On the complexity of computing determinants and other challenges in symbolic computation

Erich Kaltofen
North Carolina State University
www.kaltofen.net

Matrix determinant definition

$$\det(Y) = \det\left(\begin{bmatrix} y_{1,1} & \cdots & y_{1,n} \\ y_{2,1} & \cdots & y_{2,n} \\ \vdots & & \vdots \\ y_{n,1} & \cdots & y_{n,n} \end{bmatrix}\right) = \sum_{\sigma \in S_n} \left(\operatorname{sign}(\sigma) \prod_{i=1}^{n} y_{i,\sigma(i)} \right),$$

where $y_{i,j}$ are from an *arbitrary commutative ring*,
and S_n is the set of all permutations on $\{1, 2, \ldots, n\}$.

Interesting rings: $\mathbb{Z}$, $\mathbb{K}[x_1, \ldots, x_n]$, $\mathbb{K}[x]/(x^n)$

Fast matrix multiplication

Strassen's [1969] $O(n^{2.81})$ matrix multiplication algorithm

$$
\begin{array}{l|l}
m_1 \leftarrow (a_{1,2} - a_{2,2})(b_{2,1} - b_{2,2}) & \\
m_2 \leftarrow (a_{1,1} + a_{2,2})(b_{1,1} + b_{2,2}) & \\
m_3 \leftarrow (a_{1,1} - a_{2,1})(b_{1,1} + b_{1,2}) & \\
m_4 \leftarrow (a_{1,1} + a_{1,2})b_{2,2}) & a_{1,1}b_{1,1} + a_{1,2}b_{2,1} = m_1 + m_2 - m_4 + m_6 \\
m_5 \leftarrow a_{1,1}(b_{1,2} - b_{2,2}) & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} = m_4 + m_5 \\
m_6 \leftarrow a_{2,2}(b_{2,1} - b_{1,1}) & a_{2,1}b_{1,1} + a_{2,2}b_{2,1} = m_6 + m_7 \\
m_7 \leftarrow (a_{2,1} + a_{2,2})b_{1,1}) & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} = m_2 - m_3 + m_5 - m_7
\end{array}
$$

Problems reducible to matrix multiplication:

linear system solving, determinants [Bunch and Hopcroft 1974],...

Coppersmith and Winograd [1990]: $O(n^{2.38})$

Why the determinant complexity is important

Theorem [Giesbrecht 1992]
Suppose you have a Monte Carlo randomized algorithm on a random access machine that can compute the determinant of an $n \times n$ matrix in $D(n)$ arithmetic operations.

Then you have a Monte Carlo randomized algorithm on a random access machine that can multiply two $n \times n$ matrices in $O(D(n))$ arithmetic operations.

No proof is known for Las Vegas or deterministic algorithms.

Life after Strassen: 1. black box matrix concept

$A \in \mathbb{F}^{n \times n}$ singular

$\mathbb{F}$ an arbitrary, e.g., finite field

Perform linear algebra operations, e.g., $A^{-1}b$ [Wiedemann 86] with

$O(n)$	black box calls and
$n^2(\log n)^{O(1)}$	arithmetic operations in $\mathbb{F}$ and
$O(n)$	intermediate storage for field elements

Black box model is useful for dense, structured matrices

$$\begin{bmatrix} 1 & \cdots & & \cdots & \frac{1}{n} \\ & & \vdots & & \\ & & \frac{1}{i+j-1} & & \\ & & \vdots & & \\ \frac{1}{n} & \cdots & & \cdots & \frac{1}{2n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ \\ \vdots \\ \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \\ \vdots \\ \\ b_n \end{bmatrix}$$

(Hilbert matrix)

Savings may be in space, not time: $O(1)$ vs. $O(n^2)$.

Idea for Wiedemann's algorithm

$A \in \mathbb{F}^{n \times n}$, $\mathbb{F}$ a (possibly finite) field

$\phi^A(\lambda) = c'_0 + \cdots + c'_m \lambda^m \in \mathbb{F}[\lambda]$ **minimum polynomial** of A

$\forall\, u, v \in \mathbb{F}^n\colon\ \forall\, j \geq 0\colon$

$$u^{Tr} A^j \phi^A(A) v = 0$$

$$\Updownarrow$$

$$c'_0 \cdot \underbrace{u^{Tr} A^j v}_{a_j} + c'_1 \cdot \underbrace{u^{Tr} A^{j+1} v}_{a_{j+1}} + \cdots + c'_m \cdot \underbrace{u^{Tr} A^{j+m} v}_{a_{j+m}} = 0$$

$$\Updownarrow$$

$\{a_0, a_1, a_2, \ldots\}$ is generated by a linear recursion

Theorem [Wiedemann 1986]: *For* ***random*** $u, v \in \mathbb{F}^n$,
a linear generator for $\{a_0, a_1, a_2, \ldots\}$ *is one for* $\{I, A, A^2, \ldots\}$.

$$\forall\, j \geq 0\colon\ c_0 a_j + c_1 a_{j+1} + \cdots + c_d a_{j+d} = 0$$

$$\Downarrow \text{ (with high probability)}$$

$$c_0 A^j v + c_1 A^{j+1} v + \cdots + c_d A^{j+d} v = \mathbf{0}$$

$$\Downarrow \text{ (with high probability)}$$

$$c_0 A^j + c_1 A^{j+1} + \cdots + c_d A^{j+d} = \mathbf{0}$$

that is, with high probability $\phi^A(\lambda)$ divides $c_0 + c_1\lambda + \cdots + c_d\lambda^d$

Algorithm homogeneous Wiedemann

Input: $A \in \mathbb{F}^{n \times n}$ singular
Output: $w \neq \mathbf{0}$ such that $Aw = \mathbf{0}$

Step W1: Pick random $u, v \in \mathbb{F}^n$; $\quad b \leftarrow Av$;
for $i \leftarrow 0$ **to** $2n-1$ **do** $a_i \leftarrow u^{Tr} A^i b$.
(Requires $2n$ black box calls.)

Step W2: Compute a linear recurrence generator for $\{a_i\}$,
$c_\ell \lambda^\ell + c_{\ell+1}\lambda^{\ell+1} + \cdots + c_d \lambda^d, \quad \ell \geq 0, d \leq n, c_\ell \neq 0.$

Step W3: $\widehat{w} \leftarrow c_\ell v + c_{\ell+1} Av + \cdots + c_d A^{d-\ell} v$;
(With high probability $\widehat{w} \neq 0$ and $A^{\ell+1}\widehat{w} = 0$.)
Compute first k with $A^k \widehat{w} = 0$; **return** $w \leftarrow A^{k-1}\widehat{w}$.
(Requires $\leq n$ black box calls.)

Step W2 detail

Coefficients $c_0, \ldots, c_n$ can be found by computing a non-trivial solution to the Toeplitz system

$$\begin{bmatrix} a_n & a_{n-1} & \cdots & & a_1 & a_0 \\ a_{n+1} & a_n & & & a_2 & a_1 \\ \vdots & a_{n+1} & \ddots & & \vdots & a_2 \\ & \vdots & & & & \vdots \\ a_{2n-2} & & & \ddots & a_{n-1} & \\ a_{2n-1} & a_{2n-2} & \cdots & & a_n & a_{n-1} \end{bmatrix} \cdot \begin{bmatrix} c_n \\ c_{n-1} \\ c_{n-2} \\ \vdots \\ \\ c_0 \end{bmatrix} = \mathbf{0}$$

or by the Berlekamp/Massey algorithm.

Cost: $O(n(\log n)^2 \operatorname{loglog} n)$ arithmetic ops.

Flurry of recent results

Lambert [96], Teitelbaum [98], Eberly & Kaltofen [97]	relationship of Wiedemann and Lanczos approach
Villard [97]	analysis of **block** Wiedemann algorithm
Giesbrecht [97] and Mulders & Storjohann [99]	computation of **diophantine** solutions
Giesbrecht, Lobo & Saunders [98]	certificates for inconsistency
Chen, Eberly, Kaltofen, Saunders, Villard & Turner [2K]	butterfly network, sparse and diagonal preconditioners
Villard [2K] & Storjohann [01]	characteristic polynomial
Kaltofen & Villard [2K]	fast algorithm for determinant of a **dense** integer matrix
Turner [01]	determinant-preserving preconditioners

Life after Strassen: 2. bit complexity

Linear system solving $x = A^{-1}b$ where $A \in \mathbb{Z}^{n\times n}$ and $b \in \mathbb{Z}^n$:

With Strassen and Chinese remaindering [McClellan 1973]:

Step 1: For prime numbers $p_1, \ldots, p_k$ Do
Solve $Ax^{[j]} \equiv b \pmod{p_j}$ where $x^{[j]} \in \mathbb{Z}/(p_j)$

Step 2: Chinese remainder $x^{[1]}, \ldots, x^{[k]}$ to $A\overline{x} \equiv b \pmod{p_1 \cdots p_k}$

Step 3: Recover denominators of x_i by continued fractions of $\dfrac{\overline{x}_i}{p_1 \cdots p_k}$.

Length of integers: $k = (n \ \max\{\log \|A\|, \log \|b\|\})^{1+o(1)}$

Bit complexity: $n^{3.38} \ \max\{\log \|A\|, \log \|b\|\}^{1+o(1)}$

With Hensel lifting [Moenck and Carter 1979, Dixon 1982]:

Step 1: For $j = 0, 1, \ldots, k$ and a prime p Do

Compute $\overline{x}^{[j]} = x^{[0]} + px^{[1]} + \cdots + p^j x^{[j]} \equiv x \pmod{p^{j+1}}$

1.a. $\widehat{b}^{[j]} = \dfrac{b - A\overline{x}^{[j-1]}}{p^j} = \dfrac{\widehat{b}^{[j-1]} - Ax^{[j-1]}}{p}$

1.b. $x^{[j]} \equiv A^{-1}\widehat{b}^{[j]} \pmod{p}$ reusing $A^{-1} \bmod p$

Step 2: Recover denominators of x_i by continued fractions of $\dfrac{\overline{x}_i^{[k]}}{p^k}$.

With classical matrix arithmetic:

Bit complexity of 1.a: $n(n \max\{\log \|A\|, \|b\|\})^{1+o(1)} + n^2(\log \|A\|)^{1+o(1)}$

Total bit complexity: $(n^3 \max\{\log \|A\|, \log \|b\|\})^{1+o(1)}$

Diophantine solutions
by Giesbrecht, Mulders&Storjohann:
Find several rational solutions.

$$A(\tfrac{1}{2}x^{[1]}) = b, \quad x^{[1]} \in \mathbb{Z}^n$$
$$A(\tfrac{1}{3}x^{[2]}) = b, \quad x^{[2]} \in \mathbb{Z}^n$$
$$\gcd(2,3) = 1 = 2 \cdot 2 - 1 \cdot 3$$
$$A(2x^{[1]} - x^{[2]}) = 4b - 3b = b$$

$\Longrightarrow$Can compute **integral** solutions of **sparse** linear systems.

Bit complexity of the determinant

With Chinese remaindering: $(n \log \|A\|)^{1+o(1)}$ times matrix multiplication complexity.

Sign of the determinant [Clarkson 92]: $n^{4+o(1)}$ if matrix is ill-conditioned.

Using denominators of linear system solutions [Abbott, Bronstein, Mulders 1999]: fast when large first invariant factor.

Using fast Smith form method $n^{3.5+o(1)}(\log \|A\|)^{2.5+o(1)}$ [Eberly, Giesbrecht, Villard 2000]

Baby steps/giant steps algorithm [Kaltofen 1992/2000]

Wiedemann preconditions A and chooses random u and v; then $\det(\lambda I - A) =$ minimal recurrence polynomial of $\{a_i\}_{i=0,1,\ldots 2n-1}$.

Detail of sequence $a_i = u^T A^i v$ computation

Let $r = \lceil \sqrt{2n} \rceil$ and $s = \lceil 2n/r \rceil$.

Substep 1. For $j = 1, 2, \ldots, r-1$ Do $v^{[j]} \leftarrow A^j v$;

Substep 2. $Z \leftarrow A^r$;
[$O(n^3)$ operations; integer length $(\sqrt{n}\ \log \|A\|)^{1+o(1)}$]

Substep 3. For $k = 1, 2, \ldots, s$ Do ${u^{[k]}}^T \leftarrow u^T Z^k$;
[$O(n^{2.5})$ operations; integer length $(n\ \log \|A\|)^{1+o(1)}$]

Substep 4. For $j = 0, 1, \ldots, r-1$ Do
For $k = 0, 1, \ldots, s$ Do $a_{kr+j} \leftarrow \langle u^{[k]}, v^{[j]} \rangle$.

The state-of-the-art [Kaltofen & Villard 2001]

Theorem 1

The determinant of an integer matrix can be computed in $O(n^{2.698}(\log \|A\|)^{1+o(1)})$ *bit operations.*

Theorem 2

The determinant and adjoint of a matrix over a commutative ring can be computed with $O(n^{2.698})$ *ring additions, subtractions and multiplications.*

Problem 1 (from my 3ECM 2000 talk)

Improve the bit complexity of algorithms for the determinant, resultant, linear system solution, Toeplitz systems, over the integers.

Coppersmith's 1992 blocking

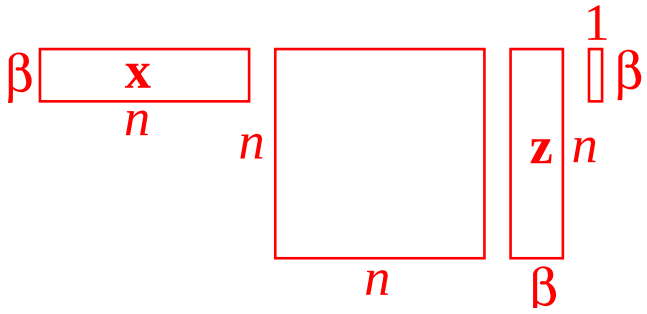
Use of the block vectors $\mathbf{x} \in \mathbb{F}^{n\times\beta}$ in place of u

$\mathbf{z} \in \mathbb{F}^{n\times\beta}$ in place of v

$$\mathbf{a}_i = \mathbf{x}^{Tr} A^{i+1} \mathbf{z} \in \mathbb{F}^{\beta\times\beta}, \quad 0 \le i < 2n/\beta + 2.$$

Find a **vector** polynomial $c_\ell \lambda^\ell + \cdots + c_d \lambda^d \in \mathbb{F}^\beta[\lambda]$, $d = \lceil n/\beta \rceil$:

$$\forall\, j \ge 0\colon \quad \sum_{i=\ell}^{d} \mathbf{a}_{j+i} c_i = \sum_{i=\ell}^{d} \mathbf{x}^{Tr} A^{i+j}\, A\mathbf{z} c_i = \mathbf{0} \in \mathbb{F}^{\beta\times\beta}$$

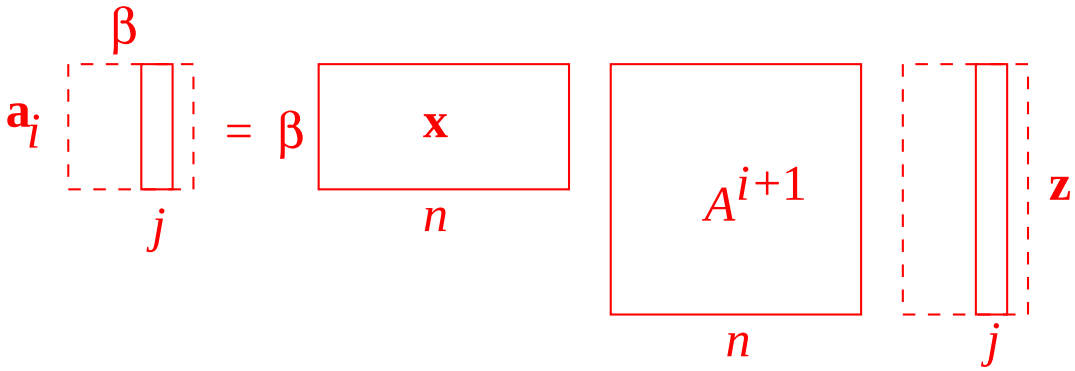


Then, analogously to before, with high probability

$$\widehat{w} = \sum_{i=\ell}^{d} A^{i-\ell}\, \mathbf{z} c_i \neq \mathbf{0}, \quad A^{\ell+1}\widehat{w} = \sum_{i=\ell}^{d} A^i A\mathbf{z} c_i = \mathbf{0} \in \mathbb{F}^n$$

Advantages of blocking

1. Parallel coarse- and fine-grain implementation



   The j^{th} processor computes the j^{th} column of the sequence of (small) matrices.

2. Faster sequential running time:
   a. multiple solutions [Coppersmith; Montgomery 1994];
   b. $1+\epsilon$ matrix times vector ops [Kaltofen 1995];
   c. determinant [Kaltofen & Villard 2000];
   d. charpoly of sparse matrix [Villard & Storjohann 2001]

Analysis of blocking

All vector polynomials that generate $\{\mathbf{a}_i\}$ form a **module** over $\mathbb{F}[\lambda]$.

β vectors of minimal degree determinant form an $\mathbb{F}[\lambda]$-basis

A matrix canonical (Popov, Hermite) version of the basis defines a unique minimal polynomial $\mathbf{c}_0 + \mathbf{c}_1\lambda + \cdots + \mathbf{c}_d\lambda^d \in \mathbb{F}^{\beta\times\beta}[\lambda]$

From

$$(I - \lambda A)^{-1} = I + A\lambda + A^2\lambda^2 + \cdots$$

$$\mathbf{x}^{Tr}(I - \lambda A)^{-1}\mathbf{y}(\mathbf{c}_d + \cdots + \mathbf{c}_0\lambda^d) = R(\lambda) \in \mathbb{F}[\lambda]^{\beta\times\beta}$$

we obtain a matrix Padé approximation ("realization")

$$\mathbf{x}^{Tr}(I - \lambda A)^{-1}\mathbf{y} = \textstyle\sum_i \mathbf{a}_i\lambda^i = R(\lambda)(\mathbf{c}_d + \cdots + \mathbf{c}_0\lambda^d)^{-1}$$

Theorem

Let $A \in \mathbb{F}^{n \times n}$, $\mathbf{x} \in \mathbb{F}^{n \times \beta}$, $\mathbf{y} \in \mathbb{F}^{n \times \beta}$

$s_1, \ldots, s_\phi$ *denote all invariant factors of* $\lambda I - A$.

Then for all i, *the* i*-th invariant factors of* $\mathbf{c}_d + \cdots + \mathbf{c}_0 \lambda^d$ ***divide*** s_i.

Furthermore, for random projection matrices $\mathbf{x} \in S^{n \times \beta}$, $\mathbf{y} \in S^{n \times \beta}$ *for all* i *the* i*-th invariant factors of* $\mathbf{c}_d + \cdots + \mathbf{c}_0 \lambda^d$ ***are equal*** *to* s_i *with probability* $\geq 1 - \dfrac{2n}{|S|}$

Advantages of blocking continued:

3. Captures multiple invariant factors
   $\Longrightarrow$ better probability of success [Villard 1997]

Computation of the matrix linear generator

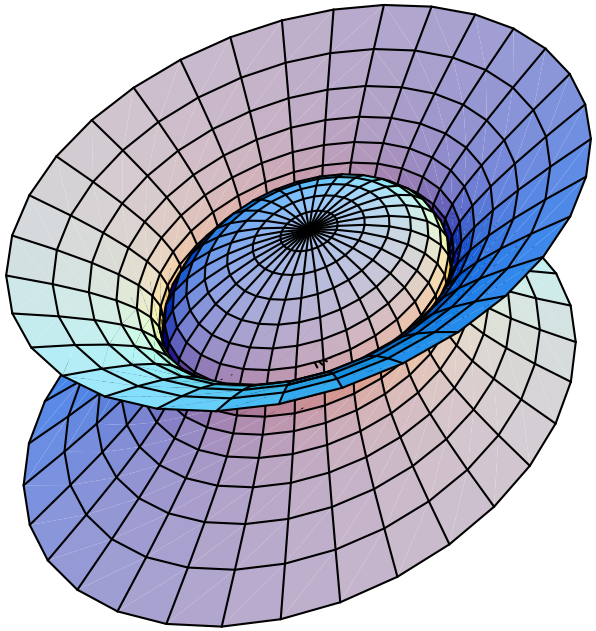
Explicitly in Popov form by block Berlekamp/Massey algorithm [Coppersmith 1994, Thomé 2001] or implicitly in a block Lanczos version

Explicitly by a power Hermite-Padé approximation
[Beckermann & Labahn 1994]

By a block Toeplitz solver [Kaltofen 1995]

Factorization of nearby polynomials over the complex numbers

$$81x^4 + 16y^4 - 648z^4 + 72x^2y^2 - 648x^2 - 288y^2 + 1296 = 0$$



$$(9x^2 + 4y^2 + 18\sqrt{2}\,z^2 - 36)(9x^2 + 4y^2 - 18\sqrt{2}\,z^2 - 36) = 0$$

$$81x^4 + 16y^4 - 648.003z^4 + 72x^2y^2 + .002x^2z^2 + .001y^2z^2$$
$$- 648x^2 - 288y^2 - .007z^2 + 1296 = 0$$

Problem 2 *[Kaltofen LATIN'92]*
Given is a polynomial $f(x,y) \in \mathbb{Q}[x,y]$ *and* $\epsilon \in \mathbb{Q}$.

Decide in polynomial time in the degree and coefficient size if there is a factorizable $\tilde{f}(x,y) \in \mathbb{C}[x,y]$ *with*

$$\|f - \tilde{f}\| \leq \epsilon \text{and } \deg(\tilde{f}) \leq \deg(f),$$

for a reasonable coefficient vector norm $\|\cdot\|$.

Theorem *[Hitz, Kaltofen, Lakshman ISSAC'99]*
We can compute in polynomial time in the degree and coefficient size if there is an $\tilde{f}(x,y) \in \mathbb{C}[x,y]$ *with a factor of a* ***constant degree*** *and* $\|f - \tilde{f}\|_2 \leq \epsilon$.

Numerical algorithms

Conclusion on my exact algorithm [JSC 1985]:

"*D. Izraelevitz at Massachusetts Institute of Technology has already implemented a version of algorithm 1 using complex floating point arithmetic. Early experiments indicate that the linear systems computed in step (L) tend to be* ***numerically ill-conditioned.*** *How to overcome this numerical problem is an important question which we will investigate.*"

Sasaki *et al.* [Japan J. Indust. Applied Math, 1991]: Combine sums of powers of roots to low degree polys

Stetter, Huang, Wu and Zhi [ISSAC'2K]: Hensel lift factor combinations numerically and eliminate extraneous factors early

Corless, Giesbrecht, Kotsieras, van Hoeij, Watt [ISSAC'01]: sample curve by points and interpolate

Will our systems guarantee their answers?

Maple 6 allows calls to NAG numeric library routines

Basic polynomial algorithms with floating point coefficients are under development

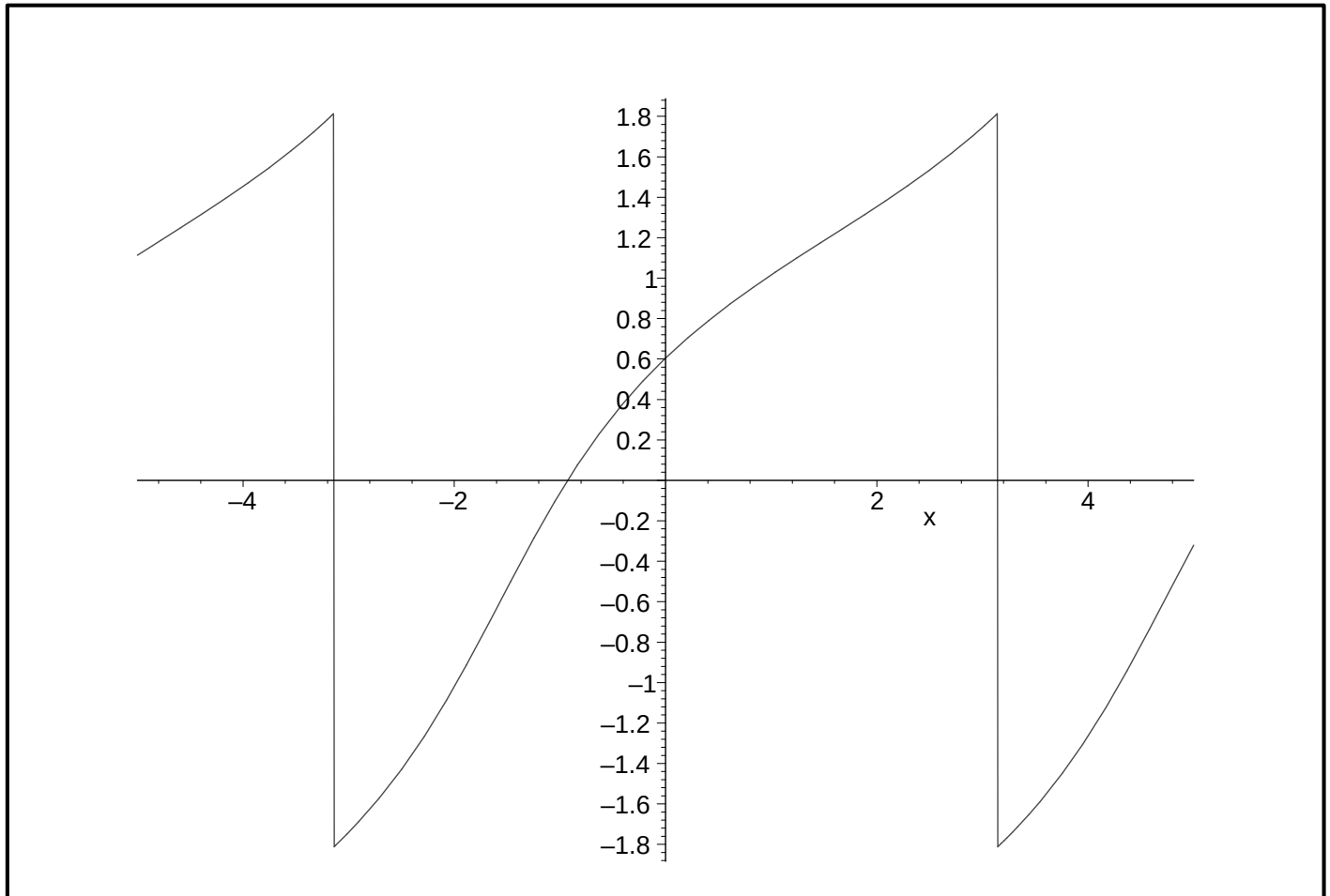
```
> # Example by Corless and Jeffrey
> f := 1/(sin(x) + 2);
```

$$f := \frac{1}{\sin(x) + 2}$$

```
> g := int(f, x);
```

$$g := \frac{2}{3}\sqrt{3}\arctan(\frac{1}{3}(2\tan(\frac{1}{2}x) + 1)\sqrt{3})$$

```
> plot(g, x=-5..5);
```



Early termination strategies

Early termination in Newton interpolation

For $i \leftarrow 1, 2, \ldots$ *Do*

Pick ***random*** p_i *and from* $f(p_i)$

compute

$$f^{[i]}(x) \leftarrow c_0 + c_1(x - p_1) + c_2(x - p_1)(x - p_2) + \cdots$$
$$\equiv f(x) \pmod{(x - p_1) \cdots (x - p_{i+1})}$$

If $c_i = 0$ *stop.*

End For

Threshold η:
In order to obtain a better probability, we require $c_i = 0$ more than once before terminating.

Kaltofen and Wen-shin Lee [2000] have developed early termination strategies for sparse interpolation algorithm.

Wen-shin Lee has a new heuristic to find a sparse shift, i.e., a number a such that $f(y-a)$ has the minimum terms of the form y^{e_i}.

Problem 3 [from my 2000 3ECM talk]
Provide reasonable correctness specifications for our systems in the presence of floating point numbers, randomizations, and multivalued functions.