HAL open science

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-03601563v1

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
chistina.katsamaki@inria.fr

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

arXiv:2203.02832v2 [cs.CG] 3 May 2022

ABSTRACT

Given a parametric polynomial curve $\gamma : [a, b] \to \mathbb{R}^n$, how can we sample a random point $\mathfrak{x} \in \mathrm{im}(\gamma)$ 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 $\gamma : I := [a, b] \to \mathbb{R}^n$, we are interested in generating a random point $\mathfrak{x} \in \gamma(I)$ that is uniformly distributed with respect the arc-length. To do this, we only need to sample a random variable $\mathfrak{t} \in I$ with density proportional to the speed of the curve $\|\gamma'\|_2$. Even if we perform exact arithmetic operations using real numbers with infinite precision, this problem does not admit an exact solution—the integral $\int \|\gamma'\|_2$ 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 [13]. In the context of parametric curves, this refers to generating a finite subset of points that captures the curve [24]. Of course, random sampling can be used for generating samples in the second sense, but it is not the best method since randomness might produce points too near to the ones already produced.

Now, being able to sample random points in an algebraic variety plays an important role in the application of topological data analysis (TDA) to algebraic geometry [2]—if we sample enough random points on an algebraic variety, we can determine its topology as shown in Figure 1. However, in order to bridge the gap between the theoretical assumptions on the random samples (as those stated in [22]) and the random samples that we actually generate, we have to understand how imperfect our generation of these random samples is. This paper is a first step towards filling this gap in our theoretical understanding in the simplest case: sampling random points on parametric polynomial curves.

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:

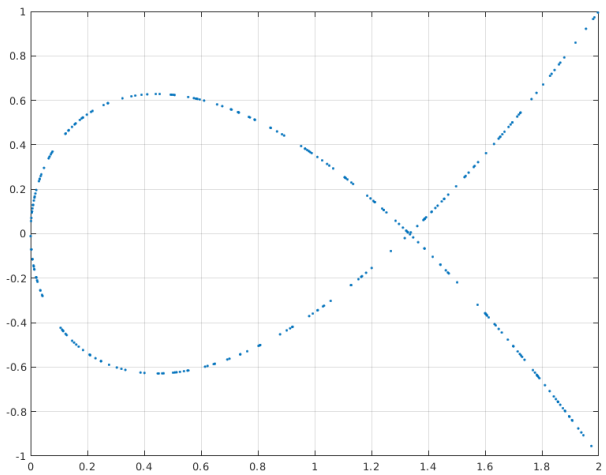

Figure 1: A sample of 300 random points from the curve $\gamma : [-1,1] \to \mathbb{R}^2$ give by $\gamma(t) = (3t^3 - 2t, 2t^2)$ generated by our algorithm CurveSampler. For more details on this example see 6.1.

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

$$O(\ell^3(1+\log d\mathtt{C}(\gamma))^3 d^3 \mathtt{C}(\gamma)^3)$$

arithmetic operations and with error $2^{-\ell}$ with respect to the total variation distance, where $\mathtt{C}(\gamma)$, the condition number for sampling γ, 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:

$$\left\| \|\gamma'(\theta(s))\|_2 \theta'(s) - \chi_{[0,L]}(s) \right\|_1,$$

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.

Notation

For a real function $h : I \subseteq \mathbb{R} \to \mathbb{R}$, its L^1 norm is $\|h\|_1 := \int_{\mathbb{R}} |h(x)|\, ds$, where, by convention, we take $h(x) = 0$ for $x \notin I$. Its L^∞-norm is defined as $\|h\|_\infty := \sup_{x\in I} |h(x)|$.

Given a random variable $\mathfrak{x} \in I$, we will write $\mathfrak{x} \sim \varphi$ to indicate that $\mathfrak{x}$ is distributed according to φ, i.e., $\mathbb{P}(\mathfrak{x} \in J) = \int_J \varphi$. We will also denote by $\mathcal{B}$ the set of Borel subsets of $\mathbb{R}$.

Organization of the paper

In §2 we define the total variation distance of two random variables and introduce the notion of an efficient sampler. In §3 we give bounds on the total variation distance. Inverse transform sampling is illustrated in §4; we also describe how the univariate solving (that inverse transform sampling requires) is done using the bisection method. In §5 we present our curve sampler, study its efficiency and analyze its complexity. We also include a review of Chebyshev approximations. In the end, we present our experimental results in §6 and our conclusions in §7.

2 HOW GOOD IS A SAMPLING METHOD?

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

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 $\mathfrak{x}, \tilde{\mathfrak{x}} \in \mathbb{R}$ be random variables. The *total variation distance* (*TV distance*) of $\mathfrak{x}$ and $\tilde{\mathfrak{x}}$ is defined as

$$\mathrm{dist}_{\mathrm{TV}}(\mathfrak{x}, \tilde{\mathfrak{x}}) := \sup_{B\in\mathcal{B}} |\mathbb{P}(\mathfrak{x} \in B) - \mathbb{P}(\tilde{\mathfrak{x}} \in B)|,$$

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 polylogarithmic 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 $\mathfrak{x} \in \mathbb{R}$, an *efficient sampler for* $\mathfrak{x}$ is a pair of BSS programs $\mathrm{S} : \mathbb{N} \times \mathbb{R}^k \times [0,1]^l \to \mathbb{R}$ and $\mathrm{P} : \mathbb{N} \to \mathbb{R}^k$ such that: S1) on input (ℓ, x, u), the run-time of S is at

most poly(ℓ), S2) on input ℓ, the run-time of P is at most exp(ℓ), and S3) if $\mathfrak{u} \in [0,1]^l$ is uniformly distributed, then $\mathfrak{x}_\ell := \mathrm{S}(\ell, \mathrm{P}(\ell), \mathfrak{u})$ is a random variable such that

$$\mathrm{dist}_{\mathrm{TV}}(\mathfrak{x}_\ell, \mathfrak{x}) \leq 2^{-\ell}.$$

The program P in Def. 2.2 represents the preprocessing that the sampler requires in order to produce the correct result and $\mathfrak{u}$ the source of randomness. The output of P is used together with the source of randomness $\mathfrak{u}$ 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 $\tilde{\mathfrak{x}}$ is discrete, and if this is the case, since the target random variable $\mathfrak{x}$ is continuous, then

$$\mathrm{dist}_{\mathrm{TV}}(\mathfrak{x}, \tilde{\mathfrak{x}}) = 1.$$

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 $\tilde{\mathfrak{x}}$ into a continuous random variable by adding random noise to the values $\tilde{x_1}, \dots, \tilde{x_a}$ that $\tilde{\mathfrak{x}}$. To do this, we sample

$$x_i + \mathfrak{y}_i,$$

with probability $\mathbb{P}(\tilde{\mathfrak{x}} = \tilde{x_i})$, where $\mathfrak{y}_1, \dots, \mathfrak{y}_a$ some set of continuous random variables that we can sample. The resulting random variable will be continuous with density

$$\sum_{i=1}^{a} \mathbb{P}(\tilde{\mathfrak{x}} = \tilde{x_i}) \delta_{\mathfrak{y}_i}(t - \tilde{x_i}).$$

Hence, if the $\mathfrak{y}_i$ 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 $\mathfrak{x}, \mathfrak{y} \in I$, 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 $\delta_{\mathfrak{x}}$ and $\delta_{\mathfrak{y}}$ to denote the density functions of $\mathfrak{x}$ and $\mathfrak{y}$ 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 $\mathfrak{x}, \mathfrak{y} \in \mathbb{R}$ be continuous random variables. Then*

$$\mathrm{dist}_{\mathrm{TV}}(\mathfrak{x}, \mathfrak{y}) \leq \left\| \delta_{\mathfrak{x}} - \delta_{\mathfrak{y}} \right\|_1. \tag{3.1}$$

Proof. Let $B \in \mathcal{B}$. By definition, $\mathbb{P}(\mathfrak{x} \in B) = \int_B \delta_{\mathfrak{x}}$ and $\mathbb{P}(\mathfrak{y} \in B) = \int_B \delta_{\mathfrak{y}}$. Therefore, $|\mathbb{P}(\mathfrak{x} \in B) - \mathbb{P}(\mathfrak{y} \in B)| = \left| \int_B (\delta_{\mathfrak{x}} - \delta_{\mathfrak{y}}) \right| \leq \left\| \delta_{\mathfrak{x}} - \delta_{\mathfrak{y}} \right\|_1$. □

We recall that a usual way to bound the L^1-norm is to use the L^∞-norm. For a function $h : I \subset \mathbb{R} \to \mathbb{R}$, we have that

$$\|h\|_1 \leq \lambda(I) \|h\|_\infty \tag{3.2}$$

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 $\mathfrak{x} \in I$ and $\mathfrak{y} \in I$ using the information on how these random variables behave on a certain partition $\{J_i\}_{i=1}^k$ of I.

Definition 3.2. Let $\mathfrak{x} \in I$ be a continuous random variable and $J \subseteq I$. The *restriction of $\mathfrak{x}$ to J*, $\mathfrak{x}_{|J}$, is the random variable whose density is given by $(\delta_{\mathfrak{x}})_{|J} / \mathbb{P}(\mathfrak{x} \in J)$, if $\mathbb{P}(\mathfrak{x} \in J) \neq 0$, and by $1/\lambda(J)$, where λ is the Lebesgue's measure, otherwise.

One can see that to sample $\mathfrak{x}_{|J}$ using $\mathfrak{x}$ we only have to sample $\mathfrak{x}$ until it lies on J and, when this happens, output the sampled value. Note that this method requires on average $\mathbb{P}(\mathfrak{x} \in J)^{-1}$ attempts, so the larger $\mathbb{P}(\mathfrak{x} \in J)$ 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 $\mathfrak{x}, \mathfrak{y} \in I$ be a random variables and $\{J_i\}_{i=1}^k$ be a partition of I. Then*

$$\mathrm{dist}_{\mathrm{TV}}(\mathfrak{x}, \mathfrak{y}) \leq \sum_{i=1}^{k} \mathbb{P}(\mathfrak{x} \in J_i)\, \mathrm{dist}_{\mathrm{TV}}(\mathfrak{x}_{|J_i}, \mathfrak{y}_{|J_i}) + \sum_{i=1}^{k} |\mathbb{P}(\mathfrak{x} \in J_i) - \mathbb{P}(\mathfrak{y} \in J_i)| \tag{3.3}$$

Proof. Assume, without loss of generality that for all i, $\mathbb{P}(\mathfrak{x} \in J_i)$ and $\mathbb{P}(\mathfrak{y} \in J_i)$ are positive. Otherwise, the statement still holds, but the proof is slightly more convoluted. Fix $B \in \mathcal{B}$. By the definition of the $\mathfrak{x}_{|J_i}$, $\mathbb{P}(\mathfrak{x} \in B \cap J_i \mid \mathfrak{x} \in J_i) = \mathbb{P}(\mathfrak{x}_{|J_i} \in B)$. Therefore

$$\mathbb{P}(\mathfrak{x} \in B) = \sum_{i=1}^{k} \mathbb{P}(\mathfrak{x}_{|J_i} \in B) \mathbb{P}(\mathfrak{x} \in J_i), \tag{3.4}$$

and so, after some elementary operations, we bound $|\mathbb{P}(\mathfrak{x} \in B) - \mathbb{P}(\mathfrak{y} \in B)|$ by

$$\sum_{i=1}^{k} |\mathbb{P}(\mathfrak{x}_{|J_i} \in B) - \mathbb{P}(\mathfrak{y}_{|J_i} \in B)| \mathbb{P}(\mathfrak{x} \in J_i) + \sum_{i=1}^{k} |\mathbb{P}(\mathfrak{x} \in J_i) - \mathbb{P}(\mathfrak{y} \in J_i)|.$$

Now, by maximizing over $B \in \mathcal{B}$, we conclude. □

The above proposition suggests the strategy of partition sampling (Algorithm PartitionSampler). In other words, to sample $\mathfrak{x}$, we only need to sample the $\mathfrak{x}_{|J_i}$ and to compute the probabilities $\mathbb{P}(\mathfrak{x} \in J_i)$ with enough precision for some partition $\{J_i\}_{i=1}^k$ of I.

Algorithm 1: PartitionSampler

Input : Partition $J_1, \dots, J_k$ of I
$\tilde{p} \in \Delta^{k-1} := \{p \in \mathbb{R}^k_{\geq} 0 \mid \|p\|_1 = 1\}$
Approximate samplers S_i for $\mathfrak{x}_{|J_i}$
Output: Approximate sample of $\mathfrak{x} \in I$

Sample $i \in \{1, \dots, k\}$ with probability $\tilde{p}_i$
$\mathfrak{x} \leftarrow \mathrm{S}_i$ /* We use sampler S_i to get random $\mathfrak{x} \in J_i$ */
Output $\mathfrak{x}$

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 $\mathbb{P}(\mathfrak{x} \in J_i)$, we can choose sufficiently good values for $\tilde{p}$ 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 $\tilde{\mathfrak{x}} \in I$ of $\int_a^{\tilde{\mathfrak{x}}} \varphi(s)\, \mathrm{d}s = \mathfrak{u}$, for $\mathfrak{u} \in [0, 1]$ uniformly distributed, has density φ (see Alg.2 for its pseudocode).

Algorithm 2: InverseTransformSampler

Input : $I = [a, b]$, $\varphi : I \to [0, \infty)$ such that $\int_I \varphi(t)\, \mathrm{d}t = 1$
Output: $\mathfrak{x} \sim \varphi$

1 Sample $\mathfrak{u} \in [0, 1]$ uniformly
2 Find the solution $\mathfrak{x}$ of $\int_a^{\mathfrak{x}} \varphi(s)\, \mathrm{d}s = \mathfrak{u}$
3 Output $\mathfrak{x}$

However, any reader of this pseudocode will be suspicious about all the details swept under the rug in line 2. How do we solve $\int_a^{\mathfrak{x}} \varphi(s)\, \mathrm{d}s = \mathfrak{u}$? 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 φ be

$$\Phi(x) := \int_a^x \varphi(s)\, \mathrm{d}s. \tag{4.1}$$

We want to solve the equation

$$\Phi(x) = u \in [0, 1]. \tag{4.2}$$

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 ℓ 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 BisectionSampler.

Algorithm 3: BisectionSampler

Input : $\varphi : [a, b] \to [0, \infty)$ such that $\int_0^1 \varphi(t)\, \mathrm{d}t = 1$
$\ell \in \mathbb{N}$
Output: Approximate sample $\mathfrak{x}$ for φ

Sample $\mathfrak{u} \in [0, 1]$ uniformly, $i \leftarrow 0$
$x_l \leftarrow a$, $v_l \leftarrow \mathrm{sign}\,(\mathfrak{u} - \Phi(x_l))$, $x_r \leftarrow b$, $v_r \leftarrow \mathrm{sign}\,(\mathfrak{u} - \Phi(x_r))$
while $i < \ell$ **do**
  $x_m \leftarrow (x_l + x_r)/2$, $v_m \leftarrow \mathrm{sign}\,(\mathfrak{u} - \Phi(x_m))$
  if $v_l = v_m$ **then**
    $x_r \leftarrow x_m$, $v_r \leftarrow v_m$, $i \leftarrow i + 1$
  else
    $x_l \leftarrow x_m$, $v_l \leftarrow v_m$, $i \leftarrow i + 1$
Sample $\tilde{\mathfrak{x}} \in [x_l, x_r]$ uniformly
Output $\tilde{\mathfrak{x}}$

The following theorem shows that BisectionSampler produces a nice sampler.

Theorem 4.1. *Let $\varphi : [a, b] \to [0, \infty)$, $\mathfrak{x} \sim \varphi$ and $\tilde{\mathfrak{x}}_\ell$ the output of BisectionSampler for $\ell \in \mathbb{N}$. Then*

$$\mathrm{dist}_{\mathrm{TV}}(\mathfrak{x}, \tilde{\mathfrak{x}}_\ell) \leq 2^{-\ell} |b - a| \max_{x \in [a,b]} |\varphi'(x)|. \tag{4.3}$$

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

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

Therefore, by Proposition 3.3, to compute $\mathrm{dist}_{\mathrm{TV}}(\mathfrak{x}, \tilde{\mathfrak{x}}_\ell)$, we only have compute the TV distance between $\mathfrak{x}_{|J_k}$ and the uniformly distributed $\mathfrak{u}_k \in J_k$. Now, after an elementary computation and Proposition 3.1,

$$\mathrm{dist}_{\mathrm{TV}}(\mathfrak{x}_{|J_k}, \mathfrak{u}_k) \leq \mathbb{P}(\mathfrak{x} \in J_k)^{-1} \|\varphi_{|J_k} - 2^\ell \mathbb{P}(\mathfrak{x} \in J_k)\|_1. \tag{4.4}$$

Now, by the mean value theorem, $\mathbb{P}(\mathfrak{x} \in J_k) = b\varphi(\xi_k)/2^\ell$ for some $\xi_k \in J_k$, so we obtain

$$\mathrm{dist}_{\mathrm{TV}}(\mathfrak{x}_{|J_k}, \mathfrak{u}_k) \leq \mathbb{P}(\mathfrak{x} \in J_k)^{-1} \|\varphi_{|J_k} - \varphi(\xi_k)\|_1. \tag{4.5}$$

Now, $|\varphi(s) - \varphi(\xi_k)| \leq b2^{-\ell} \max_{x \in J_k} |\varphi'(x)|$, by the mean value theorem, and so $\mathrm{dist}_{\mathrm{TV}}(\mathfrak{x}_{|J_k}, \mathfrak{u}_k) \leq \mathbb{P}(\mathfrak{x} \in J_k)^{-1} b 2^{-2\ell} \max_{x \in J_k} |\varphi'(x)|$, concluding the proof. □

Note that 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 $\mathfrak{x} \in \gamma(I)$ uniformly with respect to the arc-length, we only need to sample a random parameter $\mathfrak{t} \in I$ distributed according to the normalized speed

$$\varphi(t) := \left(\int_{-1}^{1} \|\gamma'(s)\|_2 \mathrm{d}s\right)^{-1} \|\gamma'(t)\|_2 \tag{5.1}$$

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

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

$$\tilde{\Phi}(t) := \int_{-1}^{t} \tilde{\varphi}(s)\, \mathrm{d}s$$

is a lot easier than computing

$$\Phi(t) := \int_{-1}^{t} \varphi(s)\, \mathrm{d}s.$$

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 *kth Chebyshev polynomial* is the polynomial given by

$$\text{Ч}_k(X) = \sum_{i=0}^{k} \binom{k}{2i} (1 - X^2)^i X^{k-2i}, \tag{5.2}$$

where Ч is the first initial of Chebyshev in the Cyrillic script. Alternatively, note that Ч_k satisfies

$$\text{Ч}_k(x) = \cos(k \arccos(x)). \tag{5.3}$$

for $x \in I$. As a consequence, the k zeros of Ч_k are given by

$$\zeta_{a,k} := \cos\left(\frac{(1 + 2a)\pi}{2k}\right) \tag{5.4}$$

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

5.1.1 Chebyshev interpolation. The *kth Chebyshev interpolant* of $f : I \to \mathbb{R}$ is the unique degree k polynomial $\text{Ч}I_k(f)$ satisfying for $a \in \{0, \ldots, k\}$,

$$\text{Ч}I_k(f)(\zeta_{a,k+1}) = f(\zeta_{a,k+1}). \tag{5.5}$$

To compute $\text{Ч}I_k(f)$, there is no need to solve the system above thanks to the following proposition.

Proposition 5.1. *[19, Thm. 6.7] Let*

$$\text{Ч}I_k(f) = \frac{c_0}{2} + \sum_{a=1}^{k} c_a \text{Ч}_a$$

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

$$c_a = \frac{2}{k+1} \sum_{i=0}^{k} f(\zeta_{i,k}) \text{Ч}_a(\zeta_{i,k}). \qquad \square$$

Let us remind that $\text{Ч}I_k(f)$ 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 $\text{Ч}I(f)$, 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 $p = \sum_{a=0}^{k} c_a \text{Ч}_k$, then for every $x \in \mathbb{R}$,*

$$p(x) = \frac{1}{2}(\text{Б}_0(x) - \text{Б}_2(x)) \tag{5.6}$$

where $\text{Б}_0(x)$ and $\text{Б}_2(x)$ are computed through the following backwards-recursive relation

$$\begin{cases} \text{Б}_{k+1}(x) &= \text{Б}_{k+2}(x) = 0 \\ \text{Б}_a(x) &= 2x\text{Б}_{a+1}(x) - \text{Б}_{a+2}(x) + c_a. \end{cases} \qquad \square$$

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_ρ 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_ρ, then*

$$\|f - \text{Ч}I_k(f)\|_\infty \leq \frac{4\|f\|_{E_\rho} \rho^{-k}}{\rho - 1}, \tag{5.7}$$

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

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*

$$\|f\|_{E_\rho} \leq \rho^d \|f\|_\infty. \qquad \square$$

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 φ, i.e., the normalized speed of γ (5.1). The hard part will be estimating a sufficiently small ρ so that φ admits an analytic extension to the interior of E_ρ.

5.1.4 Integration formulas. Imagine we want to compute the integral (definite or indefinite) of $\text{Ч}I_k(f)$. To do this, we use the following proposition.

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

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

Then

$$\sum_{a=1}^{k+1} pc_a \text{Ч}_a$$

with

$$pc_a = \begin{cases} \frac{c_{a-1}-c_{a+1}}{2a}, & \text{if } a = 1, \dots, k-1 \\ \frac{c_{k-1}}{2k}, & \text{if } a = k \\ \frac{c_k}{2(k+1)}, & \text{if } a = k+1 \end{cases}$$

is a primitive function of p.

Moreover,

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

The following fact will be useful later. Let $F(t) := \int_{-1}^{t} f(s)\mathrm{d}s$ 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 φ and gives possible ρs we can take.

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

$$\rho^*(\gamma) := \min_{\substack{z \in \mathbb{C} \\ \|\gamma'(z)\|_2 = 0}} \left\{ \frac{|z+1| + |z-1| + \sqrt{(|z+1| + |z-1|)^2 - 4}}{2} \right\},$$

then φ admits an analytic extension φ_{an} to the interior of E_ρ and

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

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

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

Recall that E_ρ 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 ρ.

Since φ_{an}^2 is a polynomial of degree $2d$ on I, it is a polynomial of degree $2d$ on the interior of E_ρ. Hence, by Bernstein's inequality (Theorem 5.4),

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

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

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 $\text{Ч}I_k(\varphi)$ such that $\|\text{Ч}I_k(\varphi) - \varphi\|_\infty \leq \varepsilon$, then the degree of this interpolant has to satisfy

$$k \geq \frac{1}{\log \rho^*(\gamma)} \left(\ln \frac{1}{\varepsilon} + \log \|\varphi\|_{E_{\rho^*(\gamma)}} + 2 - \log(\rho^*(\gamma) - 1) \right). \quad (5.8)$$

by Theorem 5.3.

We now give theoretical bounds for ρ^*. 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 ith component, γ_i, is given by

$$\gamma_i = \sum_j \gamma_{i,j} T^j.$$

Then the *condition number for sampling* γ is quantity

$$\mathtt{C}(\gamma) := \frac{\|\gamma'\|_o}{\inf_{t \in I} \|\gamma'(t)\|_2} \in [1, \infty] \quad (5.9)$$

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

Remark 5.9. The condition number for sampling γ, $\mathtt{C}(\gamma)$, is finite as long as φ is non-vanishing in I. Note that the idea is that the nearer is γ 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*

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

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

$$I_\varepsilon \ni z \mapsto |\gamma_i'(z)|/\|\gamma_i'\|_o$$

is $(\mathrm{e} \cdot d)$-Lipschitz. Hence, the map

$$I_\varepsilon \ni z \mapsto \|\gamma'(z)\|_2/\|\gamma'\|_o.$$

is $(\mathrm{e} \cdot d)$-Lipschitz. In this way, if $\varepsilon = 1/(\mathrm{e} \cdot d \cdot \mathtt{C}(\gamma))$, we have that $\|\gamma'\|^2$ does not have zeros inside I_ε. Now, if for some $\rho \geq 1$, $E_\rho \subseteq I_\varepsilon$, then $\rho \leq \rho^*(\gamma)$.

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

Now, $\rho = 1 + \varepsilon$, with $\varepsilon = 1/(\mathrm{e} \cdot d \cdot \mathtt{C}(\gamma)) \leq 1$, satisfies these inequalities. Hence the claim follows. $\square$

5.3 Acceleration through splitting the curve

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

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 6.3 and Figure 3).

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

$$\Re z_1, \dots, \Re z_d$$

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

5.4 The sampler

We give the sampler for the curve $\gamma : I \to \mathbb{R}^n$, 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: $\gamma : I \to \mathbb{R}^n$ of degree d
$\ell \in \mathbb{N}$
Output: Approximate sample $\mathfrak{t}$ of
$$\varphi := \|\gamma'\|_2 / \int_{-1}^{1} \|\gamma'\|_2(s)\mathrm{d}s$$

1 $\varphi \leftarrow \|\gamma'\|_2 / \int_{-1}^{1} \|\gamma'\|_2(s)\mathrm{d}s$ /* off-line */
2 $Z \leftarrow \{z \in \mathbb{C} \mid \gamma'(z) = 0\}$ /* off-line */
3 $\rho^* \leftarrow \min_{z \in Z} \frac{|z+1|+|z-1|+\sqrt{(|z+1|+|z-1|)^2-4}}{2}$ /* off-line */
4 $M \leftarrow \|\varphi\|_{E_{\rho^*}}$ /* off-line */
5 $k \leftarrow 5 + \ell + \lceil (\log M - \log(\rho^* - 1))/\log \rho^* \rceil$ /* off-line */
6 $\tilde{\varphi} \leftarrow \Psi I_k(\varphi) / \int_{-1}^{1} \Psi I_k(\varphi)(s)\mathrm{d}s$ /* off-line */
7 $\tilde{\Phi} \leftarrow$Integral of $\Psi I_k(\varphi)$ /* off-line */
8 $\ell_B \leftarrow 1 + \ell + \max\{0, \log \|\tilde{\varphi}'\|_\infty\}$ /* off-line */
9 $\mathfrak{t} \leftarrow$BisectionSampler$(\tilde{\varphi}, \ell_B)$
10 Output $\mathfrak{t}$

5.5 Complexity of CurveSampler

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

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

$$O(\ell^2(1 + \log \mathsf{C}(\gamma))^2 \mathsf{C}(\gamma)^2)$$

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

$$O(\ell^3(1 + \log d\mathsf{C}(\gamma))^3 d^3 \mathsf{C}(\gamma)^3)$$

on-line arithmetic operations to achieve an error of $2^{-\ell}$ 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,

$$\|\tilde{\varphi} - \varphi\|_1 \le 2\|\Psi I_k(\varphi) - \varphi\|_1 \le \frac{16M}{\rho^* - 1}(\rho^*)^{-k} \le 2^{-(1+\ell)} \quad (5.10)$$

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

$$\mathrm{dist}_{\mathrm{TV}}(\mathfrak{s}, \tilde{\mathfrak{t}}) \le 2^{-(1+\ell)} \quad (5.11)$$

where $\mathfrak{t} \sim \tilde{\varphi}$. On the other hand, by Theorem 4.1,

$$\mathrm{dist}_{\mathrm{TV}}(\tilde{\mathfrak{t}}, \mathfrak{t}) \le 2^{1-\ell_B}\|\tilde{\varphi}'\|_\infty \le 2^{-(1+\ell)}.$$

Hence,

$$\mathrm{dist}_{\mathrm{TV}}(\mathfrak{s}, \mathfrak{t}) \le 2^{-\ell}$$

and so to show that CurveSampler is an efficient sampler for φ, 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 ℓ. 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_{ρ^*} 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

$$O(\ell^2(1 + \log d\mathsf{C}(\gamma))^2 d^2 \mathsf{C}(\gamma)^2)$$

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

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

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

$$\|\Psi I_k(\varphi)' - \varphi'\|_\infty \le 16Mk^2(\rho^*)^{3-k}/((\rho^*) - 1).$$

Thus,

$$\|\tilde{\varphi}'\|_\infty \le \|\varphi'\|_\infty + \frac{16Mk^2\rho^{3-k}}{\rho - 1} \le \|\varphi'\|_\infty + 4k^2(\rho^*)^{2-\ell}.$$

Now, on the one hand,

$$4k^2(\rho^*)^{2-\ell} \le O(\ell^2(1 + \log d\mathsf{C}(\gamma))^2 d^2 \mathsf{C}(\gamma)^2);$$

and on the other hand,

$$|\varphi'| = \langle \gamma', \gamma'' \rangle / \|\gamma'\|_2^2 \le \|\gamma''\|_2 / \|\gamma'\|_2$$

where $\|\gamma''\|_2 \le d\|\gamma'\|_o$ by [27, Proposition 3.6]. Thus $\|\varphi'\|_\infty \le d\mathsf{C}(\gamma)$. Putting this together, we obtain, the bound $\ell_B \le O(\ell + \log d\mathsf{C}(\gamma) + \ell^2(1 + \log d\mathsf{C}(\gamma))^2 d^2 \mathsf{C}(\gamma)^2)$. 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[1], 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 $\text{Ч}_k(x)$, (ii) `roots()` to compute the zero set of the speed $\|\gamma'(t)\|_2^2$ for the computation of ρ^* (line 3 in CurveSampler), and (iii) `fmincon()` to solve the optimization problem required to compute $\|f\|_{E_\rho}$ (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 $\gamma : [-1, 1] \ni t \mapsto (3t^2 - 2t, 2t^2)$,

$$\|\gamma'(t)\|_2 = \sqrt{42t^2 - 24t + 4},$$

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

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 $\gamma : [-1, 1] \to \mathbb{R}^n$ of degree $d \in \{5, 10, 15, 20\}$ with $n \in \{20, 40, 60, 80, 100\}$ components and with error specifications $2^\ell \in \{10, 100\}$. The coefficients of the polynomials in γ are independent, identically distributed standard Gaussian random variables. In the table, we display the degree of the Chebyshev interpolant (k), the preprocess 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 $\gamma : [-1, 1] \to \mathbb{R}^{50}$ 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^ℓ.

In Figure 2, we simply show the evolution of the degree. In Figure 5.3, 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.

[1]For example, we allow 2^ℓ to be any real number and not only a power of 2.

6.4 A polynomial density: Figures 4 and 5

We consider the curve $\gamma : [-1, 1] \ni t \mapsto (1 + T + \cdots + T^d)(1, 1, 1) \in \mathbb{R}^3$ 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 $\mathsf{C}(\gamma)$ 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 examples, 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 $\mathsf{C}(\gamma)$, introduced in Definition 5.8, and its analysis in terms of the bit-size and for a random γ. (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 improved the quality of this paper.

		$2^{-\ell} = 0.1$			$2^{-\ell} = 0.01$		
d	n	k	$Pr.\ T$	T/s	k	$Pr.\ T$	T/s
	20	20	0.82	0.53	24	0.94	0.58
	40	32	1.08	0.44	39	1.28	0.47
5	60	27	0.92	0.44	32	1.09	0.48
	80	29	1.04	0.43	34	1.18	0.45
	100	25	1.24	0.54	29	1.30	0.59
	20	34	1.34	0.57	40	1.51	0.60
	40	21	0.76	0.40	27	0.89	0.42
10	60	35	1.17	0.44	41	1.37	0.47
	80	38	1.31	0.53	44	1.65	0.56
	100	37	1.64	0.64	43	2.03	0.65
	20	35	1.29	0.59	41	1.70	0.61
	40	46	1.53	0.48	53	1.70	0.50
15	60	47	3.40	0.49	54	3.54	0.50
	80	43	1.65	0.49	50	1.90	0.53
	100	49	2.36	0.63	55	2.36	0.65
	20	67	2.82	0.70	79	4.41	0.82
	40	63	2.07	0.53	72	2.65	0.57
20	60	51	2.38	0.66	58	2.24	0.56
	80	70	2.60	0.59	80	4.38	0.76
	100	56	2.55	0.62	63	3.24	0.65

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^{ℓ})

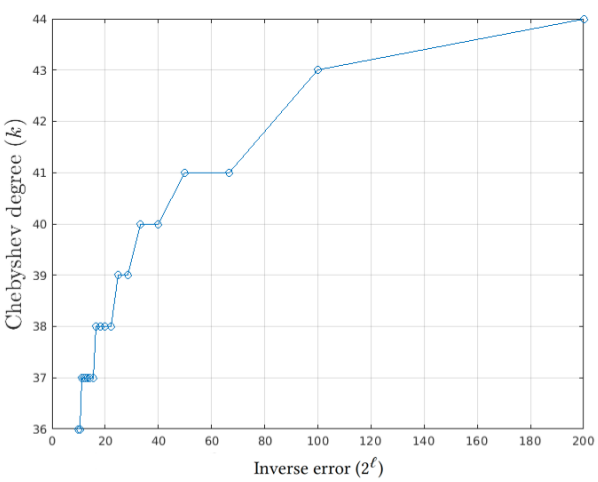

Figure 2: Degree of the Chebyshev interpolant (k) against the inverse error (2^{ℓ}) for a curve of degree 10 in $\mathbb{R}^{50}$.

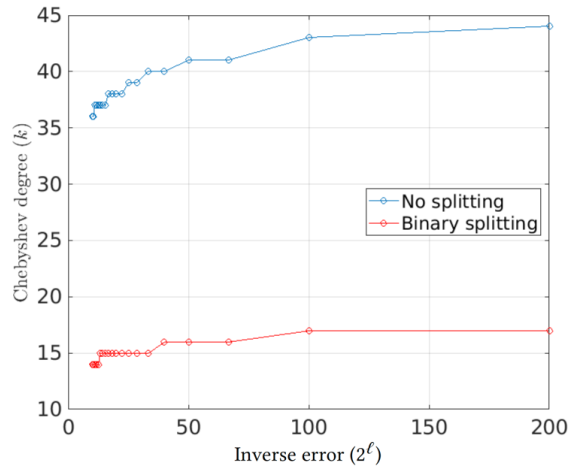

Figure 3: Maximum degree of the Chebyshev interpolant (k) against the inverse error (2^{ℓ}) for a curve of degree 10 in $\mathbb{R}^{50}$ without splitting the interval (blue) and splitting the interval in four equal intervals (red)

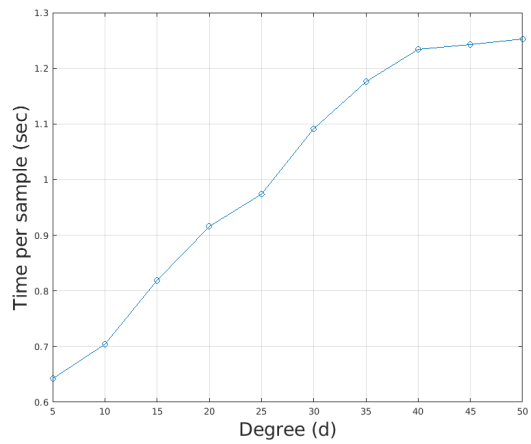

Figure 4: Time per sample after preprocessing against the degree (d) for the 3D curve with each coordinate being the degree d polynomial $1 + T + \cdots + T^d$.

REFERENCES

[1] Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. 1998. *Complexity and real computation.* Springer-Verlag, New York. xvi+453 pages. DOI: http://dx.doi.org/10.1007/978-1-4612-0701-6 With a foreword by Richard M. Karp.

[2] Paul Breiding, Sara Kališnik, Bernd Sturmfels, and Madeleine Weinstein. 2018. Learning algebraic varieties from samples. *Rev. Mat. Complut.* 31, 3 (2018), 545–593. DOI: http://dx.doi.org/10.1007/s13163-018-0273-6

[3] Zongchen Chen and Santosh S. Vempala. 2019. Optimal Convergence Rate of Hamiltonian Monte Carlo for Strongly Logconcave Distributions. In *APPROX/RANDOM 2019 (LIPIcs)*, Dimitris Achlioptas and László A. Végh (Eds.), Vol. 145. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Wadern, Germany, 64:1–64:12. DOI: http://dx.doi.org/10.4230/LIPIcs.APPROX-RANDOM.2019.64

[4] Siddhartha Chib and Edward Greenberg. 1995. Understanding the metropolis-hastings algorithm. *The american statistician* 49, 4 (1995), 327–335. DOI: http://dx.doi.org/10.1080/00031305.1995.10476177

[5] Raaz Dwivedi, Yuansi Chen, Martin J Wainwright, and Bin Yu. 2019. Log-concave sampling: Metropolis-Hastings algorithms are fast. *Journal of Machine Learning Research* 20, 183 (2019), 1–42.

[6] Luiz Henrique de Figueiredo. 1995. IV.4 - Adaptive Sampling of Parametric Curves. In *Graphics Gems V*, Alan W. Paeth (Ed.). Academic Press, Boston, 173–178. DOI:

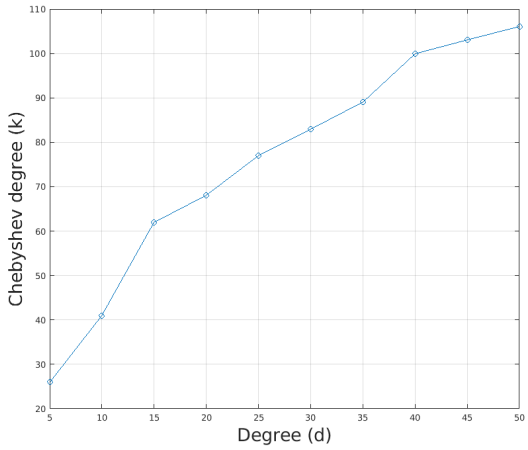

Figure 5: Degree of the Chebyshev interpolant (k) against the degree (d) for the 3D curve with each coordinate being the degree d polynomial $1 + T + \cdots + T^d$.

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

[7] Michael S. Floater and Atgeirr F. Rasmussen. 2006. Point-based methods for estimating the length of a parametric curve. *J. Comput. Appl. Math.* 196, 2 (2006), 512–522. DOI: http://dx.doi.org/10.1016/j.cam.2005.10.001

[8] Michael S. Floater, Atgeirr F. Rasmussen, and Ulrich Reif. 2007. Extrapolation methods for approximating arc length and surface area. *Numer. Algorithms* 44, 3 (2007), 235–248. DOI: http://dx.doi.org/10.1007/s11075-007-9095-1

[9] L. Fox and I. B. Parker. 1968. *Chebyshev polynomials in numerical analysis.* Oxford University Press, London-New York-Toronto, Ont. ix+205 pages.

[10] W. R. Gilks and P. Wild. 1992. Adaptive Rejection Sampling for Gibbs Sampling. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 41, 2 (1992), 337–348. http://www.jstor.org/stable/2347565

[11] Jens Gravesen. 1997. Adaptive subdivision and the length and energy of Bézier curves. *Comput. Geom.* 8, 1 (1997), 13–31. DOI: http://dx.doi.org/10.1016/0925-7721(95)00054-2

[12] James Johndrow and Aaron Smith. 2018. Fast mixing of metropolis-hastings with unimodal targets. *Electronic Communications in Probability* 23, none (2018), 1 – 9. DOI: http://dx.doi.org/10.1214/18-ECP170

[13] Ravi Kannan and Santosh Vempala. 1997. Sampling Lattice Points. (1997). DOI: http://dx.doi.org/10.1145/258533.258665

[14] Yin Tat Lee, Zhao Song, and Santosh S Vempala. 2018. Algorithmic theory of ODEs and sampling from well-conditioned log-concave densities. (2018). arXiv:1812.06243.

[15] Josef Leydold. 1998. A Rejection Technique for Sampling from Log-Concave Multivariate Distributions. *ACM Trans. Model. Comput. Simul.* 8, 3 (jul 1998), 254–280. DOI: http://dx.doi.org/10.1145/290274.290287

[16] Laszlo Lovasz and Santosh Vempala. 2006. Fast Algorithms for Logconcave Functions: Sampling, Rounding, Integration and Optimization. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06).* IEEE, Berkeley, CA, USA, 57–68. DOI: http://dx.doi.org/10.1109/FOCS.2006.28

[17] László Lovász and Santosh Vempala. 2006. Hit-and-run from a corner. *SIAM J. Comput.* 35, 4 (2006), 985–1005.

[18] Oren Mangoubi and Nisheeth K Vishnoi. 2019. Nonconvex sampling with the Metropolis-adjusted Langevin algorithm. In *Proceedings of the Thirty-Second Conference on Learning Theory (Proceedings of Machine Learning Research)*, Alina Beygelzimer and Daniel Hsu (Eds.), Vol. 99. PMLR, Phoenix, USA, 2259–2293. https://proceedings.mlr.press/v99/mangoubi19a.html

[19] J. C. Mason and D. C. Handscomb. 2003. *Chebyshev polynomials.* Chapman & Hall/CRC, Boca Raton, FL. xiv+341 pages.

[20] Radford M. Neal. 2003. Slice sampling. *Ann. Statist.* 31, 3 (06 2003), 705–767. DOI: http://dx.doi.org/10.1214/aos/1056562461

[21] Radford M. Neal. 2011. MCMC using Hamiltonian dynamics. In *Handbook of Markov chain Monte Carlo.* CRC Press, Boca Raton, FL, USA, 113–162.

[22] Partha Niyogi, Stephen Smale, and Shmuel Weinberger. 2008. Finding the homology of submanifolds with high confidence from random samples. *Discrete Comput. Geom.* 39, 1-3 (2008), 419–441. DOI: http://dx.doi.org/10.1007/s00454-008-9053-2

[23] Sheehan Olver and Alex Townsend. 2013. Fast inverse transform sampling in one and two dimensions. (2013). arXiv:1307.1223

[24] Luca Pagani and Paul J. Scott. 2018. Curvature based sampling of curves and surfaces. *Computer Aided Geometric Design* 59 (2018), 32–48. DOI: http://dx.doi.org/10.1016/j.cagd.2017.11.004

[25] Les Piegl and Wayne Tiller. 1997. *The NURBS Book (2nd Ed.).* Springer-Verlag, Berlin, Heidelberg.

[26] R.L. Smith. 1996. The hit-and-run sampler: a globally reaching markov chain sampler for generating arbitrary multivariate distributions. In *Proceedings Winter Simulation Conference.* IEEE, Coronado, CA, USA, 260–264. DOI: http://dx.doi.org/10.1109/WSC.1996.873287

[27] J. Tonelli-Cueto and E. Tsigaridas. 2021. Condition Numbers for the Cube. I: Univariate Polynomials and Hypersurfaces. (2021). To appear in the special issue of the Journal of Symbolic Computation for ISSAC 2020. Available at arXiv:2006.04423.

[28] Lloyd N. Trefethen. 2013. *Approximation theory and approximation practice.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA. viii+305 pp.+back matter pages.

[29] Santosh Vempala. 2005. Geometric random walks: a survey. *Combinatorial and computational geometry* 52, 573-612 (2005), 2.

[30] Marcelo Walter and Alain Fournier. 1996. Approximate arc length parameterization. In *Proceedings of the 9th Brazilian symposium on computer graphics and image processing.* Citeseer, Caxambu, Minas Gerais, Brazil, 143–150.