & & & \end{bmatrix}$$

Compress, operate, decompress

Let $Z = \begin{bmatrix} 0 & & \\ 1 & 0 & \\ & 1 & 0 \\ & & 1 \end{bmatrix}$. Then $T - ZTZ^t$ is equal to

$$\begin{bmatrix} 1 & t_1 & t_2 & t_3 \\ t_1 & 1 & t_1 & t_2 \\ t_2 & t_1 & 1 & t_1 \\ t_3 & t_2 & t_1 & 1 \end{bmatrix} - \begin{bmatrix} & & & \\ & 1 & t_1 & t_2 \\ & t_1 & 1 & t_1 \\ & t_2 & t_1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & t_1 & t_2 & t_3 \\ t_1 & & & \\ t_2 & & & \\ t_3 & & & \end{bmatrix} = GJG^t$$

where $J = \mathrm{diag}[1, -1]$ and $G^t = \begin{bmatrix} 1 & t_1 & t_2 & t_3 \\ 0 & t_1 & t_2 & t_3 \end{bmatrix}^t$.

An $n \times n$ T is uniquely defined by Z, J and an $n \times 2$ generator G.

Compress, operate, decompress

- The Schur complement of A in $\begin{bmatrix} A & B \\ C & D \end{bmatrix}$ is $D - CA^{-1}B$.
- Gaussian elimination:

$$T = \left[\begin{array}{c|c} 1 & v^t \\ \hline v & U \end{array}\right] \longrightarrow \left[\begin{array}{c|c} 1 & 0 \\ \hline v & U - vv^t \end{array}\right] \longrightarrow \cdots \longrightarrow \left[\begin{array}{c|ccc} 1 & & & \\ \hline v & * & & \\ & * & * & \\ & * & * & * \end{array}\right]$$

Iterate, for the Schur complements of T are "as structured as" T.

($i.e.$ each of them is uniquely defined by an equation similar to $T - ZTZ^t = GJG^t$.)

Compress, operate, decompress

Elimination on a generator G rather than on the Toeplitz matrix T:

$$G = \begin{bmatrix} 1 & 0 \\ t_1 & t_1 \\ t_2 & t_2 \\ t_3 & t_3 \end{bmatrix} \xrightarrow{\text{shift}} \begin{bmatrix} 1 & t_1 \\ t_1 & t_2 \\ t_2 & t_3 \end{bmatrix} \xrightarrow{\times\theta^{(2)}} \begin{bmatrix} * & 0 \\ * & * \\ * & * \end{bmatrix} \xrightarrow{\text{shift}} \begin{bmatrix} & \\ * & * \\ * & * \end{bmatrix} \xrightarrow{\times\theta^{(3)}} \cdots \rightarrow \begin{bmatrix} & \\ & \\ * & \end{bmatrix}$$

Compress, operate, decompress

Elimination on a generator G rather than on the Toeplitz matrix T:

$$G = \begin{bmatrix} 1 & 0 \\ t_1 & t_1 \\ t_2 & t_2 \\ t_3 & t_3 \end{bmatrix} \xrightarrow{\text{shift}} \begin{bmatrix} 1 & t_1 \\ t_1 & t_2 \\ t_2 & t_3 \end{bmatrix} \xrightarrow{\times\theta^{(2)}} \begin{bmatrix} * & 0 \\ * & * \\ * & * \end{bmatrix} \xrightarrow{\text{shift}} \begin{bmatrix} & \\ * & * \\ * & * \end{bmatrix} \xrightarrow{\times\theta^{(3)}} \cdots \rightarrow \begin{bmatrix} & \\ & \\ * & \end{bmatrix}$$

At step i, zeroing the $(i, 2)$ entry by $\theta^{(i)}$ yields the ith column of the factor L in $T = LL^t$; shifting down the first column yields a generator for the $(i+1)$st Schur complement.

Compress, operate, decompress

Elimination on a generator G rather than on the Toeplitz matrix T:

$$G = \begin{bmatrix} 1 & 0 \\ t_1 & t_1 \\ t_2 & t_2 \\ t_3 & t_3 \end{bmatrix} \xrightarrow{\text{shift}} \begin{bmatrix} 1 & t_1 \\ t_1 & t_2 \\ t_2 & t_3 \end{bmatrix} \xrightarrow{\times\theta^{(2)}} \begin{bmatrix} * & 0 \\ * & * \\ * & * \end{bmatrix} \xrightarrow{\text{shift}} \begin{bmatrix} * & * \\ * & * \end{bmatrix} \xrightarrow{\times\theta^{(3)}} \cdots \rightarrow \begin{bmatrix} \\ \\ * \end{bmatrix}$$

At step i, zeroing the $(i, 2)$ entry by $\theta^{(i)}$ yields the ith column of the factor L in $T = LL^t$; shifting down the first column yields a generator for the $(i+1)$st Schur complement.

Speed and stability issues

- n Schur elim. steps $\begin{bmatrix} * & * \\ * & * \\ * & * \end{bmatrix} \xrightarrow{\times\theta} \begin{bmatrix} * & 0 \\ * & * \\ * & * \end{bmatrix} \xrightarrow{\text{shift}} \begin{bmatrix} & \\ * & * \\ * & * \end{bmatrix}$ followed by

  2 backward substitutions yield a solution $x = T^{-1}b$ in $O(n^2)$.

- Backward stability [Chandrasekaran & Sayed - 1996]

  implementation of hyperbolic rotations θ is critical.

Extension of this fast and backward stable solver to Sylvester matrices?

$$S = \left[\begin{array}{ccc|ccc} a_0 & & & b_0 & & \\ a_1 & a_0 & & b_1 & b_0 & \\ a_2 & a_1 & & b_2 & b_1 & b_0 \\ a_3 & a_2 & & & b_2 & b_1 \\ & a_3 & & & & b_2 \end{array}\right]$$

is not Toeplitz,
not symmetric,
not positive definite.

Sylvester matrices are quasi-Toeplitz matrices

Let $Z = \begin{bmatrix} 0 & & & & \\ 1 & 0 & & & \\ & 1 & 0 & & \\ & & 1 & 0 & \\ & & & 1 & 0 \end{bmatrix}$. Then $S - ZSZ^t$ is equal to

$$\left[\begin{array}{ccc|ccc} a_0 & & & b_0 & & \\ a_1 & a_0 & & b_1 & b_0 & \\ a_2 & a_1 & & b_2 & b_1 & b_0 \\ a_3 & a_2 & & & b_2 & b_1 \\ & a_3 & & & & b_2 \end{array}\right] - \left[\begin{array}{ccc|ccc} & a_0 & & & b_0 & \\ & a_1 & & a_0 & b_1 & b_0 \\ & a_2 & & a_1 & b_2 & b_1 \\ & a_3 & & a_2 & & b_2 \end{array}\right] = \left[\begin{array}{ccc|c} a_0 & & & b_0 \\ a_1 & & & b_1 \\ a_2 & & & b_2 - a_0 \\ a_3 & & & -a_1 \\ & & & -a_2 \end{array}\right] = GB$$

where G has only 2 columns and B has only 2 rows.

Fast and backward stable quasi-Toeplitz solver

[Chandrasekaran & Sayed - 1998]

- "Regularize": embedding $M = \begin{bmatrix} S^t S & S^t \\ S & \end{bmatrix}$ of order $2(m+n)$.

- M is still structured: $\mathrm{rank}(M - (Z \oplus Z)M(Z \oplus Z)^t) \leq 4$.

Fast and backward stable quasi-Toeplitz solver

[Chandrasekaran & Sayed - 1998]

- "Regularize": embedding $M = \begin{bmatrix} S^t S & S^t \\ S & \end{bmatrix}$ of order $2(m+n)$.

- M is still structured: $\mathrm{rank}(M - (Z \oplus Z)M(Z \oplus Z)^t) \leq 4$.

Compress M by finding a $2(m+n) \times 4$ generator G such that

$$M - (Z \oplus Z)M(Z \oplus Z)^t = GJG^t, \quad J = \mathrm{diag}[1, 1, -1, -1].$$

Fast and backward stable quasi-Toeplitz solver

Operate on G: $2(m+n)$ Schur elimination steps factor M as

$$M = \begin{bmatrix} S^t S & S^t \\ S & \end{bmatrix} = \begin{bmatrix} R^t & \\ Q & \Delta \end{bmatrix} \times \begin{bmatrix} R & Q^t \\ & -\Delta^t \end{bmatrix}$$

where $\Delta^{-1}Q$ is numerically orthogonal (whereas Q is not).

Fast and backward stable quasi-Toeplitz solver

Operate on G: $2(m+n)$ Schur elimination steps factor M as

$$M = \begin{bmatrix} S^t S & S^t \\ S & \end{bmatrix} = \begin{bmatrix} R^t & \\ Q & \Delta \end{bmatrix} \times \begin{bmatrix} R & Q^t \\ & -\Delta^t \end{bmatrix}$$

where $\Delta^{-1}Q$ is numerically orthogonal (whereas Q is not).

Decompress: writing $S = \Delta(\Delta^{-1}Q)R$, a fast and backward stable solution to $Sx = b$ can be computed as

$$x = R^{-1}Q^t\Delta^{-t}\Delta^{-1}b.$$

Conclusions

Approach	Time	Space	Stability	ϵ-GCD
polynomial	quadratic / cubic	linear / quadratic	forward	yes
matrix	quadratic	quadratic	backward	no

Conclusions

Approach	Time	Space	Stability	ϵ-GCD
polynomial	quadratic / cubic	linear / quadratic	forward	yes
matrix	quadratic	quadratic	backward	no

In progress - joint work with Robert M. Corless (U. Western Ontario), Lihong Zhi (U. Western Ontario).

Conclusions

Approach	Time	Space	Stability	ϵ-GCD
polynomial	quadratic / cubic	linear / quadratic	forward	yes
matrix	quadratic	quadratic	backward	no

In progress - joint work with Robert M. Corless (U. Western Ontario), Lihong Zhi (U. Western Ontario).

Slides and references available at `www.ens-lyon.fr/~cpjeanne`