On the Error of Random Sampling: Uniformly Distributed Random Points on Parametric Curves
Apostolos Chalkis, Christina Katsamaki, Josué Tonelli-Cueto

To cite this version:
Apostolos Chalkis, Christina Katsamaki, Josué Tonelli-Cueto. On the Error of Random Sampling: Uniformly Distributed Random Points on Parametric Curves. ISSAC ’22 - International Symposium on Symbolic and Algebraic Computation, Jul 2022, Villeneuve-d’Ascq, France. 10.1145/3476446.3536190. hal-03601563

HAL Id: hal-03601563
https://inria.hal.science/hal-03601563
Submitted on 25 Jul 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
On the Error of Random Sampling: Uniformly Distributed Random Points on Parametric Curves

Apostolos Chalkis
National & Kapodistrian Univ. Athens
GeomScale org
Athens, Greece
achalkis@di.uoa.gr

Christina Katsamaki
Inria Paris & IMJ-PRG
Sorbonne Université
Paris, France
christina.katsamaki@inria.fr

Josué Tonelli-Cueto
Inria Paris & IMJ-PRG
Sorbonne Université
Paris, France
josue.tonelli.cueto@bizkaia.eu

ABSTRACT
Given a parametric polynomial curve \( \gamma : \gamma : [a, b] \rightarrow \mathbb{R}^n \) in such a way that it is distributed uniformly with respect to the arc-length? Unfortunately, we cannot sample exactly such a point—even assuming we can perform exact arithmetic operations. So we end up with the following question: how does the method we choose affect the quality of the approximate sample we obtain? In practice, there are many answers. However, in theory, there are still gaps in our understanding. In this paper, we address this question from the point of view of complexity theory, providing bounds in terms of the size of the desired error.

KEYWORDS
parametric curve, sampling, sampling error, Chebyshev, approximation

1 INTRODUCTION
Given a parametric polynomial curve \( \gamma : \gamma : [a, b] \rightarrow \mathbb{R}^n \), we are interested in generating a random point \( x \in \gamma([a, b]) \) that is uniformly distributed with respect to the arc-length. To do this, we only need to sample a random variable \( l \in [a, b] \) with density proportional to the speed of the curve \( \|\gamma'(l)\|_2 \). Even if we perform exact arithmetic operations using real numbers with infinite precision, this problem does not admit an exact solution—the integral \( \int_{[a, b]} \|\gamma'(l)\|_2 \) cannot be expressed in terms of elementary functions. The goal of this paper is to estimate how much the generated random sample differs from the one that we want and how does the desired error affect the complexity.

1.1 Random Samples: Why do we care?
At this point, to avoid possible confusion, we want to clarify that we are studying random sampling on a parametric curve. Unfortunately, in the literature, the term 'sample' carries two meanings. In the context of computational statistics, this refers to generating a random point \( x \in \gamma([a, b]) \) in such a way that it is distributed uniformly with respect to the arc-length. To do this, we only need to sample a random variable \( l \in [a, b] \) with density proportional to the speed of the curve \( \|\gamma'(l)\|_2 \). Even if we perform exact arithmetic operations using real numbers with infinite precision, this problem does not admit an exact solution—the integral \( \int_{[a, b]} \|\gamma'(l)\|_2 \) cannot be expressed in terms of elementary functions. The goal of this paper is to estimate how much the generated random sample differs from the one that we want and how does the desired error affect the complexity.

1.2 Errors in Random Sampling: State of the art
Sampling from a density is an important and well-studied problem in computational statistics. There are many methods to sample according to a certain distribution: Acceptance-Rejection (AR) method [15], Adaptive Rejection Sampling [10], Slice Sampling [20], etc. Currently, the state-of-the-art samplers are the so-called Markov Chain Monte Carlo (MCMC) algorithms. They have plenty of success stories, such as Hamiltonian Monte Carlo [21], Hit-and-Run [26] and Metropolis-Hastings [4].

However, most of the aforementioned methods apply only for log-concave distributions [3, 5, 14, 16, 17]. The algorithm in [18] can be used to sample from (multivariate) non-convex density functions, but it does not handle the case where the density is restricted to an interval (or a bounded set in general). Moreover, for the univariate setting, the existing MCMC algorithms [12] either do not provide any error guarantees or could lead to arbitrarily high run-times. In particular, this means that the error-control for generating random samples on a parametric curve is an open problem.

1.3 Analyzed Method and Contributions
We analyze the method proposed by Olver and Townsed [23]. This method is not limited to random sampling on parametric curves, but it holds for general density functions. They demonstrate empirically its efficiency, however, no theoretical analysis is given. We aim to do this in the particular case of sampling random points on parametric polynomial curves, uniformly with respect to the arc-length.

The underlying idea of Olver and Townsed [23] is to use Chebyshev approximations to make inverse transform sampling tractable at the cost of an error in the produced random sample. The cumulative distribution function might not be expressible in terms of elementary functions—and this happens in the case of interest.

We provide the first error analysis for this method in the case where we are sampling random points on a parametric polynomial curve. For this error analysis, we work in the BSS model (see §2.2) for convenience. This model is suitable for developing a complexity theory over the real numbers, since a BSS machine is like a Turing machine, but it can operate with real numbers and it performs arithmetic operations and comparisons at unit cost. The main result of this paper is the following one:
we feel that these methods might not generalize easily to higher dimensions, that we plan to deal with in the future. Nevertheless, we leave for future work a careful study of these methods.

For these problems, there is an extensive literature [6–8, 11, 24, 25, 30] together with an extended experimental analysis, which confirms the theoretical results.

Figure 1: A sample of 300 random points from the curve \( \gamma : [-1, 1] \to \mathbb{R}^2 \) give by \( \gamma(t) = (3t^3 - 2t, 2t^3) \) generated by our algorithm CurveSampler. For more details on this example see 6.1.

Theorem 1.1. Let \( \gamma : I \to \mathbb{R}^n \) be a polynomial parameterized curve of degree \( d \). The algorithm CurveSampler samples points from \( \gamma \) uniformly with respect to the arc-length by performing

\[
O\left( t^3 (1 + \log \mathcal{C}(\gamma))^3 \mathcal{C}(\gamma)^3 \right)
\]

arithmetic operations and with error \( 2^{-\ell} \) with respect to the total variation distance, where \( \mathcal{C}(\gamma) \), the condition number for sampling \( \gamma \), is given Definition 5.8.

Our analysis is generic enough to be applicable to more general densities, but we focus on parametric curves since this is the case of interest in our research program. This paper is the first one studying reductions of randomness sources in the BSS model of computation. Even more, we provide an open-source Matlab implementation (see §6) together with an extended experimental analysis, which confirms the theoretical results.

1.4 Related Problems

Although we are analyzing the method of [23], we note that our problem—generating random points in parametric polynomial curves— is related to a lot of problems in the literature. On the one hand, this problem is related to obtaining arc-length parametrizations and generating deterministic uniform samples—points that are equidistributed with respect the arc-length—of parametric curves. For these problems, there is an extensive literature [6–8, 11, 24, 25, 30] and an analysis of these methods might be possible following our strategy—we only have to control the following \( L^1 \)-norm:

\[
\| \|\gamma'(\theta(s))\|_2 \theta'(s) - \lambda_{[0,L]}(s) \|_1 \|,
\]

where \( \theta : [0, L] \to I \) is a reparametrization of \( \gamma : I \to \mathbb{R}^n \). However, we feel that these methods might not generalize easily to higher dimensions, that we plan to deal with in the future. Nevertheless, we leave for future work a careful study of these methods.

2 HOW GOOD IS A SAMPLING METHOD?

Given a continuous random variable \( x \in \mathbb{R} \) (target random variable), we want to construct a sampler—an algorithm—whose output is identically distributed to \( x \). If the distribution of \( x \) is simple enough, the latter can be easily done. However, in general, we cannot sample \( x \) exactly and we can only obtain a random variable \( \tilde{x} \in \mathbb{R} \) (sampled random variable) which behaves approximately like \( x \). Hence the following question arises: how well does the sampled random variable \( \tilde{x} \) approximate the target random variable \( x \)?

2.1 Total Variation Distance

The total variation distance measures how much the probabilities of two arbitrary events differ; the smaller the total variation distance is, the harder it is to distinguish the sampled random variable from the target random variable.

Definition 2.1. [29] Let \( x, \tilde{x} \in \mathbb{R} \) be random variables. The total variation distance (TV distance) of \( x \) and \( \tilde{x} \) is defined as

\[
\text{dist}_{TV}(x, \tilde{x}) := \sup_{B \in \mathcal{B}} |\mathbb{P}(x \in B) - \mathbb{P}(\tilde{x} \in B)|,
\]

where \( \mathcal{B} \) is the set of Borel subsets of \( \mathbb{R} \).

2.2 What is an efficient sampler?

For a sampler to be efficient, we want it to run in time that is polynomial in the error. We use the Blum-Shub-Smale (BSS) model of computation [1] to avoid problems arising from approximating continuous random variables with discrete ones. In the BSS model, real numbers can be stored exactly as a single unit during computations, and operations with real numbers are done at unit cost. We call BSS program, a program, i.e., a finite list of commands, that can be implemented in a BSS machine.

We introduce the notion of an efficient sampler, which will be useful in measuring the performance of our sampling method.

Definition 2.2. Given a random variable \( x \in \mathbb{R} \), an efficient sampler for \( x \) is a pair of BSS programs \( S : \mathbb{N} \times \mathbb{R}^k \times [0,1]^I \to \mathbb{R} \) and \( P : \mathbb{N} \to \mathbb{R}^k \) such that: S1) on input \((l, x, u)\), the run-time of \( S \) is at least \( 2^{-l} \) and P2) \( S(l, x, u) \to x \).
most poly(t), S2) on input t, the run-time of P is at most exp(t), and S3) if u ∈ [0, 1] is uniformly distributed, then x_f := S(t, P(t), u) is a random variable such that
\[ \text{dist}_{TV}(x_f, x) \leq 2^{-f}. \]

The program P in Def. 2.2 represents the preprocessing that the sampler requires in order to produce the correct result and u the source of randomness. The output of P is used together with the source of randomness u as the inputs of the program S, which produces the sample.

**Remark 2.3 (On-line and off-line computations).** In a more classical conception of an efficient sampler, the program P would be missing. However, we have to take into account that samplers are not intended to be a run-once program, but they are intended to run many times. Because of this, it is reasonable to allow off-line computations—precomputations—even if these are expensive, as long as we don’t have to repeat them. In this way, our definition gives this possibility.

### 2.3 And if we have finite precision?

If we have finite precision, then the sampled random variable ̃x is discrete, and if this is the case, since the target random variable x is continuous, then
\[ \text{dist}_{TV}(x, ̃x) = 1. \]

Hence, the TV distance does not allow us to evaluate how a discrete sampler approximates a continuous random variable directly.

A way around this problem is to turn the discrete approximation ̃x into a continuous random variable by adding random noise to the values ̃x_1, ..., ̃x_a that ̃x. To do this, we sample
\[ x_i + η_i, \]
with probability P( ̃x = ̃x_i), where η_1, ..., η_a some set of continuous random variables that we can sample. The resulting random variable will be continuous with density
\[ \sum_{i=1}^{a} P( ̃x = ̃x_i) δ_{x_i}(t - ̃x_i). \]

Hence, if the η_i are simple enough, we have just specified a way of turning our discrete sample random variable into a continuous one for which we can evaluate the quality using the TV distance.

Unfortunately, analyzing in detail the precision needed goes beyond the scope of this paper. However, we note that all the methods produced in this paper—see the next two sections—are of this form and thus, in the future, we will perform a careful study of how the considered random samplers behave under finite precision.

### 3 GENERAL BOUNDS FOR THE TV DISTANCE

Let I be a real interval. To bound the TV distance between two random variables x, η ∈ I, we will use two methods: L^1-norms and interval partitions. The first one is used to approximate a random variable, by approximating its distribution. The second one approximates a random variable by approximating it in several intervals.

For the rest of this section, we use δ_x and δ_η to denote the density functions of x and η respectively.

#### 3.1 Bounds using L^1-norms

The following proposition shows the main technique that we apply for getting bounds on the TV distance.

**Proposition 3.1.** Let x, η ∈ I be continuous random variables. Then
\[ \text{dist}_{TV}(x, η) \leq \|δ_x - δ_η\|_1. \]  

**Proof.** Let B ∈ B. By definition, P(x ∈ B) = \( \int_B δ_x \) and P(η ∈ B) = \( \int_B δ_η \). Therefore, \( |P(x ∈ B) - P(η ∈ B)| = \left| \int_B (δ_x - δ_η) \right| \leq \|δ_x - δ_η\|_1. \)

We recall that a usual way to bound the L^1-norm is to use the L^∞-norm. For a function h : I ⊂ ℝ → ℝ, we have that
\[ \|h\|_1 \leq \lambda(I) \|h\|_∞ \]  

where λ is Lebesgue’s measure—the length.

#### 3.2 Bounds using partitions

The following proposition allows us to bound the total variation distance between x ∈ I and η ∈ I using the information on how these random variables behave on a certain partition \( \{J_i\}_{i=1}^{k} \) of I.

**Definition 3.2.** Let x ∈ I be a continuous random variable and J ⊂ I. The restriction of x to J, x|_J, is the random variable whose density is given by \( (δ_1)_J/ P(x ∈ J) \), if P(x ∈ J) ≠ 0, and by 1/λ(J), where λ is the Lebesgue’s measure, otherwise.

One can see that to sample x|_J using x we only have to sample x until it lies on J and, when this happens, output the sampled value. Note that this method requires on average P(x ∈ J)^{-1} attempts, so the larger P(x ∈ J) is, the more efficient this method becomes.

The following proposition deals with the inverse problem: how do the errors in the partition of an interval add up?

**Proposition 3.3.** Let x, η ∈ I be a random variables and \( \{J_i\}_{i=1}^{k} \) be a partition of I. Then
\[ \text{dist}_{TV}(x, η) \leq \sum_{i=1}^{k} P(x ∈ J_i) \text{dist}_{TV}(x|_{J_i}, η|_{J_i}) + \sum_{i=1}^{k} |P(x ∈ J_i) - P(η ∈ J_i)| \]  

**Proof.** Assume, without loss of generality that for all i, P(x ∈ J_i) and P(η ∈ J_i) are positive. Otherwise, the statement still holds, but the proof is slightly more convoluted. Fix B ∈ B. By the definition of the x|_{J_i}, P(x ∈ B ∩ J_i | x ∈ J_i) = P(x|_{J_i} ∈ B). Therefore
\[ P(x ∈ B) = \sum_{i=1}^{k} P(x|_{J_i} ∈ B)P(x ∈ J_i), \]  

and so, after some elementary operations, we bound \( |P(x ∈ B) - P(η ∈ B)| \) by
\[ \sum_{i=1}^{k} |P(x|_{J_i} ∈ B) - P(η|_{J_i} ∈ B)|P(x ∈ J_i) + \sum_{i=1}^{k} |P(x ∈ J_i) - P(η ∈ J_i)|. \]

Now, by maximizing over B ∈ B, we conclude. □
The above proposition suggests the strategy of partition sampling (Algorithm \textsc{PartitionSampler}). In other words, to sample \( x \), we only need to sample the \( x_i \) and to compute the probabilities \( \mathbb{P}(x \in j_i) \) with enough precision for some partition \( \{j_1^k \}_{i=1} \) of \( I \).

\textbf{Algorithm 1: \textsc{PartitionSampler}}

\textbf{Input:} Partition \( J_1, \ldots, J_k \) of \( I \)
\[ p \in \Delta^{k-1} := \{ p \in \mathbb{R}_+^k \mid \| p \|_1 = 1 \} \]
Approximate samplers \( S_i \) for \( x_i \)

\textbf{Output:} Approximate sample \( x \) for \( x \)
\begin{enumerate}
\item Sample \( i \in \{1, \ldots, k \} \) with probability \( p_i \)
\item \( x \leftarrow S_i \) (We use sampler \( S_i \) to get random \( x \) in \( j_i \))
\item Output \( x \)
\end{enumerate}

We omit the formal proof that this procedure gives an efficient sampler if the \( S_i \) are efficient samplers. To see this, we only have to note that this method’s run-time will be at most the run-time of the \( S_i \). As we can precompute the probabilities \( \mathbb{P}(x \in j_i) \), we can choose sufficiently good values for \( p \) within the required restrictions for almost all cases we consider here.

4 INVERSE TRANSFORM SAMPLING BY BISECTION

Let \( I = [a, b] \subset \mathbb{R} \) and \( \varphi : I \to [0, \infty) \) be a density function. Inverse transform sampling is based on the fact that the solution \( \tilde{x} \in I \) of
\[ \int_a^x \varphi(s) \, ds = u, \text{ for } u \in [0, 1] \text{ uniformly distributed, has density } \varphi \]
(see Alg. 2 for its pseudocode).

\textbf{Algorithm 2: \textsc{InverseTransformSampler}}

\textbf{Input:} \( I = [a, b] \), \( \varphi : I \to [0, \infty) \) such that \( \int_a^b \varphi(t) \, dt = 1 \)
\textbf{Output:} \( x \sim \varphi \)
\begin{enumerate}
\item Sample \( u \in [0, 1] \) uniformly
\item Find the solution \( x \) of \( \int_a^x \varphi(s) \, ds = u \)
\item Output \( x \)
\end{enumerate}

However, any reader of this pseudocode will be suspicious about all the details swept under the rug in line 2. How do we solve \( \int_a^x \varphi(s) \, ds = u \)? And how fast can we do it? Even though this is an important question regarding the complexity of sampling, we feel that it is unaddressed by the literature, so we discuss it.

Let the cumulative distribution function corresponding to the density \( \varphi \) be
\[ \Phi(x) := \int_a^x \varphi(s) \, ds. \]
We want to solve the equation
\[ \Phi(x) = u \in [0, 1]. \]
When solving this equation, the bisection method outputs an interval containing the root. This interval is found by repeatedly subdividing the initial interval, and selecting the one such that \( u - \Phi \) has different signs at the two endpoints. Let \( \ell \in \mathbb{N} \); we stop subdividing after \( \ell \) iterations. In the end, we choose a point at random in the final interval. We integrate the bisection method in inverse transform sampling in the algorithm \textsc{BisectionSampler}.

\textbf{Algorithm 3: \textsc{BisectionSampler}}

\textbf{Input:} \( \varphi : [a, b] \to [0, \infty) \) such that \( \int_a^b \varphi(t) \, dt = 1 \) \( \ell \in \mathbb{N} \)
\textbf{Output:} Approximate sample \( x \) for \( \varphi \)
\begin{enumerate}
\item Sample \( u \in [0, 1] \) uniformly, \( i \leftarrow 0 \)
\item \( x_l \leftarrow a, x_r \leftarrow \text{sign} (u - \Phi(x_l)), x_r \leftarrow b, \)
\item \( u_r \leftarrow \text{sign} (u - \Phi(x_r)) \)
\item \( \text{while } i < \ell \text{ do} \)
\item \( x_m \leftarrow (x_l + x_r) / 2, u_m \leftarrow \text{sign} (u - \Phi(x_m)) \)
\item \( \text{if } u_l = u_m \text{ then} \)
\item \( x_r \leftarrow x_m, u_r \leftarrow u_m, i \leftarrow i + 1 \)
\item \( \text{else} \)
\item \( x_l \leftarrow x_m, u_l \leftarrow u_m, i \leftarrow i + 1 \)
\item Sample \( \tilde{x} \in [x_l, x_r] \) uniformly
\item Output \( \tilde{x} \)
\end{enumerate}

The following theorem shows that \textsc{BisectionSampler} produces a nice sampler.

\textbf{Theorem 4.1.} Let \( \varphi : [a, b] \to [0, \infty) \), \( x \sim \varphi \) and \( \tilde{x}_\ell \) the output of \textsc{BisectionSampler} for \( \ell \in \mathbb{N} \). Then
\[ \text{dist}_{TV}(x, \tilde{x}_\ell) \leq 2^{-\ell} |b - a| \max_{x \in [a, b]} |\varphi'(x)|. \] (4.3)

\textbf{Proof.} Without loss of generality, assume that \( a = 0 \). For \( k \in \{0, \ldots, 2^{\ell} - 1 \} \), let \( J_k := [bk/2^\ell, b(k + 1)/2^\ell] \). The \( \tilde{x}_\ell \) produced by \textsc{BisectionSampler} can also be produced as follows:

1. Choose the interval \( J_k \) at random with probability \( \mathbb{P}(x \in J_k) \).
2. Note that the solution of (4.2) lies in \( J_k \) with probability \( \mathbb{P}(x \in J_k) \).

(2) Sample \( u_k \in J_k \) uniformly.

Therefore, by Proposition 3.3, to compute \( \text{dist}_{TV}(x, \tilde{x}_\ell) \), we only have compute the TV distance between \( x \) and the uniformly distributed \( u_k \in J_k \). Now, after an elementary computation and Proposition 3.1,
\[ \text{dist}_{TV}(x, \tilde{x}_\ell) \leq \mathbb{P}(x \in J_k)^{-1} |\varphi|_{J_k} - 2^\ell \mathbb{P}(x \in J_k)|. \] (4.4)

Now, by the mean value theorem, \( \mathbb{P}(x \in J_k) = \varphi(\tilde{x}_k)/2^\ell \) for some \( \tilde{x}_k \in J_k \), so we obtain
\[ \text{dist}_{TV}(x, \tilde{x}_\ell) \leq \mathbb{P}(x \in J_k)^{-1} |\varphi|_{J_k} - \varphi(\tilde{x}_k)|. \] (4.5)

Now, \( |\varphi(s) - \varphi(\tilde{x}_k)| \leq b2^{-\ell} \max_x \varphi'(x) \), by the mean value theorem, and so \( \text{dist}_{TV}(x, \tilde{x}_\ell) \leq \mathbb{P}(x \in J_k)^{-1} b2^\ell \max_x \varphi'(x) \), concluding the proof. \( \Box \)

Note that \textsc{BisectionSampler} needs to perform a minimum of
\[ \max \left\{ 0, \sup \{ \log |\varphi'(x)| \mid x \in [a, b] \} \right\} \]
iterations. Moreover, note that in the bisection method, we can interchange precomputation and computation with no effect to our notion of the efficient sampler.
5 SAMPLING POINTS FROM A CURVE

Let $\gamma : I = [-1, 1] \to \mathbb{R}^n$ be the parametrization of a real polynomial curve of degree $d$—after a linear change of coordinates we can always assume that $I = [-1, 1]$. Since we want to generate random points $x \in \gamma(I)$ uniformly with respect to the arc-length, we only need to sample a random parameter $t \in I$ distributed according to the normalized speed

$$\varphi(t) = \left( \int_{t_1}^t \| y'(s) \| ds \right)^{-1} \| y'(t) \|,$$  \hspace{1cm} (5.1)

and then take the random variable $\gamma(t) \in \gamma(I)$ which will have the desired distribution.

When $n \geq 2$, we have that $\varphi$ is not a polynomial. Because of this, to perform the inverse transform sampling, even by bisection, we will approximate $\varphi$ by a Chebyshev approximation $\tilde{\varphi}$ for which computing

$$\tilde{\varphi}(t) = \int_{t_1}^t \tilde{\varphi}(s) ds$$

is a lot easier than computing

$$\Phi(t) = \int_{t_1}^t \varphi(s) ds.$$

Now, to make the Chebyshev approximation faster, we will split the interval into subintervals.

First, we review the Chebyshev approximation; then, we apply it to the case of interest; finally, we show how splitting accelerates the Chebyshev approximation. The algorithm appears in Algorithm 4. Again, we observe that this algorithm is very similar to the one proposed in [23], but our main contribution is not the sampler itself but the error analysis using the total variation distance.

5.1 Chebyshev approximations

We follow mainly [19] and [28]. Recall that the $k$th Chebyshev polynomial is the polynomial given by

$$\mathcal{U}_k(X) = \sum_{i=0}^{k} \binom{k}{i} (1-X^2)^i X^{k-2i},$$  \hspace{1cm} (5.2)

where $\mathcal{U}$ is the first initial of Chebyshev in the Cyrillic script. Alternatively, note that $\mathcal{U}_k$ satisfies

$$\mathcal{U}_k(x) = \cos(k \arccos(x)).$$  \hspace{1cm} (5.3)

for $x \in I$. As a consequence, the $k$ zeros of $\mathcal{U}_k$ are given by

$$z_{a,k} = \cos\left( \frac{(1+2a)\pi}{2k} \right)$$  \hspace{1cm} (5.4)

with $a \in \{0, \ldots, k-1\}$.

5.1.1 Chebyshev interpolation. The $k$th Chebyshev interpolant of $f : I \to \mathbb{R}$ is the unique degree $k$ polynomial $\mathcal{U}_k(f)$ satisfying for $a \in \{0, \ldots, k\}$,

$$\mathcal{U}_k(f)(z_{a,k}) = f(z_{a,k}).$$  \hspace{1cm} (5.5)

To compute $\mathcal{U}_k(f)$, there is no need to solve the system above thanks to the following proposition.

Proposition 5.1. [19, Thm. 6.7] Let

$$\mathcal{U}_k(f) = \frac{c_0}{2} + \sum_{a=1}^{k} c_a \mathcal{U}_a$$

be the $k$th Chebyshev interpolant of $f : I \to \mathbb{R}$. Then

$$c_a = \frac{2}{k+1} \sum_{i=0}^{k} f(\xi_{i,k} \mathcal{U}_a(\xi_{i,k}) - \xi_{i,k}).$$

\hspace{1cm} \square

Let us remind that $\mathcal{U}_k(f)$ is not equivalent to truncating the Chebyshev series up to degree $k$ [28, Ch. 4].

5.1.2 Evaluation of Chebyshev interpolants. Given a Chebyshev interpolant $\mathcal{U}_k(f)$, we can expand it in the monomial basis and then evaluate it using, for example, Ruffini-Horner’s method. However, we have a version of Ruffini-Horner’s method that works directly for Chebyshev expansions of a polynomial.

Proposition 5.2. [9, pp. 55-56] Let $p = \sum_{a=0}^{k} c_a \mathcal{U}_k$, then for every $x \in \mathbb{R}$,

$$p(x) = \frac{1}{2} (\Gamma_0(x) - \Gamma_2(x))$$  \hspace{1cm} (5.6)

where $\Gamma_0(x)$ and $\Gamma_2(x)$ are computed through the following backwards-recursive relation

$$\begin{cases}
\Gamma_{k+1}(x) = \Gamma_{k+2}(x) = 0 \\
\Gamma_a(x) = 2x \Gamma_{a+1}(x) - \Gamma_{a+2}(x) + c_a.
\end{cases}$$

\hspace{1cm} \square

5.1.3 Speed of convergence. To estimate the error of the Chebyshev approximation, we consider the so-called Bernstein ellipse given by

$$E_{\rho} := \{ z \in \mathbb{C} \mid |z + \sqrt{z^2 - 1}| = \rho \},$$

where $\rho > 1$. Note that $E_{\rho}$ is the ellipse with foci $-1$ and 1 and focal distance $\rho + \rho^{-1}$. We have the following theorem.

Theorem 5.3. [28, Theorem 8.2]. If $f$ is analytic on the elliptic disc given by $E_{\rho}$, then

$$\| f - \mathcal{U}_k(f) \|_\infty \leq \frac{4 \| f \|_{E_{\rho}, \rho^{-k}}}{\rho - 1}.$$  \hspace{1cm} (5.7)

where $\| f \|_{E_{\rho}} := \max_{z \in E_{\rho}} |f(z)|$.

To obtain theoretical bounds of $\| f \|_{E_{\rho}}$, the following inequality by Bernstein will make our job easier.

Theorem 5.4. [28, Exercise 8.6] Let $f$ be a polynomial of degree $d$ and $\rho > 1$. Then

$$\| f \|_{E_{\rho}} \leq \rho \rho \| f \|_\infty.$$  \hspace{1cm} \square

Note that this inequality does not serve to approximate polynomials by a Chebyshev interpolant of a lower degree. However, we will use it to control the quantity of interest for $\varphi$, i.e., the normalized speed of $\varphi$ (5.1). The hard part will be estimating a sufficiently small $\rho$ so that $\varphi$ admits an analytic extension to the interior of $E_{\rho}$. 

\hspace{1cm} On the Error of Random Sampling: Uniformly Distributed Random Points on Parametric Curves
5.1.4 Integration formulas. Imagine we want to compute the integral (definite or indefinite) of $\Psi_k(f)$. To do this, we use the following proposition.

**Proposition 5.5.** [9, pp. 54-55][19, pp. 45-46, 59] Let

$$ p = \frac{c_0}{2} + \sum_{a=1}^{k} c_a \Psi_a. $$

Then

$$ \sum_{a=1}^{k+1} p_{ca} \Psi_a $$

with

$$ p_{ca} = \begin{cases} \frac{c_{a-1} - c_{a+1}}{2a}, & \text{if } a = 1, \ldots, k - 1 \\ \frac{c_{a-1}}{2a}, & \text{if } a = k \\ \frac{c_{a+1}}{2a}, & \text{if } a = k + 1 \end{cases} $$

is a primitive function of $p$. Moreover,

$$ \int_{-1}^{1} p(x) \, dx = c_0 - \sum_{a=1}^{n} \left( 1 + (-1)^a \frac{a}{a^2 - 1} \right) c_a. \square $$

The following fact will be useful later. Let $F(t) := \int_{-1}^{t} f(s) \, ds$ for $t \in [-1, 1]$, then

$$ ||F||_\infty \leq ||f||_1 \leq 2 ||f||_\infty. $$

Hence, Theorem 5.3 allows us to also control the error of the integral approximation.

5.2 Bernstein ellipse for the speed

The following theorem shows that the conditions of Theorem 5.3 are satisfied for the speed $\varphi$ and gives possible $\rho$s we can take.

**Theorem 5.6.** Let $\varphi$ be given as in (5.1), non-vanishing in $I$. Let $\rho > 1$ be such that $\rho$ does not exceed

$$ \rho^* = \left( \min_{|\varphi(z)| = 0} \left\{ \frac{|z + 1| + |z - 1| + \sqrt{(|z + 1| + |z - 1|)^2 - 4}}{2} \right\} \right) \geq 1, $$

Then $\varphi$ admits an analytic extension $\varphi_{an}$ to the interior of $E_\rho$ and

$$ ||\varphi_{an}||_{E_\rho} \leq \rho^2 ||\varphi||_\infty. $$

**Proof.** Note that $\varphi$ is the square root of the polynomial $||\varphi'||^2$. To analytically extend such a function to the interior of the ellipse $E_\rho$, we need to know that no complex root $z$ of $||\varphi'||^2$ lies inside $E_\rho$. Now, $z$ lies inside the interior of $E_\rho$ if and only if

$$ \rho \geq \frac{|z + 1| + |z - 1| + \sqrt{(|z + 1| + |z - 1|)^2 - 4}}{2}. $$

Recall that $E_\rho$ is the ellipse with foci $-1$ and $1$ and focal distance $\rho + \rho^{-1}$, so $z$ lies in its interior if and only if $|z + 1| + |z - 1| \leq \rho + \rho^{-1}$. The latter is equivalent to the above inequality for $\rho$.

Since $\varphi_{an}$ is a polynomial of degree 2d on $I$, it is a polynomial of degree 2d on the interior of $E_\rho$. Hence, by Bernstein’s inequality (Theorem 5.4),

$$ ||\varphi_{an}||_{E_\rho} \leq \rho^{2d} ||\varphi_{an}||_\infty = \rho^{2d} ||\varphi^2||_\infty. $$

Since the square root of the maximum is the maximum of the square root, the desired bound follows. \qed

**Remark 5.7.** Note that Theorem 5.6 gives a conservative bound for $||\varphi||_{E_\rho}$, which we use for giving an upper bound for the complexity. Since we can precompute this quantity, we do this in the off-line part of CURVESAMPLER, so that we get better run times.

Observe that $\rho^*(\gamma)$ is optimal. In this way, if we want a Chebyshev interpolant $\Psi_k(\varphi)$ such that $||\Psi_k(\varphi) - \varphi||_\infty \leq \varepsilon$, then the degree of this interpolant has to satisfy

$$ k \geq \frac{1}{\log \rho^*(\gamma)} \left( \log \log \frac{1}{\varepsilon} + \log ||\varphi||_{E_{\rho^*}} (\gamma) + 2 - \log(\rho^*(\gamma) - 1) \right) $$

by Theorem 5.3.

We now give theoretical bounds for $\rho^*$. But before, let us define the geometric parameter that will appear in these bounds.

**Definition 5.8.** Let $\gamma : I \to \mathbb{R}^n$ be the parametrization of a real polynomial curve of degree $d$ such that the polynomial in the rth component, $\gamma_r$, is given by

$$ \gamma_r = \sum_{j} \gamma_{r,j} T^j. $$

Then the condition number for sampling $\gamma$, $C(\gamma)$, is quantity

$$ C(\gamma) := \frac{||\gamma'||_2}{\inf_{t \epsilon I} ||\gamma'(t)||_2} \in [1, \infty] $$

where $||\gamma'||_2 = \sum_{i=1}^{n} \sum_{j} |\gamma_{i,j}|$ is the sum of the absolute value of all coefficients of $\gamma$ multiplied each by the degree of their term.

**Remark 5.9.** The condition number for sampling $\gamma$, $C(\gamma)$, is finite as long as $\varphi$ is non-vanishing in $I$. Note that the idea is that the nearer is $\gamma$ to have zero speed at a point, the harder it is to sample a random point in it.

**Theorem 5.10.** Let $\gamma : I \to \mathbb{R}^n$ be the parametrization of a real polynomial curve of degree $d$ with non-vanishing speed. Then

$$ \rho^*(\gamma) \geq 1 + \frac{1}{\varepsilon \cdot d \cdot C(\gamma)}. $$

**Proof.** Since $\varphi$ is non-vanishing, $C(\gamma) < \infty$. Let $\epsilon \in (0, 1/\rho)$ and consider $I_{\epsilon} := \{ z \in \mathbb{C} \mid \text{dist}(z, I) \leq \epsilon \}$. Now, by [27, Proposition 3.6], for each $i$, the map

$$ I_{\epsilon} \ni z \mapsto ||\gamma_i'(z)||/||\gamma_i'||_o $$

is $(\epsilon \cdot d)$-Lipschitz. Hence, the map

$$ I_{\epsilon} \ni z \mapsto ||\gamma_i'(z)||/||\gamma_i'||_o $$

is $(\epsilon \cdot d)$-Lipschitz. In this way, if $\varepsilon = 1/(\epsilon \cdot d \cdot C(\gamma))$, we have that $||\gamma_i'||_o$ does not have zeros inside $I_{\epsilon}$. Now, if for some $\rho \geq 1$, $E_{\rho} \subseteq I_{\epsilon}$, then $\rho \leq \rho^*(\gamma)$.

By the definition of $E_\rho$, $E_\rho \subseteq I_{\epsilon}$ if and only if a) $(\rho + \rho^{-1})/2 - 1 \leq \epsilon$ (major semiaxis bound) and b) $\sqrt{\rho^2 + \rho^{-2} - 2}/2 \leq \epsilon$ (minor semiaxis bound).

Now, $\rho = 1 + \epsilon$, with $\epsilon = 1/(\epsilon \cdot d \cdot C(\gamma)) \leq 1$, satisfies these inequalities. Hence the claim follows. \qed
5.3 Acceleration through splitting the curve

Whenever we split the curve, we should expect the value of $\rho^*(y)$ to increase. The reason for this is that, after renormalization of a smaller interval, the zeros of $y'$ are further away and so the value of $\rho$ should increase. However, we observe that this increase will depend on the value of $\rho^*(y)$, so there is not a uniform constant factor improvement independent of $\rho^*(y)$.

The above paragraph suggests that we can just perform a fixed number of binary subdivision steps to accelerate the algorithm. This strategy does indeed accelerate the sampler, as shown by experiments (see subsection 3.6 and Figure 3).

To conclude, note that as we compute $\rho^*(y)$, we have to compute also the complex roots of $y'$. Therefore we can split the interval $I$ along the points

$$z_1, \ldots, z_d$$

where $z_1, \ldots, z_d, \overline{z}_d$ are the complex roots of $y'$. This subdivision accelerates the algorithm significantly as it forces the roots of $y'$ to lie on the endpoints of each interval or outside.

5.4 The sampler

We give the sampler for the curve $y : I \to \mathbb{R}^n$, CurveSampler, without indicating the subdivision procedure. We observe that excepting the last call, the sampler performs the majority of its operations off-line, so they don’t have to be repeated in each call.

**Algorithm 4: CurveSampler**

```
Input: $y : I \to \mathbb{R}^n$ of degree $d$
$\ell \in \mathbb{N}$
Output: Approximate sample $t$ of

$\varphi := \|y''(t)/\int_{-1}^1 y''(s)ds\|_2$ /* off-line */

1. $\varphi \leftarrow \|y''(z)/\int_{-1}^1 y''(s)ds\|_2$ /* off-line */
2. $Z \leftarrow \{z \in \mathbb{C} \mid y'(z) = 0\}$ /* off-line */
3. $\gamma^* \leftarrow \min_{z \in Z} \frac{z+1|z-1|+\sqrt{(z+1)^2+(z-1)^2}}{2}$ /* off-line */
4. $M \leftarrow \|\varphi\|_{E,\rho}$ /* off-line */
5. $\ell \leftarrow 5 + \ell + (\log M - \log (\rho^*-1))/\log \rho^*$ /* off-line */
6. $\hat{\varphi} \leftarrow \|\hat{y}^k(\varphi)/\int_{-1}^1 \hat{y}^k(\varphi)(s)ds\|_2$ /* off-line */
7. $\hat{\gamma} \leftarrow \text{Integral of } \hat{y}^k(\varphi)$ /* off-line */
8. $t_0 \leftarrow 1 + \ell + \max(0, \log \|\hat{\varphi}\|_\infty)$ /* off-line */
9. $t \leftarrow \text{BisectionSampler}(\hat{\varphi}, t_0)$
Output $t$
```

5.5 Complexity of CurveSampler

Recall that we are working in the BSS model with square roots, so we assume that we can evaluate $\|y''(t)\|_2$ exactly. Our main theorem is the following one.

**Theorem 5.11.** Let $y : I \to \mathbb{R}^n$ be a polynomial parameterized curve. CurveSampler is an efficient sampler for $t \in 1$ uniformly distributed with respect to the normalized speed of $y$. Moreover, it performs

$$O(t^2(1 + \log dC(y))^2dC(y)^3)$$

off-line arithmetic operations, where $O$ has constants depending on the degree of $y$; and

$$O(t^3(1 + \log dC(y))^3dC(y)^3)$$

on-line arithmetic operations to achieve an error of $2^{-t}$ in the TV distance.

**Remark 5.12.** Even though we are ignoring the complexity of the offline part—many of those parts can be done in poly($d$) arithmetic operations up to the desired degree of precision. This is why we focus on the dependence on the error.

**Proof of Theorem 5.11.** On the one hand, by Theorem 5.3,

$$\|\gamma - \varphi\|_1 \leq 2\|\hat{y}^k(\varphi) - \varphi\|_1 \leq \frac{16M}{\rho^*-1}(\rho^*)^{k-1} \leq 2^{-t(1+t)}$$

(5.10)

where the first inequality follows from $\|\varphi\|_1 = 1$, and so if $s \sim \varphi$, then, by Proposition 3.1,

$$\text{dist}_{TV}(s, t) \leq 2^{-t(1+t)}$$

(5.11)

where $t \sim \varphi$. On the other hand, by Theorem 4.1,

$$\text{dist}_{TV}(t, 1) \leq 2^{1-t}\|\varphi\|_\infty \leq 2^{-(1+t)}.$$ 

Hence,

$$\text{dist}_{TV}(s, t) \leq 2^{-t}$$

and so to show that CurveSampler is an efficient sampler for $\varphi$, we only need to bound the complexity as desired.

For the off-line part, we need to bound the number of arithmetic operations in terms of $t$. In line 2, we use some solver; for line 3, we just minimize over the roots found in line 2; for line 4, we only have to parameterize the boundary of $E_{\rho^*}$ and find the minimum, due to the maximum modulus principle; for line 5, we do the assignment; for line 6, we use Proposition 5.1 and the second part of Proposition 5.5; for line 7, we use the first part of Proposition 5.5. By observing these, we see that the number of arithmetic operations is at most $O(k^2)$, which by the definition of $k$ in line 5 and Theorems 5.6 and 5.10 transforms to

$$O(t^2(1 + \log dC(y))^2dC(y)^3)$$

where we use that $1/\log(1 + x) \leq 2/x$.

For the on-line part, we perform $t_0$ evaluations of $\hat{\varphi}$, making each evaluation, by Proposition 5.2, $O(k)$ operation. Thus we perform $O(t_0k)$ arithmetic operation. We have to bound $t_0$ now. However, this is equivalent to bounding $\|\hat{\varphi}\|_\infty$.

Using the theory of Chebyshev polynomials [28], we have that

$$\|\hat{y}^k(\varphi) - \varphi\|_1 \leq 16Mk^2(\rho^*)^{3-k}/((\rho^*) - 1).$$

Thus,

$$\|\hat{\varphi}\|_\infty \leq \|\varphi\|_\infty + \frac{16Mk^2\rho^*^{3-k}}{\rho^*-1} \leq \|\varphi\|_\infty + 4k^2(\rho^*)^{2-t}.$$ 

Now, on the one hand,

$$4k^2(\rho^*)^{2-t} \leq O(t^2(1 + \log dC(y))^2dC(y)^3);$$

and on the other hand,

$$|\varphi| = \|y''(t)/\|y''\|_2 \leq \|y''(t)/\|y''\|_2^2$$

where $\|y''\|_2 \leq d\|y''\|_2$ by [27, Proposition 3.6]. Thus $\|\varphi\|_\infty \leq dC(y)$. Putting this together, we obtain the bound $t_0 \leq O(t + \log dC(y) + t^2(1 + \log dC(y))^2dC(y)^2)$. Hence we are done. □
6 IMPLEMENTATION AND EXPERIMENTS

We provide an open-source implementation in Matlab of the studied method that can be accessed at

github.com/TolisChal/sampling_curves

It is an original implementation, up to technical modifications, of CurveSampler using BisectionSampler which allows us to sample random points from a given parametric polynomial curve uniformly with respect to the arc-length.

Our implementation relies on a few standard routines from Matlab’s toolbox. In particular, we use (i) chebyshevT() to evaluate the kth degree Chebyshev polynomial \( T_k(x) \), (ii) roots() to compute the zero set of the speed \( \|y'(t)\|_2^2 \) for the computation of \( \rho^* \) (line 3 in CurveSampler), and (iii) fmincon() to solve the optimization problem required to compute \( \|f\|_{E_n} \), (line 4). All computations were performed on a PC with Intel® Pentium(R) CPU G4400 @ 3.30GHz × 2 CPU and 16GB RAM.

6.1 The example of Figure 1

In the example of Figure 1, we execute CurveSampler to produce a sample of 300 random points. During the execution, we can see that the speed of the curve \( y : [-1, 1] \ni t \mapsto (3t^2 - 2t, 2t^3) \),

\[
\|y'(t)\|_2 = \sqrt{42t^2 - 24t + 4},
\]

is approximated by a Chebyshev approximation of degree 35. In this example, we took \( t = 4 \), so that \( 2^{-t} < 0.1 \).

6.2 Random experiments: Table 1

In Table 1, we show the results of performing several random experiments. For these experiments, we consider random parametric polynomial curves \( y : [-1, 1] \ni t \mapsto (3t^2 - 2t, 2t^3) \) with \( n \in \{20, 40, 60, 80, 100\} \) components and with error specifications \( 2^e \in \{10, 100\} \). The coefficients of the polynomials in \( y \) are independent, identically distributed standard Gaussian random variables. In the table, we display the degree of the Chebyshev interpolant (k), the preprocessing time (Pr. T) in seconds and the time per generated sample after preprocessing (T/s) in seconds.

We can see that the degree \( k \) of the Chebyshev interpolant increases with the degree of \( d \) of the polynomial curve. The run-time of the preprocessing takes a few seconds, while the time per sample after preprocessing is smaller than 1 second for every instance.

6.3 A curve of degree 10: Figures 2 and 3

In Figures 2 and 3, we consider a parametric polynomial curve \( y : [-1, 1] \ni t \mapsto (3t^5 + T) \) of degree 10. In both cases, we plot the degree of the used Chebyshev interpolant (k) with respect to the inverse of the error \( 2^e \).

In Figure 2, we simply show the evolution of the degree. In Figure 3, we illustrate the theoretical discussion of subsection 5.3 by plotting the degree used when we don’t split the interval (blue line) against the maximum degree used when we split the interval into four sub-intervals. This shows that a few subdivisions can significantly reduce the degree of the used Chebyshev interpolants.

6.4 A polynomial density: Figures 4 and 5

We consider the curve \( y : [-1, 1] \ni t \mapsto (1 + T + \cdots + T^d)(1, 1, 1) \in \mathbb{R}^3 \) to have an explicit example where we can see the evolution of the algorithm with respect the degree \( d \). In Figure 4, we can see the the run-time per sample after preprocessing against \( d \); and, in Figure 5, we can see the degree of the Chebyshev interpolant (k) against the degree. Interestingly, both increase sub-linearly with the degree \( d \).

7 CONCLUSIONS

In this paper, we initiated the study of the errors in random sampling in the context of algebraic geometry by studying a sampling method in the context of parametric polynomial curves. More precisely, we show that the method in [23] is efficient for generating random point on a parametric polynomial curve both theoretically (see Theorem 5.11) and in practice (see §6). However, this is just the first step towards obtaining error bounds for the methods generating random points in algebraic varieties—needed for theoretical guarantees for applications of TDA in algebraic geometry [2].

Interestingly, the experiments suggest that the considered method might be faster than what our theoretical estimates suggest. This discrepancy might be because the bounds in Theorem 5.6 or 5.10 are too pessimistic. To improve these in the future, we might need a sharper definition of the condition number \( C(y) \) or substituting some of the inequalities—especially those in which the degree appears—by inequalities that adapt better to the geometry of each curve. Thus, we feel that further theoretical work is needed to fully understand CurveSampler. We also note that alternative strategies to BisectionSampler, for example, those using Newton’s method, need to be analyzed in the future.

In an extended version of this paper, we will include: (1) The analysis of the algorithm under the assumption of finite precision. (2) Variations of the condition number \( C(y) \), introduced in Definition 5.8, and its analysis in terms of the bit-size and for a random \( y \). (3) A comparison with methods relying on approximate arc-length parametrizations or deterministic samples.

Acknowledgements. The last author is supported by a postdoctoral fellowship of the 2020 “Interaction” program of the Fondation Sciences Mathématiques de Paris. He is grateful to Evgenia Lagoda for moral support and Gato Suchen for the mathematical discussions regarding §4.

The last two authors are partially supported by ANR JCJC GALOP (ANR-17-CE40-0009), the PGMO grant ALMA, and the PHC GRAPE.

The authors are thankful to Elias Tsigaridas and the referees for useful comments and suggestions that helped improve the quality of this paper.

For example, we allow \( 2^e \) to be any real number and not only a power of 2.
On the Error of Random Sampling: Uniformly Distributed Random Points on Parametric Curves

Table 1: Random experiments: degree of the Chebyshev interpolant ($k$), preprocessing time ($Pr. T$) and time per sample after preprocessing ($T/s$) in terms of the degree ($d$), the ambient dimension ($n$) and the error ($2^\ell$)

<table>
<thead>
<tr>
<th>$d$</th>
<th>$n$</th>
<th>$k$</th>
<th>$Pr. T$</th>
<th>$T/s$</th>
<th>$k$</th>
<th>$Pr. T$</th>
<th>$T/s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>20</td>
<td>2</td>
<td>0.82</td>
<td>0.53</td>
<td>24</td>
<td>0.94</td>
<td>0.58</td>
</tr>
<tr>
<td>40</td>
<td>32</td>
<td>1.08</td>
<td>0.44</td>
<td>0.39</td>
<td>39</td>
<td>1.28</td>
<td>0.47</td>
</tr>
<tr>
<td>5</td>
<td>60</td>
<td>0.92</td>
<td>0.44</td>
<td>0.32</td>
<td>32</td>
<td>1.09</td>
<td>0.48</td>
</tr>
<tr>
<td>8</td>
<td>29</td>
<td>1.04</td>
<td>0.43</td>
<td>0.34</td>
<td>34</td>
<td>1.18</td>
<td>0.45</td>
</tr>
<tr>
<td>100</td>
<td>25</td>
<td>1.24</td>
<td>0.54</td>
<td>0.29</td>
<td>30</td>
<td>1.30</td>
<td>0.59</td>
</tr>
<tr>
<td>20</td>
<td>34</td>
<td>1.34</td>
<td>0.57</td>
<td>0.40</td>
<td>40</td>
<td>1.51</td>
<td>0.60</td>
</tr>
<tr>
<td>40</td>
<td>21</td>
<td>0.76</td>
<td>0.40</td>
<td>0.27</td>
<td>41</td>
<td>0.89</td>
<td>0.42</td>
</tr>
<tr>
<td>10</td>
<td>60</td>
<td>1.17</td>
<td>0.44</td>
<td>0.41</td>
<td>1.37</td>
<td>0.47</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>38</td>
<td>1.31</td>
<td>0.53</td>
<td>0.44</td>
<td>1.65</td>
<td>0.56</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>37</td>
<td>1.64</td>
<td>0.64</td>
<td>0.43</td>
<td>2.03</td>
<td>0.65</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>35</td>
<td>1.29</td>
<td>0.59</td>
<td>0.41</td>
<td>1.70</td>
<td>0.61</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>46</td>
<td>1.53</td>
<td>0.48</td>
<td>0.53</td>
<td>1.70</td>
<td>0.50</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>60</td>
<td>3.40</td>
<td>0.49</td>
<td>0.54</td>
<td>3.54</td>
<td>0.50</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>43</td>
<td>1.65</td>
<td>0.49</td>
<td>0.50</td>
<td>1.90</td>
<td>0.53</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>49</td>
<td>2.36</td>
<td>0.63</td>
<td>0.55</td>
<td>2.36</td>
<td>0.65</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>67</td>
<td>2.82</td>
<td>0.70</td>
<td>0.79</td>
<td>4.11</td>
<td>0.82</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>63</td>
<td>2.07</td>
<td>0.53</td>
<td>0.72</td>
<td>2.65</td>
<td>0.57</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>51</td>
<td>2.38</td>
<td>0.66</td>
<td>0.58</td>
<td>2.24</td>
<td>0.56</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>70</td>
<td>2.60</td>
<td>0.59</td>
<td>0.80</td>
<td>4.38</td>
<td>0.76</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>56</td>
<td>2.55</td>
<td>0.62</td>
<td>0.63</td>
<td>3.24</td>
<td>0.65</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: Degree of the Chebyshev interpolant ($k$) against the inverse error ($2^\ell$) for a curve of degree 10 in $\mathbb{R}^{50}$.

Figure 3: Maximum degree of the Chebyshev interpolant ($k$) against the inverse error ($2^\ell$) for a curve of degree 10 in $\mathbb{R}^{50}$ without splitting the interval (blue) and splitting the interval in four equal intervals (red).

Figure 4: Time per sample after preprocessing against the degree ($d$) for the 3D curve with each coordinate being the degree $d$ polynomial $1 + T + \cdots + T^d$.

REFERENCES


Figure 5: Degree of the Chebyshev interpolant \( k \) against the degree \( d \) for the 3D curve with each coordinate being the degree \( d \) polynomial \( 1 + T + \cdots + T^d \).

http://dx.doi.org/10.1016/B978-0-12-543457-7.50032-2


