Computing Periods... *

Junhee Cho, Sewon Park, and Martin Ziegler

KAIST School of Computing, {junheecho,sewon,ziegler}@kaist.ac.kr

Abstract. A *period* is the difference between the volumes of two semi-algebraic sets. Recent research has located their worst-case complexity in low levels of the Grzegorczyk Hierarchy. The present work introduces, analyzes, and evaluates three rigorous algorithms for rigorously computing periods: a deterministic, a randomized, and a 'transcendental' one.

Keywords: Exact Real Computation, Reliable Numerics, Computational Algebraic Geometry, Randomized Algorithms

1 Introduction

A *period* is the absolutely convergent integral of a multivariate rational function with integer coefficients over Euclidean domains given by polynomial inequalities with integer coefficients [KZ01]:

$$\int_{\Delta} \frac{p(x_1,\ldots,x_d)}{q(x_1,\ldots,x_d)}\, dx_1 \cdots dx_d \ , \tag{1}$$

where $\Delta \subseteq \mathbb{R}^d$ is a Boolean combination of strict and non-strict polynomial inequalities $p_j(\vec{x}) > 0$ and $q_i(\vec{x}) \geq 0$ over, like p and q, integer coefficients: $p, q, p_1, \ldots, p_J, q_1, \ldots, q_I \in \mathbb{Z}[X_1, \cdots, X_d]$. Periods are receiving increasing interest in Algebraic Model Theory as they have finite descriptions (the polynomials' coefficients) and include all algebraic reals as well as some transcendentals:

$$\sqrt{2} \;=\; \int_{t:2t^2\leq 1} t\, dt, \quad \ln(x) \;=\; \int_1^x 1/t\, dt \;=\; \int_{t\leq x, s\cdot t\leq 1} 1\, ds\, dt, \quad \pi \;=\; \int_{x^2+y^2\leq 1} 1\, dx\, dy \tag{2}$$

Every period can be expressed as difference of two semi-algebraic volumes: For co-prime $p, q \in \mathbb{Z}[X_1, \ldots, X_d]$, Equation (1) translates to

$$\int_{\Delta_{p,q,+}} 1\, d\vec{x}\, dy \;-\; \int_{\Delta_{p,q,-}} 1\, d\vec{x}\, dy \;=\; \mathrm{vol}(\Delta_{p,q,+}) \;-\; \mathrm{vol}(\Delta_{p,q,-}) \ , \tag{3}$$

*Based on ideas presented at CCA 2017, this work was supported by the National Research Foundation of Korea (grant NRF-2017R1E1A1A03071032) and the International Research & Development Program of the Korean Ministry of Science and ICT (grant NRF-2016K1A3A7A03950702). We thank the anonymous referees for feedback!

where $\Delta_{p,q,+} := \{(\vec{x},y) : 0 \le y \cdot q(\vec{x}) \le p(\vec{x}) \wedge q(\vec{x}) > 0\} \cup \{(\vec{x},y) : 0 \ge y \cdot q(\vec{x}) \ge p(\vec{x}) \wedge q(\vec{x}) < 0\}$ and $\Delta_{p,q,-} := \{(\vec{x},y) : 0 \ge y \cdot q(\vec{x}) \ge p(\vec{x}) \wedge q(\vec{x}) > 0\} \cup \{(\vec{x},y) : 0 \le y \cdot q(\vec{x}) \le p(\vec{x}) \wedge q(\vec{x}) < 0\}$. Note that $\Delta_{p,q,+}, \Delta_{p,q,-} \subseteq \mathbb{R}^{d+1}$ may be unbounded even when Δ was bounded. However by means of Singularity Resolution one can w.l.o.g. restrict to compact domains [VS17, Theorem 1.1]. This shows that sum and (Cartesian) product of periods are again periods.

Many interesting open questions evolve around periods; for instance [KZ01, Problem 1] of whether there exists an algorithm that, given two representations of periods, decides whether they are equal or not? Or [KZ01, Problem 3] asking for an 'explicit' example of a real number that is not a period. Inner-mathematical candidates are $1/\pi$ and $e = \sum_n 1/n!$, but proving so seems infeasible with the current methods. The family of (semi-algebraic domains and thus also of) periods being countable, non-periods must be abundant.

In fact every period is computable in the sense of Recursive Analysis [Tur37, Wei00]; therefore any uncomputable real, such as the Halting Problem encoded in binary or random reals [BDC01] like Chaitin's Ω, cannot be periods. More precisely each period is of *lower elementary* complexity in Grzegorczyk's Hierarchy [Yos08, TZ10, SWG12]. Note that such improved upper complexity bounds give rise to more candidates of non-periods.

Problem 1. Characterize the computational complexity of periods!

Moreover efficient algorithms for computing (i.e. producing guaranteed high-precision approximations to) periods enable Experimental Mathematics [KZ01, Problem 2]; cmp. [Bai17].

We present three such algorithms: a deterministic one, a randomized (Las Vegas) one, and a transcendental one (to be clarified below). We prove them correct; describe their implementation in the convenient *Exact Real Computation* paradigm; estimate their cost in the (possibly unrealistic) unit cost model; and empirically analyze and compare their behaviour in terms of the output precision and the degree of the polynomial involved.

Subsection 1.1 recalls central notions, properties, and practice of real computation; Subsection 1.2 puts things in perspective to related work. Our algorithms are presented, and proven correct, in Section 2. In Section 3 we introduce our implementation, performance measurements, and their evaluation/interpretation. Section 4 expands on future work.

1.1 Real Computation

Regular floating-point arithmetic incurs rounding errors that accumulate over time and hamper reliable computations. Interval calculations keep track of the error bounds — which may blow up beyond use and due to overlap render comparisons meaningless. The present work peruses the `iRRAM` C++ library [Mül01, MZ14], providing real numbers as abstract data type with exact operations and partial comparison: A test "$y > 0$" freezes in case $y = 0$.

Indeed it is well-known from Recursive Analysis that equality of real (and not just algebraic) numbers is equivalent to the complement of the Halting Problem [Wei00, Exercise 4.2.9]. Here, computing $y \in \mathbb{R}$ means to produce dyadic approximations $a_n/2^n$, $a_n \in \mathbb{Z}$, to y up to absolute error $1/2^n$; similarly for real arguments x.

To write total programs in spite of comparison being partial, a *parallel disjunction* is provided: calling the non-deterministic or *multivalued* 'function' `choose`$(x_1 > 0, \ldots, x_k > 0)$ returns *some* integer j such that $x_j > 0$ holds, provided such j exists.

Subject to this modified semantics of tests, *Exact Real Computation* allows to conveniently process real arguments and intermediate results as entities, naïvely without precision considerations. On the other hand the output/return value of a function in `iRRAM` merely needs to be provided in approximation up to absolute error 2^p for any parameter $p \in \mathbb{Z}$ passed. The following algorithm demonstrates this paradigm with the *trisection* method for finding the (promised unique and simple) root of a given continuous function $f : [0;1] \to [-1;1]$ while avoiding the sign test "$0 < f(a) \cdot f(b')$" to fail in case b' already happens to be a root:

Algorithm `REAL` Trisection(`INTEGER` p, `REAL` $\to$ `REAL` f)

1: `REAL` $\ni a := 0$; `REAL` $\ni b := 1$
2: **while** `choose`$\big(\, b - a > 2^{p-1} \;,\; 2^p > b - a \,\big) == 1$ **do**
3: `REAL` $a' := \frac{2}{3}a + \frac{1}{3}b$; `REAL` $b' := \frac{1}{3}a + \frac{2}{3}b$;
4: **if** `choose`$\big(\, 0 > f(a) \cdot f(b') \;,\; 0 > f(a') \cdot f(b) \,\big) == 1$
5: **then** $b := b'$ **else** $a := a'$ **end if**
6: **end while**; **return** a

Remark 2. A caveat, in *Exact Real Computation* the bit — as opposed to unit — cost of each operation may well depend on the value (and internal precision) of the data being processed. For instance a sign test "$x > 0$" will take time between linear and quadratic in $n \approx \log_2(1/|x|)$: to obtain the dyadic approximation $a_n/2^n$ of x up to error 2^{-n} and verify it to be strictly larger than 2^{-n}. Similarly, a a parallel test `choose`$(x_1 > 0, \ldots, x_k > 0)$ will take time roughly $k \cdot \min_{j<k} \log(1/|x_j|)$.

1.2 Periods and their Computational Complexity

It has been shown that periods are of *lower elementary* complexity [TZ10, SWG12]. The present subsection recalls this and related notions with their connections to resource-bounded complexity.

In Grzegorczyk's Hierarchy, *Lower Elementary* means the smallest class of total multivariate functions $f : \mathbb{N}^d \to \mathbb{N} = \{0, 1, 2, \cdots\}$ containing the constants, projections, successor, modified difference $x \dot{-} y = \max\{x - y, 0\}$, and is closed under composition and bounded summation $f(\vec{x}, y) = \sum_{z=0}^{y} g(\vec{x}, z)$.

We write $\mathcal{M}^2 = \mathcal{E}^2$ for the smallest class of such f containing the constants, projections, successor, modified difference, binary multiplication, and closed under both composition and bounded search $\mu(f)(\vec{x}, y) = \min\{z \le y : f(\bar{z}, z) = 0\}$.

A real number r is *lower elementary* if there exist lower elementary integer functions $f, g, h : \mathbb{N} \to \mathbb{N}$ with $\left|r - \frac{f(N)-g(N)}{h(N)}\right| < 1/N$ for all $N > 0$; similarly for a real number in $\mathcal{M}^2$.

A real number r is computable in *time* $t(n)$ and *space* $s(n)$ if a Turing machine can, given $n \in \mathbb{N}$ and within these resource bounds, produce some $a_n \in \mathbb{Z}$ with $|r - a_n/2^n| \le 2^{-n}$ [Ko91].

Fact 3 *a) All functions from $\mathcal{M}^2$ are lower elementary; and the latter functions grow at most polynomially in the value of the arguments. In terms of the binary input length and with respect to bit-cost, lower elementary functions are computable using a linear amount of memory for intermediate calculations and output, that is, they belong to the complexity class $\mathsf{FSPACE}(n)$.*

b) $\mathsf{FSPACE}(n)$ is closed under bounded summation and therefore coincides with the class of lower elementary functions. The 0/1-valued functions (that is, decision problems) in $\mathcal{M}^2$ exhaust the class $\mathsf{SPACE}(n)$ [Rit63, §4]*; cmp.* [Kut87].

c) π and $e = \sum_n 1/n!$ and Liouville's transcendental number $L = \sum_n 10^{-n!}$ and the Euler-Mascheroni Constant $\gamma = \lim_n \left(-\ln(n) + \sum_{k=1}^{n} 1/k \right)$ are all lower elementary [Sko08, §3].

d) The set of lower elementary real numbers constitutes a real closed field: Binary sum and product and reciprocal of lower elementary real numbers, as well as any real root of a non-zero polynomial with lower elementary coefficients, are again lower elementary [SWG12, Theorem 2].

e) Arctan, natural logarithm and exponential as well as Γ and ζ function map lower elementary reals to lower elementary reals [TZ10, §9].

f) Natural logarithm maps periods to periods; $\zeta(s)$ is a period for every integer $s \ge 2$ [KZ01, §1.1].

g) Periods are lower elementary [TZ10, Corollary 6.4].

h) Given a Boolean expression $\varphi(x_1, \cdots, x_m)$ as well as the degrees and coefficients of the polynomials p_j defining its constituents S_{p_j}, deciding whether the semi-algebraic set $\varphi\big(S_{p_1}, \cdots, S_{p_m}\big)$ is non-empty/of given dimension [Koi99] *is complete for the complexity class $\mathsf{NP}^0_{\mathbb{R}} \supseteq \mathsf{NP}$.*

Item a) follows by structural induction. Together with b) it relates resource-oriented to Grzegorczyk's structural Complexity Theory. Note that the hardness Result h) does not seem to entail a lower bound on the problem of approximating a fixed volume; in fact many of the usual reductions among real algebraic decision problems [Mee06] fail under volume considerations. Common efficient and practical algorithms tailored for approximating L, e, γ, or the period π do so up to absolute error $1/N := 2^{-n}$ within time polynomial in the binary precision parameter $n = \log_2 N$ [Kan03]; whereas the best runtime bound known for $\mathsf{SPACE}(n)$ is only exponential [Pap94, Problem 7.4.7]. On the other hand exponential-time algorithms may well be practical [FG06, KF10].

2 Our Algorithms and their Analyses

In view of Equation (3) and Subsections 1.1+1.2, this section devises and analyzes algorithms that approximate, up to guaranteed absolute error 2^{-n}, the volume of the set of solutions $\vec{x}$ to a (given) disjunction of m conjunctions of monic polynomial inequalities, each of maximum degree $\leq k$ in d variables with integer coefficients between -2^ℓ and $+2^\ell$.

By appropriate integer scaling and shifting, it suffices to restrict to the unit cube $[0;1)^d$ and to strict inequalities; see Fact 5a) below. The common basic idea underlying all of our algorithms is to divide $[0;1]^d$ into sub-cubes

$$Q_{\vec{c},N} \;:=\; \left[\tfrac{\vec{c}}{N};\tfrac{\vec{c}+\vec{1}}{N}\right) \;=\; \prod_{i=1}^{d}\left[\tfrac{c_i}{N};\tfrac{c_i+1}{N}\right), \quad \vec{c}\in[N]^d \;, \tag{4}$$

where $[N] := \{0,\ldots,N-1\}$ and $\vec{1} := (1,\ldots,1)$; then determine the signs of the polynomials p_j in *some* point $\vec{x}_{\vec{c},N} \in Q_{\vec{c},N}$; and count those, where the constraints $p_j(\vec{x}_{\vec{c},N}) > 0$ are met, with uniform weight $\mathrm{vol}(Q_{\vec{c},N}) = N^{-d}$.

However (I) a polynomial's sign may vary within a $Q_{\vec{c},N}$, the above approach can incur an error. Moreover (II) $\vec{x}_{\vec{c},N}$ may happen to be (close to) a root of p_j; in which case determining the sign of $p_j(\vec{x}_{\vec{c},N})$ may take long in terms of bit-cost, or fail entirely. The sequel describes our approaches to still achieve totally correct algorithms: Subsection 2.1 takes care of (I), while Subsections 2.3, 2.2, and 2.4 describe three different approaches to avoid (II).

2.1 Recap on Real Algebraic Geometry

Regarding (I) in the case $d = 1$ a sign change can affect, namely occur in, at most $m \cdot k$ of the sub-intervals $Q_{c,N}$: because each of the m univariate polynomials of degree $\leq k$ can have at most k roots. So taking $N \geq m \cdot k \cdot 2^n$ guarantees the required error bound 2^{-n}. A multivariate polynomial on the other hand may have infinitely many roots — which however can form only a bounded number of connected components. This allows us to generalize the 1D analysis of (I) as follows:

Lemma 4. *a) In case $d = 2$ and for any fixed non-zero polynomial $p \in \mathbb{R}[X,Y]$ of maximum degree k, at most $1+(k-1)\cdot(k-2)/2+2(N+1)\cdot k$ of the $N\times N$ sub-squares $Q_{\vec{c},N}$ can contain roots of p.*

b) In case $d = 2$ and for an arbitrary finite family of non-zero polynomials p_j of maximum degree k, at most $1+(2k-1)\cdot(2k-2)/2+4(N+1)\cdot k$ of the $N\times N$ sub-squares $Q_{\vec{c},N}$ can contain simultaneous roots of all the p_j.

c) For $d \geq 3$ and any fixed non-zero polynomial $p \in \mathbb{R}[X_1,\ldots,X_d]$ of maximum degree k, at most $k^d + k^{d-1}\cdot d\cdot(N+1)$ of the N^d sub-(hyper)cubes $Q_{\vec{c},N}$ can contain roots of p.

d) For $d \geq 3$ and an arbitrary finite family of non-zero polynomials of maximum degree k, at most $k\cdot(2k-1)^d + k\cdot(2k-1)^{d-2}\cdot d\cdot(N+1)$ of the N^d sub-(hyper)cubes $Q_{\vec{c},N}$ can contain simultaneous roots of all of them.

e) It holds $k^d + k^{d-1}\cdot d\cdot(N+1) \leq N^d\cdot 2^{-n}$ for all $N \geq 3k\cdot 2^{n/(d-1)}$ and $k \geq d \geq 3$.

Note that the number m of polynomials in a conjunction $\bigcap_{j=1}^{m} p_j(\vec{x}) = 0$ barely affects the above bounds, since in the real setting it is equivalent to the single equation of double degree $\sum_{j=1}^{m} p_j^2(\vec{x}) = 0$.

In order to guarantee absolute error bound 2^{-n}, all our algorithms described in the following subsections will in case $d = 2$, rather than apply the concise but asymptotic Lemma 4e), build on Lemma 4a) and use binary search to find the least $N \in \mathbb{N}$ satisfying $N^2 \geq 2^n \cdot \big(1 + (k-1) \cdot (k-2)/2 + 2(N+1) \cdot k\big)$.

Proof. a) By Fact 5c) below, the roots of p can form at most $c \leq 1 + (k-1) \cdot (k-2)/2$ connected components in $\mathbb{R}^2$. Of course such a component may extend through more than one of the sub-squares $Q_{\vec{c},N}$; however in order to do so, it must cross one of the $N+1$ horizontal lines or one of the $N+1$ vertical lines forming the sub-division of $[0;1]^2$. More precisely for component C to extend to $M_C \in \mathbb{N}$ of the N^2 sub-squares, it must intersect at least $M_C - 1$ of the $2(N+1)^2$ segments of the $2(N+1)$ aforementioned lines; and for all c components to extend to a total of M of the N^2 sub-squares, they have to intersect these lines in at least $M - c$ points — distinct points, since connected components do not meet. However p restricted to any of the $2(N+1)$ lines boils down to a univariate polynomial (either in X or in Y) of degree at most k, and hence can have at most k roots on each such line: requiring $M - c \leq 2k \cdot (N+1)$.

b) Joint roots of the real p_j of maximum degree k are precisely the roots of the single polynomial $\sum_j p_j^2$ of maximum degree $2k$.

c) Similarly to the proof of (a), the roots of p can form at most $c_d \leq k^d$ connected components according to Fact 5d). For any such component C to extend to $M_C \in \mathbb{N}$ of the N^d sub-(hyper)cubes, it must intersect at least $M_C - 1$ of the $d \cdot (N+1)^d$ facets of the overall subdivision induced by the $d \cdot (N+1)$ hyperplanes; and for all c_d components to extend to a total of M of the N^d sub-cubes, they have to intersect these hyperplanes in at least $M - c_d$ different components! However p restricted to any of the $d \cdot (N+1)$ hyperplanes boils down to a $(d-1)$-variate polynomial of maximum degree at most k, whose roots can form at most $c_{d-1} \leq k^{d-1}$ connected components according to Fact 5d): $M - c_d \leq c_{d-1} \cdot d \cdot (N+1)$.

d) Similarly.

e) Apply ineqality $|x|^d + |y|^d \leq (|x| + |y|)^d$ to $x^d := 2^n \cdot (k^d + d \cdot k^{d-1}) \leq 2k^d \cdot 2^{n \cdot d/(d-1)}$ and $y^d := N \cdot d \cdot k^{d-1} \cdot 2^n$, taking into account $\sqrt[d]{2} + \sqrt[d-1]{d} \leq 3$ for all $d \geq 3$. □

Fact 5 *a) The set $\{\vec{x} : p(\vec{x}) = 0\}$ of roots of a non-zero polynomial p has measure zero.*

b) Let $S_1, \ldots, S_d \subseteq \mathbb{R}$ be arbitrary subsets of cardinality $|S_i| > k$ and $p \in \mathbb{R}[X_1, \ldots X_d]$ non-zero of maximal degree $\leq k$. Then there exists some $\vec{x} \in \prod_{i=1}^{d} S_i$ with $p(\vec{x}) \neq 0$.

c) Let $p \in \mathbb{R}[X, Y]$ denote a bivariate polynomial of maximum degree k. Then the number of connected components of $\{(x, y) : x, y \in \mathbb{R}, p(x, y) = 0\}$ is at most $1 + (k-1) \cdot (k-2)/2$.

d) If $\Delta \subseteq \mathbb{R}^d$ is the zero set of one polynomial of maximum degree k, then it has at most k^d connected components; if it is the conjunction of (any number of) such sets, then it has at most $k \cdot (2k-1)^d$ connected components.

e) For $p_1, \ldots, p_d$ pairwise distinct primes, $\vec{x} := \left(e^{\sqrt{2}}, e^{\sqrt{3}}, \ldots, e^{\sqrt{p_d}}\right)$ is algebraically independent: $q(\vec{x}) \neq 0$ for every non-zero d-variate polynomial q with integer coefficients; and more generally $q\big(A \cdot \vec{x} + \vec{b}\big) \neq 0$ for any vector $\vec{b} \in \mathbb{A}^d$ with algebraic coefficients and invertible $d \times d$-matrix $A \in \mathrm{GL}_d(\mathbb{A})$.

Claim b) strengthens (a) and proceeds by induction on d: For any fixed $(x_1, \ldots x_{d-1}) \in \prod_{i=1}^{d-1} S_i$, $p(x_1, \ldots x_{d-1}, X_d)$ is a univariate polynomial of degree $\leq k$. Claim c) is *Harnack's Curve Theorem* [PP16, Theorem 48.1], d) its generalization due to Milnor and Thom [HRR90, Theorem 9]; e) is the *Lindemann-Weierstrass Theorem*, applied to linear independence of prime square roots over rationals.

2.2 A Real Randomized Algorithm

In order to avoid (II) accidentally hitting a root $\vec{x}$ of some constraint polynomial p when trying to determine its sign in the sub-cube $Q_{\vec{c},N} \ni \vec{x}$, randomization provides an arguably easiest solution: By Fact 5 the probability for this to occur is zero. So picking a random $\vec{x} \in Q_{\vec{c},N}$ gives rise to a Las Vegas-type algorithm: always correct, but with running time proportional to $\log\big(1/p(\vec{x})\big)$; recall Remark 2.

Randomization has become ubiquituous in Theoretical Computer Science — regarding discrete problems [MU05]: In the real setting it has only (yet thoroughly) been considered with respect to computability; cmp.[BGH15].

Here we have designed the first truly real random number generator in *Exact Real Computation*. Each call produces some $r \in [0;1]$ independently with respect to the uniform distribution: Repeating d times, shifting by $\vec{c}$ and scaling with $1/N$ then yields the sought $\vec{x} \in Q_{\vec{c},N}$. Internally our real generator in turn builds on a generic source of independent fair coin flips; equivalently: independent uniformly distributed integers in the range from 0 to $2^n - 1$, for any given $n \in \mathbb{N}$. (The uniform distribution on $[0;1]$ is then easily converted to other popular continuous ones such as, say, Gaussian on $\mathbb{R}$.)

2.3 A Deterministic Algorithm

With the dimension d fixed, we can avoid the problem of derandomizing *Polynomial Identity Testing* and still get an efficient deterministic algorithm in *Exact Real Computation* for finding the sign of any given non-zero $p \in \mathbb{R}[X_1, \ldots, X_d]$ of maximal degree $\leq k$ in *some* non-root $\vec{x} \in Q_{\vec{c},N}$: Based on Fact 5b), evaluate p's sign on the $(k+1)^d$ points of a $(k+1) \times \cdots \times (k+1)$ grid in $Q_{\vec{c},N}$ *in parallel*, recall Remark 2.

2.4 A Transcendental Algorithm

By Fact 5e), algebraically independent arguments $\vec{x} = (x_1, \ldots, x_d)$ avoid problem (II) and can be computed efficiently: deterministically. (Any tuple of independent *random* reals of course is algebraically independent with certainty.) This algorithm thus takes such a $\vec{x}$ and scales and shifts it by dyadic rationals to lie in $Q_{\vec{c},N}$.

3 Implementation and Evaluation

Evaluating a given d-variate polynomial of maximal degree $\leq k$ takes $\mathcal{O}(k^d)$ arithmetic operations. Combined with Lemma 4e), we conclude that the number of operations performed by the randomized and by the transcendental algorithm (Subsections 2.2+2.4) to achieve guaranteed output absolute error 2^{-n} is $\mathcal{O}\big(k^{d+1} \cdot 2^{n/(d-1)}\big)$; the deterministic algorithm (Subsections 2.3) incurs an additional factor $\mathcal{O}(k^d)$.

However this analysis only counts the number of operations, that is, referring to an algebraic or unit-cost measure — as opposed to the more realistic bit-cost measure taking into account aspects of internal or working precision.

The latter seem hard to estimate, though, since they depend not only on the output precision n and the polynomial degree k, but also on (the coefficient vector of) the polynomial constraints p under consideration: which in the worst-case may give rise to unbounded running times. For a more realistic assessment we have thus implemented, empirically evaluated and compared the practical performance of the above three algorithms on the following kinds of benchmark polynomials:

The bivariate polynomials X^2+Y^2-1 and $(X-2)^2+(X\cdot Y-1)^2$ representing[†] the transcendental periods π and $\ln(2)$; recall Equation 2); and for $d = 2$ and $d = 3$ the multivariate scaled *Wilkinson-type* Polynomials I $p_{n,d} := \prod_{i=1}^{d} \prod_{j=1}^{k} \left(X_i - \frac{j}{k}\right)$ deliberately placing roots at points on a grid that the deterministic algorithm will try to determine their signs in and *Wilkinson-type* Polynomials II $p_{n,d} := \prod_{j=1}^{k} \left(\prod_{i=1}^{d} X_i - \frac{j}{k} \right)$.

3.1 Computing Environment

The above three algorithms were implemented without multithreading in *Exact Real Computation* based on the `iRRAM` C++ library [Mül01, MZ14] commit `487a123`. Their source codes and the experiment results are available for download from url `https://github.com/junheecho/period` and `https://github.com/junheecho/iRRAM`. We have executed and timed them on a computer with Intel® Core™ i7-6950X processor with 10 cores, 20 threads running at 3.00 GHz and 64 GB of RAM. Less than 20 processes ran at the same time so that they do not impede each other. The source code is compiled with g++ 5.4.0 on Ubuntu 16.04.3 LTS. We focus on CPU time; memory was never a problem.

[†] Of course the specific periods π and $\ln(2)$ admit other, more efficient algorithms...

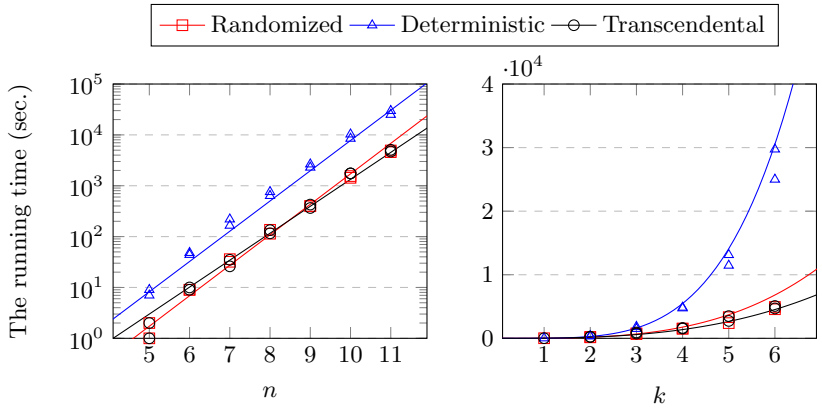
3.2 Performance Results

Input	Algorithm	Regression	R^2 score
π	Randomized	$\exp(1.31n - 9.51)$	0.99
	Deterministic	$\exp(1.14n - 7.69)$	0.88
	Transcendental	$\exp(1.34n - 9.73)$	1.00
(the ratio over the randomized)	Deterministic	$\exp(-3 \cdot 10^{-5}n + 0.03)$	0.00
	Transcendental	$\exp(-0.07n + 0.88)$	-0.66
ln(2)	Randomized	$\exp(1.38n - 11.74)$	1.00
	Deterministic	$\exp(1.30n - 10.65)$	1.00
	Transcendental	$\exp(1.31n - 10.75)$	1.00
(the ratio over the randomized)	Deterministic	$\exp(-0.09n + 1.09)$	0.62
	Transcendental	$\exp(0.07n - 0.57)$	0.38
Wilkinson-type Polynomials I	Randomized	$\exp(1.38n - 12.32) \cdot k^{3.33}$	0.89
		$\exp(1.01k + 1.37n - 11.95)$	0.49
	Deterministic	$\exp(1.37n - 12.34) \cdot k^{4.24}$	0.98
		$\exp(1.30k + 1.37n - 12.10)$	0.51
	Transcendental	$\exp(1.22n - 10.27) \cdot k^{2.94}$	0.95
		$\exp(0.90k + 1.21n - 10.01)$	0.86
(the ratio over the randomized)	Deterministic	$\exp(0.05n - 0.54) \cdot k^{0.94}$	0.63
		$\exp(0.31k + 0.05n - 0.60)$	0.73
	Transcendental	$\exp(-0.01n + 0.41) \cdot k^{-0.17}$	-0.05
		$\exp(-0.05k - 0.01n + 0.42)$	-0.06
Wilkinson-type Polynomials II	Randomized	$\exp(1.25n - 10.39) \cdot k^{2.80}$	0.98
		$\exp(0.73k + 1.24n - 9.82)$	0.76
	Deterministic	$\exp(1.27n - 10.47) \cdot k^{3.55}$	0.99
		$\exp(0.95k + 1.28n - 10.01)$	0.88
	Transcendental	$\exp(1.26n - 10.19) \cdot k^{2.64}$	0.99
		$\exp(0.66k + 1.24n - 9.36)$	0.92
(the ratio over the randomized)	Deterministic	$\exp(0.08n - 1.03) \cdot k^{1.03}$	0.73
		$\exp(0.28k + 0.08n - 0.96)$	0.85
	Transcendental	$\exp(0.02n + 0.12) \cdot k^{-0.15}$	0.11
		$\exp(-0.04k + 0.02n + 0.07)$	0.12

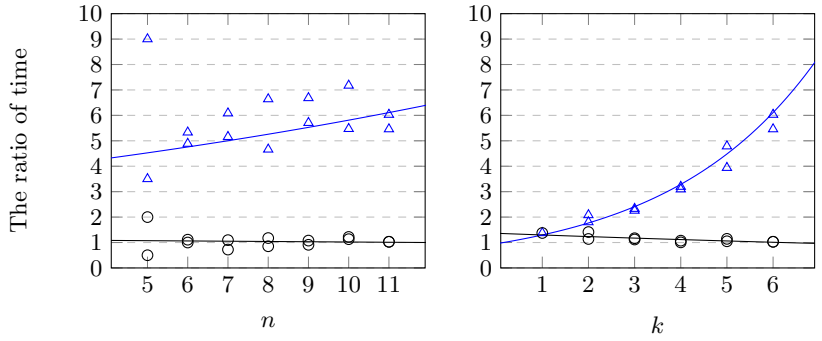
Table 1: Parameter Regression of CPU time

We have measured, for each of the three above algorithms and each of the above benchmark polynomials as input, the CPU time in dependence on n; for the Wilkinson-type Polynomials also on k. We have then fitted the results to the model/ansatz $\exp(n \cdot \beta - \alpha)$ — for the Wilkinson-type Polynomials to $\exp(n \cdot \beta - \alpha) \cdot k^{\gamma}$ — and plotted both. We have also fitted the result for the Wilkinson-type Polynomials to $\exp(k \cdot \gamma + n \cdot \beta - \alpha)$ but the R^2 scores show the aforementioned model is more suitable. We have also plotted and fitted the *ratios* of the algorithms' respective performances to the aforementioned models: See the following figures.

3.3 Interpretation



(a) The running time and the fit, $k = 6$ (left) and $n = 11$ (right)



(b) The ratio over the randomized, $k = 6$ (left) and $n = 11$ (right)

Fig. 1: CPU time (sec.) of computing periods via Wilkinson-type Polynomials I

Our measurements confirm the predicted running times polynomial in k and exponential in n. They furthermore exhibit an exponential advantage of the randomized and the transcendental algorithm over the deterministic one. The performances of the randomized and the transcendental algorithms are identical. The difference of performance is little with respect to n, but significant with respect to k; thus, the performances of all algorithms are identical when computing π and $\ln(2)$ because n is the only parameter.

Following the *Balls-into-Bins* paradigm [BCSV06], we have tried a synthesis of the randomized and the deterministic algorithm (Subsections 2.2+2.3) that evaluates the polynomial's sign at *two* random points in parallel — however with no benefit in performance.

4 Conclusion and Perspectives

We have present three algorithms rigorously computing periods: a deterministic one, that evaluates the constraint polynomials at sufficiently many dyadic points simultaneously such as to guarantee at least one missing its roots; a randomized (and arguably first rigorous real) one, that misses roots almost surely; and one evaluating at an appropriate algebraically independent argument that by definition cannot constitute a root.

Although all three take time exponential in the output precision n, they exhibit significant differences in practical performance. Perhaps surprisingly, evaluating at two random points in parallel (and thus automatically choosing the 'better' one) turned out to be slower, not faster, than a single one.

For now we have focused on the case of two (and three) variables. Future work will extend in that, and the following directions:

a) Picking up on the first paragraph of Section 3, we will try to identify reasonable parameters of the polynomials $p \in \mathbb{Z}[X_1, \ldots, X_d]$ under consideration, in addition to their maximal degree bound k, to devise a refined rigorous parameterized bit-cost analysis.
b) The question remains open as of whether every fixed period can be computed (i.e. approximated up to absolute error 2^{-n}) in time polynomial in n.
c) In the spirit of *Experimental Mathematics*, we plan to algorithmically search for new (candidate, linear or algebraic) relations among periods.

References

[Bai17] David H. Bailey. Jonathan borwein: Experimental mathematician. *Experimental Mathematics*, 26(2):125–129, 2017.

[BCSV06] Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. Balanced allocations: The heavily loaded case. *SIAM J. Comput.*, 35(6):1350–1385, 2006.

[BDC01] Verónica Becher, Sergio Daicz, and Gregory Chaitin. A highly random number. In *Combinatorics, computability and logic*, Springer Ser. Discrete Math. Theor. Comput. Sci., pages 55–68, London, 2001. Springer.

[BGH15] Vasco Brattka, Guido Gherardi, and Rupert Hölzl. Las Vegas computability and algorithmic randomness. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015)*, volume 30 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 130–142, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[HRR90] Joos Heintz, Tomás Recio, and Marie-Françoise Roy. Algorithms in real algebraic geometry and applications to computational geometry. In Jacob E. Goodman, Richard Pollack, and William Steiger, editors, *Discrete and Computational Geometry: Papers from the DIMACS Special Year*, volume 6 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 137–164. DIMACS/AMS, 1990.

[Kan03] Yasumasa Kanada. 計算機による円周率計算（特集 円周率 π）. *J. Mathematical Culture*, 1(1):72–83, 2003.

[KF10] Dieter Kratsch and Fedor V. Fomin. *Exact Exponential Algorithms*. EATCS. Springer, 2010.

[Ko91] Ker-I Ko. *Complexity Theory of Real Functions*. Progress in Theoretical Computer Science. Birkhäuser, Boston, 1991.

[Koi99] Pascal Koiran. The real dimension problem is NP_R-complete. *Journal of Complexity*, 15(2):227–238, 1999.

[Kut87] Miroslaw Kutylowski. Small Grzegorczyk classes. *Journal of the London Mathematical Society*, 36(2):193–210, 1987.

[KZ01] Maxim Kontsevich and Don Zagier. Periods. In Björn Engquist and Wilfried Schmid, editors, *Mathematics unlimited — 2001 and beyond*, pages 771–808. Springer, 2001.

[Mee06] Klaus Meer. Optimization and approximation problems related to polynomial system solving. In *Proc. 2nd Conference on Computability in Europe (CiE'06)*, pages 360–367, 2006.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[Mül01] Norbert Th. Müller. The iRRAM: Exact arithmetic in C++. In Jens Blanck, Vasco Brattka, and Peter Hertling, editors, *Computability and Complexity in Analysis*, volume 2064 of *Lecture Notes in Computer Science*, pages 222–252, Berlin, 2001. Springer. 4th International Workshop, CCA 2000, Swansea, UK, September 2000.

[MZ14] Norbert Th. Müller and Martin Ziegler. From calculus to algorithms without errors. In Hoon Hong and Chee Yap, editors, *Proc. 4th International Congress on Mathematical Software (ICMS)*, volume 8592 of *Lecture Notes in Computer Science*, pages 718–724. Springer, 2014.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[PP16] Patric Popescu-Pampu. *What is the Genus?*, volume 2162 of *Lecture Notes in Mathematics*. Springer, 2016.

[Rit63] Robert W. Ritchie. Classes of predictably computable functions. *Trans. Amer. Math. Soc.*, 106(1):139–173, 1963.

[Sko08] Dimiter Skordev. On the subrecursive computability of several famous constants. *Journal of Universal Computer Science*, 14(6):861–875, 2008.

[SWG12] Dimiter Skordev, Andreas Weiermann, and Ivan Georgiev. M^2-computable real numbers. *J. Logic Comput.*, 22(4):899–925, 2012.

[Tur37] Alan M. Turing. On computable numbers, with an application to the "Entscheidungsproblem". *Proceedings of the London Mathematical Society*, 42(2):230–265, 1937.

[TZ10] Katrin Tent and Martin Ziegler. Computable functions of reals. *Münster J. Math.*, 3:43–65, 2010.

[VS17] Juan Viu-Sos. A semi-canonical reduction for periods of kontsevich-zagier. *arXiv*, 1509.01097, 2017.

[Wei00] Klaus Weihrauch. *Computable Analysis*. Springer, Berlin, 2000.

[Yos08] Masahiko Yoshinaka. Periods and elementary real numbers. *arXiv*, 0805.0349, 2008.