Fast High-Resolution Drawing of Algebraic Curves

Nuwan Herath Mudiyanselage
Guillaume Moroz, Marc Pouget

Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

ISSAC 2022
July 4 - 7, 2022

Implicit curve drawing

Implicit curve drawing problem

Discrete representation of an algebraic curve on a fixed grid

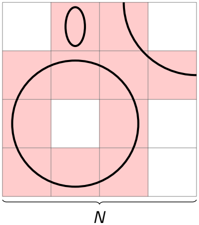
- **Input**: bivariate polynomial P of partial degree d, resolution N

$$P(x,y) = \sum_{i=0}^{d} \sum_{j=0}^{d} a_{i,j} x^i y^j$$

Implicit curve defined as the solution set

$$\{(x,y) \in \mathbb{R}^2 \mid P(x,y) = 0\}$$

- **Output**: drawing (set of pixels)



Goal: fast high-resolution drawing of high degree algebraic curves

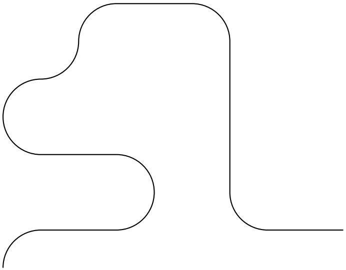
- $d \approx 100$
- $N \approx 1000$

Previous work: Marching squares, adaptative subdivision, CAD

Marching squares

The idea

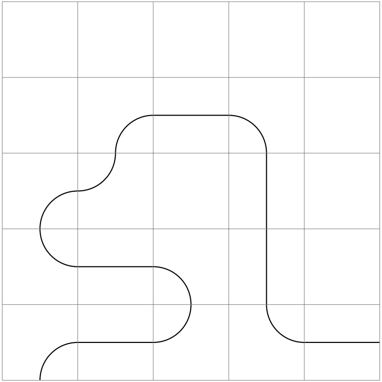
2D variant of the widely used Marching cubes algorithm
Implicit equation: $P(x, y) = 0$



Marching squares

The idea

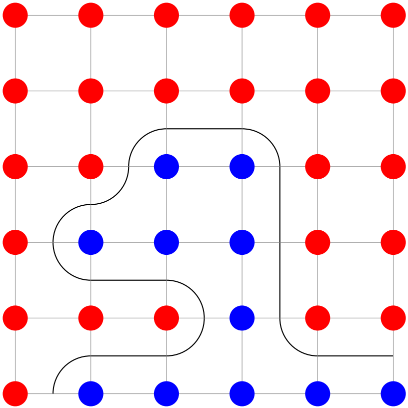
2D variant of the widely used Marching cubes algorithm
Implicit equation: $P(x, y) = 0$



Marching squares

The idea

2D variant of the widely used Marching cubes algorithm
Implicit equation: $P(x, y) = 0$

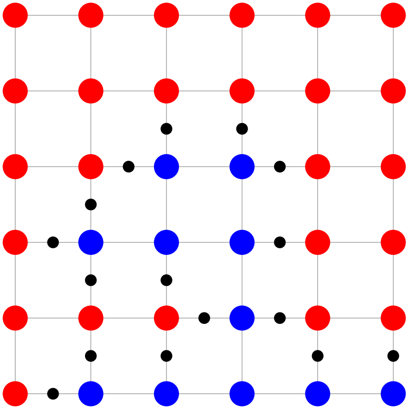


Marching squares

The idea

2D variant of the widely used Marching cubes algorithm

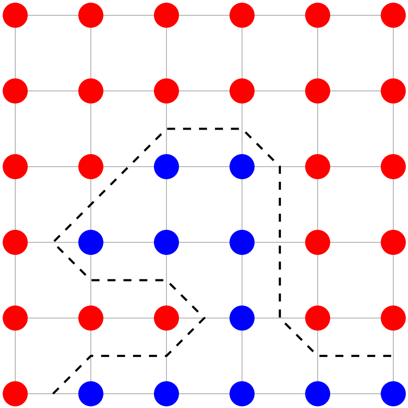
Implicit equation: $P(x, y) = 0$



Marching squares

The idea

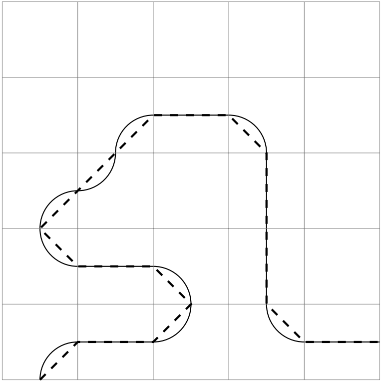
2D variant of the widely used Marching cubes algorithm
Implicit equation: $P(x, y) = 0$



Marching squares

The idea

2D variant of the widely used Marching cubes algorithm
Implicit equation: $P(x, y) = 0$



Marching squares

Complexity

Complexity (number of elementary operations)
Naive evaluation

$$O(d^2 N^2)$$

- d partial degree
- N resolution of the grid

[redacted]

With partial evaluation of $P(x, y)$, assuming $d < N$

$$O(dN^2)$$

Slow for high resolutions. . .

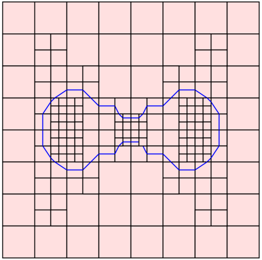
Methods providing topological correctness

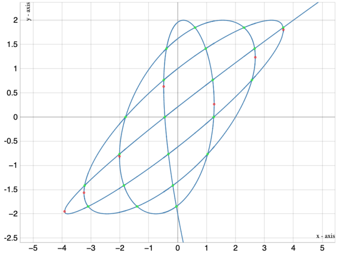
Adaptative 2D subdivision and interval arithmetic

- [Snyder, 1992]
- [Plantinga & Vegter, 2004]
- [Burr et al., 2008]
- [Lin & Yap, 2011]
- ...

Cylindrical algebraic decomposition (CAD)

- [Gonzalez-Vega & Necula, 2002]
- [Eigenwillig et al., 2007]
- [Alberti et al., 2008]
- [Cheng et al., 2009]
- [Kobel & Sagraloff, 2015]
- [Diatta et al., 2018]
- ...

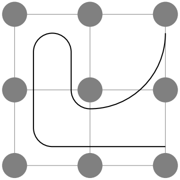




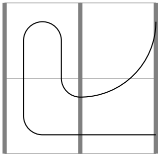
Our approach: guaranteed intersection with the grid

Our approach

Evaluation on intersections of the grid



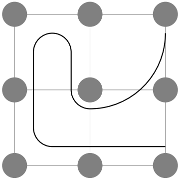
Evaluation along fibers



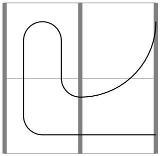
⇒ Make it fast and provide some guarantees

Our approach

~~Evaluation on intersections of the grid~~



Evaluation along fibers

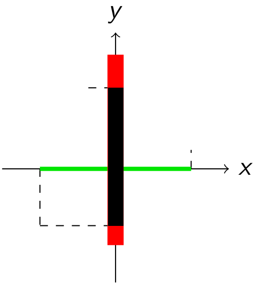


$\Rightarrow$ Make it fast and provide some guarantees

Interval arithmetic

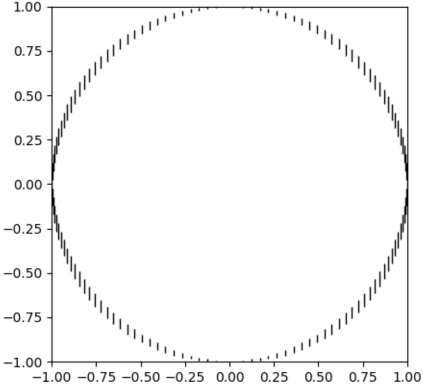
$\Box p$ is an interval extension of p if on an interval I it verifies

$$\Box p(I) \supseteq p(I).$$



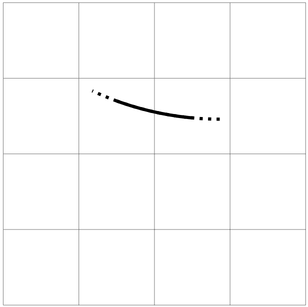
An example

$$x^2 + y^2 - 1 = 0$$



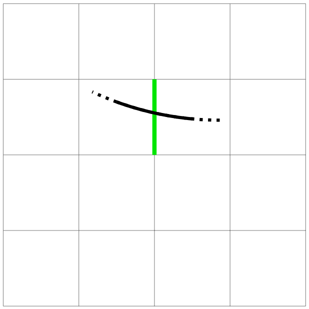
Resolution $N = 64$

Intersection detection



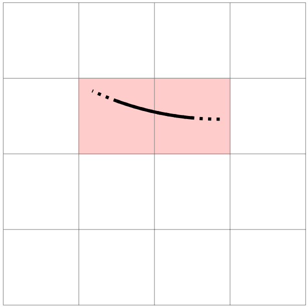
Intersection detection

- Detect a crossing between two consecutive nodes of the grid



Intersection detection

- Detect a crossing between two consecutive nodes of the grid
- Light the adjacent pixels



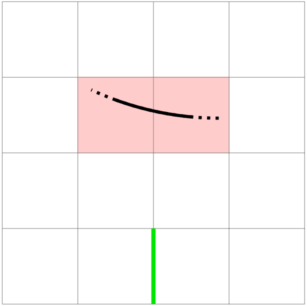
Intersection detection

- Detect a crossing between two consecutive nodes of the grid
- Light the adjacent pixels
- Exclude a segment S if

$$0 \notin \Box p(S) + [-E, E]$$

where

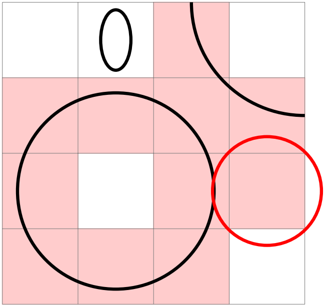
$$\begin{cases} p(y) & = \sum_{i=0}^{d} a_i y^i \\ E & = d^2 \|a\|_\infty (d^2 + N \log_2(N)) O(u) \end{cases}$$



Intersection detection

Some incorrect pixels:

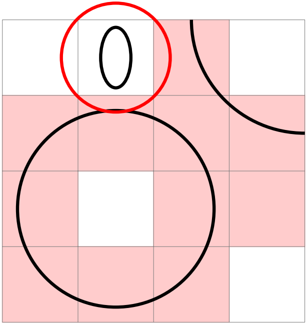
- False positive when the evaluation on an edge of a pixel is close to zero



Intersection detection

Some incorrect pixels:

- False positive when the evaluation on an edge of a pixel is close to zero
- False negative when a connected component lies inside of a pixel



Fast multipoint evaluation at Chebyshev nodes

A prerequisite to fast multipoint evaluation

Chebyshev polynomials

[redacted]

The Chebyshev polynomials (T_k) verify $\forall k \in \mathbb{N}$, $T_k(\cos\theta) = \cos(k\theta)$.

[redacted]

The first three Chebyshev polynomials

$$\cos(0 \cdot \theta) = 1 \qquad T_0 = 1$$

$$\cos(1 \cdot \theta) = \cos(\theta) \qquad T_1 = X$$

$$\cos(2 \cdot \theta) = 2\cos(\theta)^2 - 1 \qquad T_2 = 2X^2 - 1$$

A prerequisite to fast multipoint evaluation

Chebyshev polynomials

[redacted]

The Chebyshev polynomials (T_k) verify $\forall k \in \mathbb{N}, T_k(\cos\theta) = \cos(k\theta)$.

[redacted]

An arbitrary polynomial p of degree d can be written in terms of the Chebyshev polynomials:

$$p(x) = \sum_{k=0}^{d} \alpha_k T_k(x).$$

A prerequisite to fast multipoint evaluation

Chebyshev polynomials

[redacted]

The Chebyshev polynomials (T_k) verify $\forall k \in \mathbb{N}$, $T_k(\cos\theta) = \cos(k\theta)$.

[redacted]

An arbitrary polynomial p of degree d can be written in terms of the Chebyshev polynomials:

$$p(x) = \sum_{k=0}^{d} \alpha_k T_k(x).$$

[redacted]

For $N \in \mathbb{N}$, a polynomial p of degree d can be evaluated on the Chebyshev nodes $(c_n)_{0 \leq n \leq N-1}$ using the IDCT*:*

$$(p(c_n))_{0 \leq n \leq N-1} = \frac{1}{2}(\alpha_0, \ldots, \alpha_0) + \text{IDCT}((\alpha_k)_{0 \leq k \leq N-1}).$$

A prerequisite to fast multipoint evaluation

Chebyshev nodes

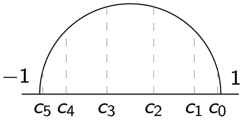
[redacted]

For $N \in \mathbb{N}$, the Chebyshev nodes are

$$c_n = \cos\left(\frac{2n+1}{2N}\pi\right), \ n = 0, \dots, N-1.$$

They are the roots of T_N.

For $N = 6$



DFT / DCT

Discrete Fourier Tranform (DFT): $x_n \to \alpha_k$

$$\alpha_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk}$$

Discrete Cosine Transform (DCT-II): $x_n \to \alpha_k$

$$\alpha_k = \sum_{n=0}^{N-1} x_n \ \cos\left[\frac{\pi(2n+1)k}{2N}\right]$$

$\Rightarrow$ Fast thanks to the FFT algorithm $O(N \log_2 N)$ [Makhoul, 1980]

Multipoint evaluation with the IDCT

Inverse Discrete Cosine Transform (IDCT): $\alpha_k \to x_n$

$$x_n = \frac{1}{2}\alpha_0 + \sum_{k=1}^{N-1} \alpha_k \ \cos\left[\frac{\pi k(2n+1)}{2N}\right]$$

$$p(c_n) = \sum_{k=0}^{N-1} \alpha_k T_k\left(\cos\left(\frac{2n+1}{2N}\pi\right)\right) = \sum_{k=0}^{N-1} \alpha_k \cos\left[\frac{\pi k(2n+1)}{2N}\right]$$

Multipoint evaluation with the IDCT

Inverse Discrete Cosine Transform (IDCT): $\alpha_k \to x_n$

$$x_n = \frac{1}{2}\alpha_0 + \sum_{k=1}^{N-1} \alpha_k \cos\left[\frac{\pi k(2n+1)}{2N}\right]$$

$$p(c_n) = \frac{1}{2}\alpha_0 + \frac{1}{2}\alpha_0 + \sum_{k=1}^{N-1} \alpha_k \cos\left[\frac{\pi k(2n+1)}{2N}\right]$$

$$(p(c_n))_{0\le n\le N-1} = \frac{1}{2}(\alpha_0, \ldots, \alpha_0) + \text{IDCT}((\alpha_k)_{0\le k\le N-1})$$

Error of the IDCT

[Makhoul, 1980] and [Brisebarre et al., 2020, Theorem 3.4] yield

[redacted]

Assume radix-2, precision-p arithmetic, with rounding unit $u = 2^{-p}$. Let $\widehat{x}$ be then computed 2^n-point IDCT of $X \in \mathbb{C}^{2^n}$, and let x be the exact value. Then

$$\|\widehat{x} - x\|_\infty = n\|X\|_\infty O(u).$$

Table: IDCT error bounds for $p = 53$ (double precision)

$N = 2^n$	1024	2048	4096	8192	16384	32768
$\|\widehat{x} - x\|_\infty / \|X\|_\infty$	7.97e-15	8.84e-15	9.72e-15	1.06e-14	1.15e-14	1.23e-14

Fast multipoint evaluation and subdivision algorithm

Algorithm: multipoint evaluation and subdivision

Illustration

$$P(x,y) = \sum \left(\sum a_{i,j} x^i \right) y^j = \sum p_j(x) y^j$$

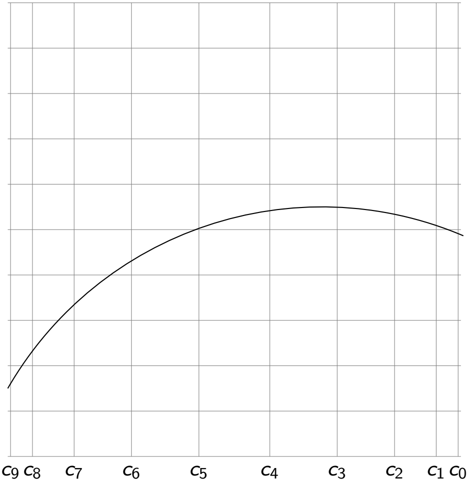
$$p_j(x) = \sum a_{i,j} x^i = \sum \alpha_{i,j} T_i(x)$$

$$(p_j(c_n))_{0 \le n \le N-1} = \frac{1}{2}(\alpha_{0,j}, \dots, \alpha_{0,j}) + \mathrm{IDCT}((\alpha_{k,j})_{0 \le k \le N-1})$$

Algorithm: multipoint evaluation and subdivision

Illustration

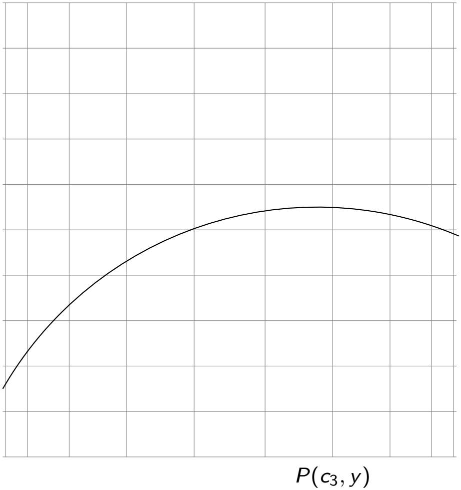
$P(c_n, y) = \sum p_j(c_n) y^j$



Algorithm: multipoint evaluation and subdivision

Illustration

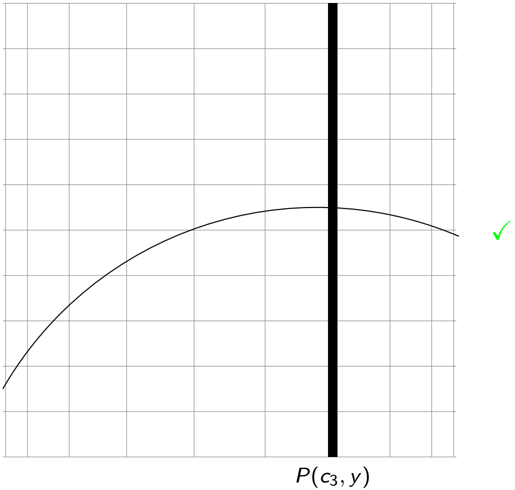
$P(c_3, y) = \sum p_j(c_3)y^j$



Algorithm: multipoint evaluation and subdivision

Illustration

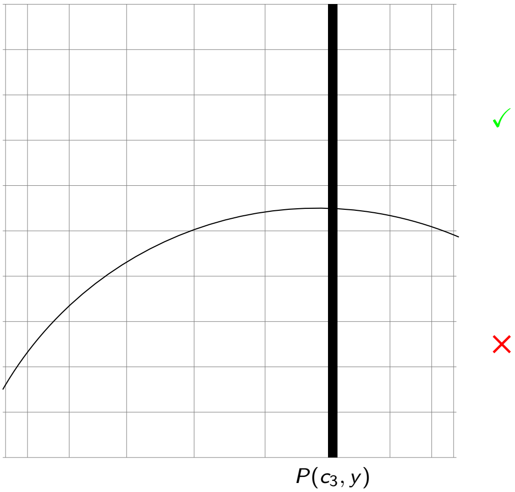
$P(c_3, y) = \sum p_j(c_3) y^j$



Algorithm: multipoint evaluation and subdivision

Illustration

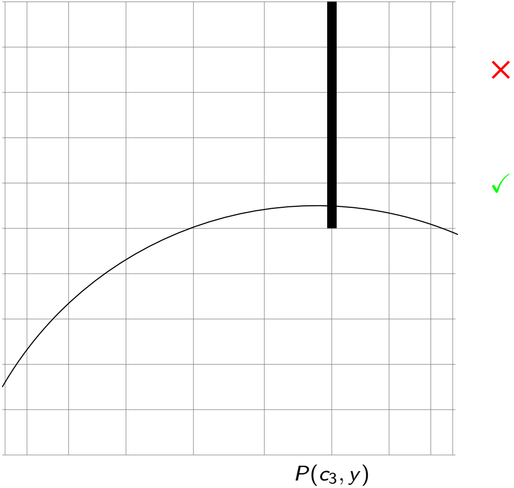
$P(c_3, y) = \sum p_j(c_3) y^j$



Algorithm: multipoint evaluation and subdivision

Illustration

$P(c_3, y) = \sum p_j(c_3) y^j$



Algorithm: multipoint evaluation and subdivision

Complexity

[redacted]

$O(dNT)$ with $1 \leq T \leq N$

T: the maximum number of nodes of the subdivision trees over all vertical fibers

With a finite number of branches in the window, we expect $T = O(\log_2(N))$

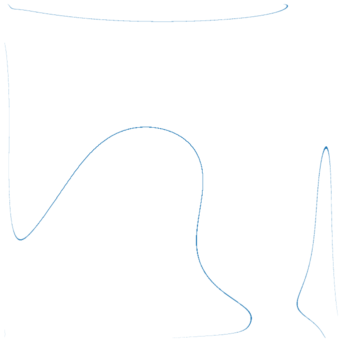
Experiments

Drawing for two families of polynomial

$\xi_{i,j} \in \mathcal{U}[-100, 100]$ i.i.d.

Kac polynomial

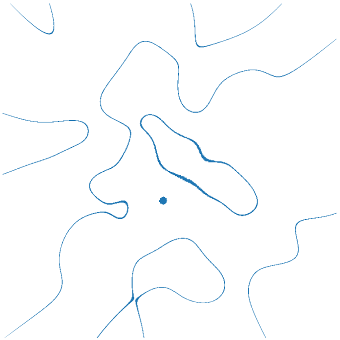
$$P(x,y) = \sum_{i+j=0}^{d} \xi_{i,j} x^i y^j$$



$d = 110$

Kostlan-Shub-Smale (KSS) polynomial

$$P(x,y) = \sum_{i+j=0}^{d} \sqrt{\frac{d!}{i!j!(d-i-j)!}} \xi_{i,j} x^i y^j$$



$d = 40$

Comparison to state-of-the-art software

- scikit $\rightarrow$ marching squares
- MATLAB $\rightarrow$ could not find the method used
- ImplicitEquations $\rightarrow$ quad-tree and interval arithmetic

- Isotop $\rightarrow$ CAD

Timing

Comparison for a polynomial

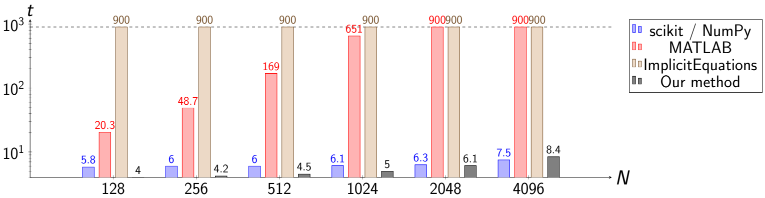


Figure: Computation times for a **Kac** polynomial of degree 40 (in seconds).

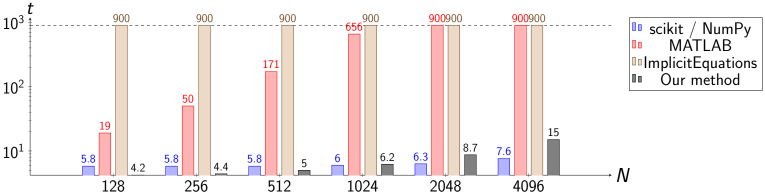
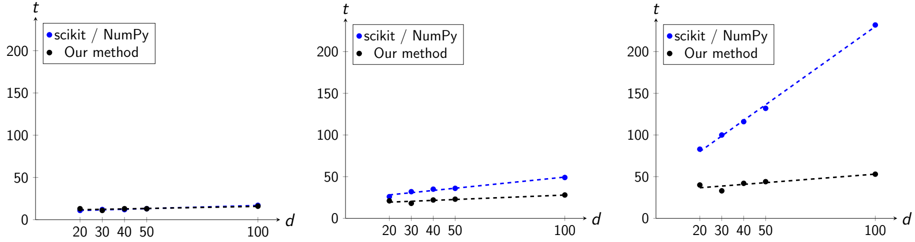


Figure: Computation times for a **KSS** polynomial of degree 40 (in seconds).

Timing

Marching squares and our method for high resolutions



$N = 8192$ $N = 16384$ $N = 32768$

Comparison of computation times for **Kac** polynomials (in seconds).

Marching cubes: $O(dN^2)$ Our method: $O(dNT)$

Timing

A CAD approach: Isotop

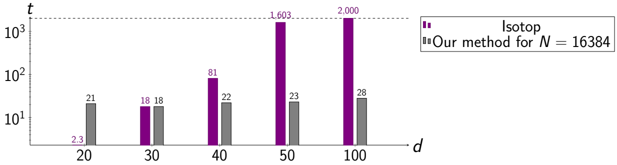


Figure: Computation times for a **Kac** polynomials (in seconds).

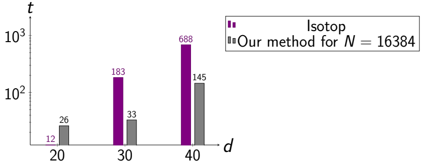


Figure: Computation times for a **KSS** polynomials (in seconds).