UPPSALA
UNIVERSITET

Summary Statistics-Based Sequential Sampling of Stochastic Time Series for Likelihood-free Parameter Inference

Niklas Kavathatzopoulos, Erik Turesson

Project in Computational Science: Report

January 2022

PROJECT REPORT

Department of Information Technology

Abstract

Likelihood-free parameter inference is a widely used method for the analysis of complex simulation models. Machine learning based inference techniques have previously involved training on random training sets. When one or more observations are to be inferred, we propose a sequential approach of generating training data sets for increased data efficiency to maximize information gain per additional sample by sampling heavily in regions where simulations are similar to the observation. Applied on complex and high-dimensional models this technique can speed up the Approximate Bayesian Computation process significantly. The technique is applied to three benchmark problems and tested on a 15-dimensional genetic oscillator. We show that the sampling technique proposed in this work effectively explores the parameter space for interesting regions, samples densely in the interesting regions, and improves inference accuracy for all test problems, including a 20-fold data efficiency improvement on one benchmark problem with a very small, hard to reach active region of interest.

Contents

1 Introduction

To understand, design, diagnose and test complex models and processes, parameter inference is an essential tool that estimates parameters from observed and measured data. When no probabilistic likelihood-function exists, or its calculation is unfeasible, likelihood-free parameter inference techniques may be required for any accurate parameter estimates in the place of traditional parameter inference approaches relying on a likelihood function. Instead, a simulation model of the problem is used to explore the parameters which are likely to produce similar observations. The highly popular rejection sampling-based likelihood-free parameter inference method, Approximate Bayesian Computation (ABC), samples possible parameters and accepts samples producing outputs similar to the observed data. Finally, it constructs a posterior probability distribution of which parameters are the most likely to create such an observation.

The performance and ease of use of ABC has been substantially enhanced by utilizing artificial neural networks to measure the distance between observations in the acceptance stage. However, these networks require large amounts of training data, especially for high-dimensional problems. Such training data sets can be expensive to generate for costly simulation models.

Conventionally, uniform random sampling is used to create this training data. Thus, data is generated in a one-shot approach from a domain without information about previously simulated samples. By utilizing information from earlier samples and sequentially generating additional training data points, we attempt to improve the data efficiency of the networks. Sequential sampling evaluates where it should generate new samples using information from previous simulations. This sampling method is effective when we expect certain regions of the parameter space to be more interesting than others. In this report, we attempt to implement sequential sampling techniques for training inference models of stochastic time series to ensure new samples are drawn in the most informative areas, in the hope of decreasing the number of samples needed for accurate inference.

Section 2 introduces likelihood-free parameter inference and the concept of summary statistics. Section 3 describes the sequential sampling methods used in this report. Then, Section 4 briefly covers some details of our implementations and experiments. Finally, Section 5 demonstrates the proposed sequential sampling method on several problems and presents the results.

2 Background

2.1 Likelihood-Free Parameter Inference

In parameter inference, we want to infer parameter values $\hat{\boldsymbol{\theta}}$ based on a model $f(\boldsymbol{\theta})$ and observed data X, meaning values of $\boldsymbol{\theta}$ for which $f(\boldsymbol{\theta})$ is similar to X. For simpler analytical models, there might exist a likelihood $P(X|\boldsymbol{\theta})$. For these cases, Bayesian inference or maximum-likelihood estimation [1] can be used to estimate $P(\hat{\boldsymbol{\theta}}|X)$. However, most complex problems would not have access to any tractable likelihood-function, meaning alternative solutions must be employed. Such methods are referred to as *likelihood-free*

inference methods. ABC, described in e.g. [2], is a widely used likelihood-free inference-method which bypasses the need of a likelihood and is applicable in a wide range of disciplines such as evolutionary science [3], ecology [4], astrophysics and cosmology [5].

ABC draws samples $\boldsymbol{\theta}$ from a prior $p(\boldsymbol{\theta})$ and accepts all samples whose realization $f(\boldsymbol{\theta})$'s distance to X is lower than some threshold τ according to some distance measure $d(f(\boldsymbol{\theta}), X)$, and rejects samples further away. New samples are evaluated until some condition, such as a number of accepted samples, is fulfilled. From all accepted samples, a posterior distribution $P(\hat{\boldsymbol{\theta}}|X)$ can be estimated.

This project focuses on problems in which the output of $f(\boldsymbol{\theta})$ is a simulated and stochastic time series. Each time step of the time series can be regarded as a dimension. Due to noise arising in high-dimensional scenarios, directly calculating $d(f(\boldsymbol{\theta}), X)$ can be problematic. This issue can be solved with *summary statistics*. Summary statistics are scalar measures (such as mean or variance) of a high-dimensional output (such as a time series). Summary statistics used in this work are further explained in Section 2.2. For a set of N summary statistics $\boldsymbol{stats}(f(\boldsymbol{\theta})) = \{stat_1(f(\boldsymbol{\theta})), \ldots, stat_N(f(\boldsymbol{\theta}))\}$, the distance is measured as $d(\boldsymbol{stats}(f(\boldsymbol{\theta})), \boldsymbol{stats}(X))$.

The choice of a set of informative summary statistics is essential for accurate inference, and the current state of the art utilizes artificial neural networks architectures to produce informative summary statistics as an alternative to traditional measurements [6, 7, 8]. These networks learn to infer the estimated posterior mean $\hat{\boldsymbol{\theta}} = E(P(\boldsymbol{\theta}|X))$ directly from observed time series. The estimated posterior mean has been shown to be the ideal summary statistic [9], hence it is used as an informative summary statistic in the ABC rejection sampling scheme. Such networks require large training data sets to produce accurate estimations, which can be expensive to generate for complex simulation models. The exact amount of observations in training data will vary on the problem, desired accuracy and model architecture. The authors of [7] use 10^6 training samples, whereas [6] observes that some 10^5 may be required for complex problems.

This project explores a sequential sampling technique to reduce the amount of training data needed by producing more informative samples. Thus, we limit the scope not to include the entire ABC scheme but to only look at how the inference performance of the convolutional neural network (CNN) architecture from [6] is affected by sequential sampling, as a more accurate network is assumed to lead to more accurate ABC.

2.2 Time Series Summary Statistics

Random perturbations or time-shifts can propagate throughout a stochastic time series, leading to large time-step-wise discrepancies even though the overall output is similar. Instead, ways of summarizing different qualities of a discrete time series $y(t)$ as a scalar are introduced. Such a measure is known as a summary statistic. The use of summary statistics in likelihood-free inference methods such as ABC is widespread since it serves as a dimensionality reduction for time series. The summary statistic approach reduces random discrepancies that can occur for a vast number of dimensions.

While numerous summary statistics exist, the following are used in this report:

Approximate Entropy Entropy measures can inform about the predictability or regularity of data [10]. Approximate entropy is introduced in [11] specifically for stochastic and noisy data, for which other entropy measures have poor performance.

Sample entropy Sample entropy is a modification of approximate entropy introduced in [12].

Spectral Edge Frequency The Spectral Edge Frequency is the frequency below which a threshold percentage x of the signals total power is contained. For a discrete time series $y(t)$ with a discrete time Fourier transform $\hat{y}(\omega)$, the power of the signal can be defined as $\sum_i |\hat{y}(\omega_i)|^2$. The 90% spectral edge frequency, ω_j, is then defined as where $\sum_{i \leq j} |\hat{y}(\omega_i)|^2 \geq 0.90 \sum_i |\hat{y}(\omega_i)|^2 > \sum_{i \leq j-1} |\hat{y}(\omega_i)|^2$. This metric allows for us to capture some oscillatory patterns of a signal.

Autocovariance The autocovarince is a measure of a processes covariance with itself at various time lags. For a time lag τ, the autocovariance of $y(t)$ with mean μ_y is defined as $K_{yy}(\tau) = E[y(t)y(t-\tau)] - \mu_y^2$ [13, p.392]. We use autocorrelation at τ of zero, one, and two discrete time steps as three separate summary statistics.

Hjorth Parameters The Hjorth parameters, introduced in [14], are *Activity*, *Mobility* and *Complexity*. Activity is the variance of the signal, $var\big(y(t)\big)$. Mobility, defined as $\sqrt{var(y'(t))/var(y(t))}$, can be interpreted as a relative average slope. Complexity is defined as $mobility(y'(t))/mobility(y(t))$.

Burstiness Burstiness measures how much intermittent frequencies of a signal vary. In [15] a burstiness score is calculated as the ratio $\frac{\sigma_y - \mu_y}{\sigma_y + \mu_y}$ where σ_y and μ_y is the standard deviation and mean of $y(t)$ respectively. Burstiness is distributed on $[-1, 1]$, where -1 corresponds to a linear or single frequency signal.

Furthermore, there exist approaches of deciding upon a subset of the most informative summary statistics, such as *approximate sufficiency* [16]. Although our sequential sampling design presented in Section 3.2 would potentially benefit from such, this is not explored further in this report.

For models where the simulation outputs multiple time series, we calculate each summary statistic separately for each series.

2.3 Convolutional Neural Networks

In contrast to a traditional dense neural network layer, a convolutional layer considers the structural properties of inputs. Neural networks utilizing a convolutional layer are known as convolutional neural networks (CNNs). This allows for increased capabilities in finding local patterns, such as oscillations or spikes, from a grid-like input. A uniformly spaced time series is essentially a one-dimensional grid that allows the use of such networks.

Replacing the matrix multiplication operation in a dense, fully-connected layer; a one-dimensional convolution operator is defined for a neuron $t \in 1, \ldots, T$ as

$$(y \ * \ w)(t) = \sum_{i=-M}^{M} y(t-i)w(t), \tag{1}$$

where y is the input and the trainable kernel w defines weights for each input neuron around t within the kernel width M.

As the learning models themselves are not the primary concern of this report, the interested reader is referred to [17, Ch.6] for an introduction to neural networks, or to [6] for the application of CNNs in the context of parameter inference for stochastic time series and the architecture design used in this report.

3 Sequential Sampling

Sequential sampling, or adaptive sampling, is the act of generating new and optimal samples based on currently existing samples $S = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_i, \ldots, \boldsymbol{\theta}_n\}$ and their realizations $\{f(\boldsymbol{\theta}_1), \ldots, f(\boldsymbol{\theta}_i), \ldots, f(\boldsymbol{\theta}_n)\}$. The aim is to make sure that each new sample is more informative for the problem than randomly drawing each sample, increasing data efficiency meaning potentially fewer samples has to be evaluated. Sequential sampling algorithms build upon the theory of *exploration* and *exploitation* [18, 19]. Exploration refers to searching the parameter space, such that there are no largely unexplored regions of the input space. Exploitation refers to sampling the input space more densely where we are currently uncertain about the output or where the output has been deemed to be more interesting to our end result. Sequential sampling differs from traditionally used one-shot approaches, where all samples are generated before any computations or simulations are performed.

In this section, we cover our slightly simplified implementation of the popular sequential sampling algorithm *Lola-Voronoi* [19], which explores by estimating undersampled regions, and exploits around samples where the output show more non-linear behaviour. However, Lola-Voronoi is designed for one-dimensional scalar function outputs and is not optimal for parameter inference since non-linearity can be high in highly irrelevant regions of the parameter space. For these reasons, in Section 3.2 we propose an alternative sequential sampling algorithm designed for multidimensional sets of summary statistics, prioritizing exploitation where output is similar to a reference series to be inferred.

3.1 The Lola-Voronoi Sampling Algorithm

Lola-Voronoi (Local Linear Approximation-Voronoi) is a sequential sampling approach that utilizes both exploration and exploitation to generate new optimal samples from a current collection of samples $S = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_i, \ldots, \boldsymbol{\theta}_n\}$ and the corresponding function values $\{f(\boldsymbol{\theta}_1), \ldots, f(\boldsymbol{\theta}_i), \ldots, f(\boldsymbol{\theta}_n)\}$ at those points, where $f(\boldsymbol{\theta}) : \mathcal{R}^d \mapsto \mathcal{R}$ is the sampled function. In this section, we cover relevant concepts of the Lola-Voronoi algorithm, but for a more extensive explanation the reader is referred to [19].

A necessary condition for exploration and selecting new samples is that the search space is bounded such that each dimension of sample $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_j, \ldots, \theta_d]^T$ fulfills $\theta_j^{min} \leq \theta_j \leq \theta_j^{max}$. In cases where the search space is not equally as large in each dimension, distance measures would be biased towards dimensions of larger magnitude. To cancel this effect, each dimension j of $\boldsymbol{\theta}$ is normalized prior to any sampling such that

$$\theta_j^{norm} = \frac{\theta_j - \theta_j^{min}}{\theta_j^{max} - \theta_j^{min}}. \tag{2}$$

3.1.1 Voronoi Exploration

The exploration step estimates the volume of the *Voronoi-cell* of each sample $\boldsymbol{\theta}_i$, meaning the parameter region for which $\boldsymbol{\theta}_i$ is the closest sample. As seen in Algorithm 1, this is done by calculating the nearest neighbour of 100N random points within the domain and approximating the volume $V(\boldsymbol{\theta}_i)$ of each Voronoi-cell by the proportion of points for which $\boldsymbol{\theta}_i$ is the nearest neighbour. Here, N is the number of current samples. The exploration step thus increases sampling priority in sparser regions of the sampled space.

Algorithm 1: Exploration through Voronoi-cell volume estimation

Data: S: set of existing samples, Ω: the bounded sample space, N: length of S

```
C ← 100 x N uniformly random points from Ω
for c_i ∈ C do
    Find θ_j ∈ S which is the closest neighbour in S of c_i
    vote[i] ← j
end
for θ_k ∈ S do
    V[θ_k] = count(vote==k) /100N
end
```

The exploration step is the most expensive component of the algorithm since it needs to be recalculated at each new iteration as new samples are added and volumes change, and it needs to find the nearest neighbour for a very large number of points each time.

3.1.2 Local Linearity Exploitation

Exploitation attempts to find where the non-linearity of the current samples is the largest by approximating a gradient for each sample point $\boldsymbol{\theta}_i$ and calculating the residual errors of that approximation for all neighbouring samples of $\boldsymbol{\theta}_i$. If the residual error is significant, so is the non-linearity of that sample point, meaning that exploitation should favour drawing new samples in that area. The original formulation finds the neighbourhood by determining an optimal set of other samples that constructs the best neighbourhood in terms of *adhesion* and *cohesion* [19]. However, in [20], the authors of [19] describe an altered approach in which the neighbourhood is defined as all points within a certain radius of the considered sample. When calculating the gradient, all samples $\boldsymbol{\theta}_j$ in the neighbourhood of $\boldsymbol{\theta}_i$, $N(\boldsymbol{\theta}_i)$, are then weighted by their adhesion A and cohesion C, defined as

$$A(\boldsymbol{\theta}_j, \boldsymbol{\theta}_i) = \min_{\boldsymbol{\theta}_k \in N(\boldsymbol{\theta}_i),\ k \neq j} ||\boldsymbol{\theta}_j - \boldsymbol{\theta}_k||_2, \quad C(\boldsymbol{\theta}_j, \boldsymbol{\theta}_i) = ||\boldsymbol{\theta}_j - \boldsymbol{\theta}_i||_2, \quad \forall \boldsymbol{\theta}_j \in N(\boldsymbol{\theta}_i). \tag{3}$$

In this report, we use a slightly further modified approach compared to [20], where we in addition to limiting samples within a radius r, also consider only a maximum m closest points to $\boldsymbol{\theta}_i$. Further, instead of a rule-based weight assignment, we assign each $\boldsymbol{\theta}_j$ in the neighbourhood the weight W_j, defined as

$$W_j = \frac{1}{2}\left(\frac{A(\boldsymbol{\theta}_j, \boldsymbol{\theta}_i)}{\max_{\boldsymbol{\theta}_k \in N(\boldsymbol{\theta}_i)} A(\boldsymbol{\theta}_k, \boldsymbol{\theta}_i)} + 1 - \frac{C(\boldsymbol{\theta}_j, \boldsymbol{\theta}_i)}{\max_{\boldsymbol{\theta}_k \in N(\boldsymbol{\theta}_i)} C(\boldsymbol{\theta}_k, \boldsymbol{\theta}_i)}\right), \tag{4}$$

such that W_j is in $[0, 1]$ for all j. The gradient $\mathbf{g}(\boldsymbol{\theta}_i) = \big(g_1, \ldots, g_d\big)$ is then computed by solving the following system using the linear least squares method

$$\begin{pmatrix} W_1 \cdot (\boldsymbol{\theta}_1 - \boldsymbol{\theta}_i) \\ \vdots \\ W_n \cdot (\boldsymbol{\theta}_n - \boldsymbol{\theta}_i) \end{pmatrix} \begin{pmatrix} g_1 \\ \vdots \\ g_d \end{pmatrix} = \begin{pmatrix} W_1 \cdot (f(\boldsymbol{\theta}_1) - f(\boldsymbol{\theta}_i)) \\ \vdots \\ W_n \cdot (f(\boldsymbol{\theta}_n) - f(\boldsymbol{\theta}_i)) \end{pmatrix}, \tag{5}$$

where $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n$ are in neighbourhood $N(\boldsymbol{\theta}_i)$, and d is the number of dimensions. The non-linearity $E(\boldsymbol{\theta}_i)$ at $\boldsymbol{\theta}_i$ is then calculated by the weighted sum of absolute errors for the linear approximation in its neighbourhood:

$$E(\boldsymbol{\theta}_i) = \sum_{j=1}^{n} W_j \left| f(\boldsymbol{\theta}_j) - (f(\boldsymbol{\theta}_i) + \mathbf{g}(\boldsymbol{\theta}_i) \cdot (\boldsymbol{\theta}_j - \boldsymbol{\theta}_i)) \right| \tag{6}$$

Thus, regions where the local linear approximation is poor are prioritized for exploitation. The steps for exploitation are summarized in Algorithm 2.

Algorithm 2: Exploitation through gradient based linear approximation

Data: S: set of existing samples

for $\boldsymbol{\theta}_i \in S$ **do**
- $\boldsymbol{\theta}_j \leftarrow$ The 8 closest neighbors of s_i
- $W_j \leftarrow$ Weight of $\boldsymbol{\theta}_j$ according to Equation (4)
- Estimate the gradient of $\boldsymbol{\theta}_i$ using Equation (5)
- $E_i \leftarrow$ The residual of the linear approximation

end

3.1.3 Generation of new Samples

Finally, a total exploration-exploitation score $H(\boldsymbol{\theta}_i)$ is calculated as the sum of V and E, while dividing each element of E with the total sum of E to make both $E(\boldsymbol{\theta}_i)$ and $V(\boldsymbol{\theta}_i)$ in $[0, 1]$ for all $\boldsymbol{\theta}_i$:

$$H(\boldsymbol{\theta}_i) = a \cdot V(\boldsymbol{\theta}_i) + b \cdot \frac{E(\boldsymbol{\theta}_i)}{\sum_j E_j}, \tag{7}$$

where a, b are weights both equal to one unless otherwise specified. By adjusting these weights, one can prioritize either exploration or exploitation. For example, if $a = 1$, and $b = 2$, E is weighed twice that of the V, prioritizing exploitation over exploration.

N new samples are generated by selecting the N samples with the highest $H(\boldsymbol{\theta})$. For each of those samples $\boldsymbol{\theta}_i$, we generate 1000 candidate samples uniformly within a radius R of $\boldsymbol{\theta}_i$. From these candidate samples, we select a point with θ_i as its closest neighbour that maximizes the distance to $\boldsymbol{\theta}_i$.

For high-dimensional problems, we observe that maximizing this distance leads to most new samples being generated on the parameter boundaries. Therefore, a maximal radius R_{max} is defined to limit this behaviour. Further, in rare cases where two existing samples are separated with only a small distance, problems can occur when none of the candidate samples have $\boldsymbol{\theta}_i$ as their closest neighbour. We ease this problem by generating a new set of candidate samples with a refined radius $R \leftarrow R/2$ and repeating the process until a suitable sample is found. We illustrate the process in Algorithm 3. For extraordinary cases where no samples have been generated after ten refinements, we generate a random sample in the parameter domain and reduce the exploitation score of the corresponding $\boldsymbol{\theta}_i$ with 5% to deter further sampling around that point.

An iterative approach can generate N samples per iteration until we reach some target number of samples. While more expensive, especially for a larger number of dimensions, it is favourable to use small values of N for optimal sample generation. Low values of N ensure that only the highest-ranked regions are sampled. Each step of the sample generation process is described in Algorithm 3 below.

Algorithm 3: Drawing new samples

Data: H: The total score of the existing samples, N: The number of samples to be drawn, R_{max}: The maximum search radius.

```
S(N) ← The samples corresponding to the N largest values of H
for θ_i ∈ S(N) do
    Establish an area with radius R around θ_i
    R ← The distance to furthest neighbour of θ_i
    R ← min(R, R_max)
    while No sample selected do
        Generate 1000 candidate samples in area around θ_i
        Force candidate samples into sampled space if outside
        Discard candidate samples outside the Voronoi cell of θ_i
        if any candidate samples remaining then
            new_samples[i] ← candidate sample with maximum separation from
              preexisting samples
        else
            R ← R/2
        end
    end
end
```

3.2 Summary Statistic Similarity-Based Exploitation

Assume we have one or more observed time series, corresponding to a simulation model whose parameters are to be inferred. We introduce an approach of exploiting regions of the parameter space which produce time series with similar summary statistics, further referred to as *Similarity-Voronoi* within this report. In contrast to Lola-Voronoi sampling, where the non-linearity is measured, this could be more effective when producing training data similar to observed results whose parameters we wish to infer. Another motivation is that the summary statistics of a stochastically generated time series are inherently not smooth functions, meaning noise risks creating significant local non-linearity. Further, this measure does not rely on neighbourhood calculations necessary for calculating the gradient and non-linearity. Therefore, it does not require updating as new samples are added, making it computationally cheaper than Lola-Voronoi exploitation.

For a set of summary statistics $stat_1, \ldots, stat_K$, we denote the summary statistics of a simulation using the parameters $\boldsymbol{\theta}_i = [\theta_1, \ldots, \theta_d]$, as $stats(\boldsymbol{\theta}_i) = [stat_1(\boldsymbol{\theta}_i), \ldots, stat_K(\boldsymbol{\theta}_i)]$, and the summary statistics of the observed time series as $stats_{true}$. The relative distance of a simulation is then calculated using the Euclidean norm in the following way:

$$d(\boldsymbol{\theta}_i) = \left\| \frac{stats(\boldsymbol{\theta}_i) - stats_{true}}{stats_{true}} \right\|_2. \tag{8}$$

Note that if we have more than one observation, $stats_{true}$ can represent their mean aggregate summary statistic vector, or a separate exploitation score for each observation could be calculated and then summed. For the rest of this report, we will only consider cases with one observation. The choice of using the Euclidean distance of summary statistics was not analysed extensively, and other alternatives such as a weighted Euclidean distance or adaptive distance measures have been studied in literature [21].

Since we desire a larger E where the similarity is high, we invert d to produce

$$E(\boldsymbol{\theta_i}) = (d(\boldsymbol{\theta_i}) + c)^{-1}, \tag{9}$$

where c is a constant to smooth out the exploitation score. Smaller values of c can lead to over-exploitation around samples with very large similarity, but values too large will make the algorithm purely exploratory. Generally, the more species and summary statistics present in the model, the lower the value of c can be, since similarities will decrease due to increasing random discrepancies. It can even assume negative values as long as $d(\boldsymbol{\theta_i}) + c > 0$ for all i.

Algorithm 4 summarizes the algorithm. Note that in contrast to Algorithm 2, since E is no longer dependant on its neighbouring samples, it is not updated in each iteration. Other than modifying the exploitation score to this, the Lola-Voronoi algorithm is used.

Algorithm 4: Exploitation through Similarity in Summary Statistics

Data: $stats_{true}$: summary statistic vector of the observed time series
$\boldsymbol{\theta}_i$, $\boldsymbol{y}_i$: a sampled parameter set and corresponding simulated time series

1 $stats(\boldsymbol{\theta}_i) \leftarrow$ Vector of summary statistics computed from $\boldsymbol{y}_i$
2 $d(\boldsymbol{\theta}_i) \leftarrow \left\| \frac{stats(\boldsymbol{\theta}_i) - stats_{true}}{stats_{true}} \right\|_2$ ▷ Element-wise division
3 $E(\boldsymbol{\theta}_i) \leftarrow (d(\boldsymbol{\theta_i}) + c)^{-1}$

4 Setup and Implementation

This section describes the implementations used to conduct numerical experiments on the sampling algorithms. The experiments include sampling and parameter inference using uniform and sequential sampling techniques. All implementations are written in `Python 3.9`.

4.1 Sequential Sampling Schemes

Both sequential sampling schemes used implementations very similar to what is described in Algorithms 1, 2, 3 and 4 with some minor optimizations.

Since the algorithms depend both on distance measures and nearest neighbour searches, we use the `SciPy` [22] implementation of a KD-tree. A KD-tree is a tree-based data structure allowing for fast nearest neighbour calculations with average time complexity of $\mathcal{O}(\log N)$, in addition to distance calculations. For our sequential sampler design, this is very useful in the Voronoi-exploration stage, in which $100N$ nearest neighbour queries are performed for N existing samples, and for the generation of new samples, which relies on distance measures for the 1000 candidate samples. As mentioned in [23], the complexity with regards to dimensionality can be reduced if an approximate nearest neighbour algorithm is applied. This was used to alleviate the problems of high computational times for our final 15-dimensional problem, we use an approximate nearest neighbour search with $\varepsilon = 0.5$ for the genetic oscillator. Since the Voronoi cell volume estimation is already highly approximated, we assume that the inaccuracies caused by this will not lead to any noticeable differences in sampling behaviour.

4.2 Convolutional Neural Networks

For all parameter inference problems in this report, we use a predefined CNN model from the `Sciope` framework [24], which is designed for parameter inference. The input shape corresponds to [number of output time series $\times$ number of timesteps] and the output shape is equal to the number of inferred parameters. A learning rate $\gamma = 0.001$ and 500 epochs with an early stopping condition of five epochs is used in the training of the CNN model. A batch size of 256 is used for the first two problems, and for the final problem a batch size of 1024 is used.

4.3 Error Measurements

This report uses two related error measures, the Mean Absolute Error (MAE) and the Normalized Mean Absolute Error ($\text{MAE}_{\%}$).

The MAE [25] is defined as

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |\boldsymbol{\theta_i} - \hat{\boldsymbol{\theta}}(Y_i)|, \tag{10}$$

where $\boldsymbol{\theta_i}$ is the true parameter values and $\hat{\boldsymbol{\theta}}(Y_i)$ is the predicted parameters from an observed Y_i. In the case of parameter inference, to account for when the parameter space

is not as large in each dimension, the normalized MAE from [6] is also introduced as

$$MAE_{\%} = \frac{1}{n}\sum_{i=1}^{n} 4|\boldsymbol{\theta}_i^{norm} - \hat{\boldsymbol{\theta}}^{norm}(Y_i)|, \tag{11}$$

where $\boldsymbol{\theta}^{norm}$ is the normalization of $\boldsymbol{\theta}$ in which each dimension j of $\boldsymbol{\theta}$ is transformed using the parameter boundaries according to Equation (2). Conveniently, a $\text{MAE}_{\%}$ of one is then the average error if one would always guess the centre of the parameter domain for random test points.

5 Experimental Problems

To explore the performance of the sequential sampling techniques, we employ four experimental problems. In Section 5.1, we use the Lola-Voronoi algorithm on the deterministic *peaks*-function with a one-dimensional output. A simple neural network is trained as a global surrogate model to predict the function value at new points. The prediction performance of the testing data for sequentially generated training data is compared to randomly generated training data. In Sections 5.2, 5.3, and 5.4 we use the same experimental models as in [6], from where we also take the implementations of these models. These three models output simulated stochastic time series for a set of input parameters, and the Similarity-Voronoi sequential sampling algorithm is used to create a training data set. The first of these models is the Moving Averages model of second-order, a highly stochastic model of two parameters. In Section 5.3, the three-dimensional *Lotka-Volterra* model is used. This popular benchmark problem models the interaction between species of prey and predators. Finally, in Section 5.4, a computationally expensive 15-dimensional genetic oscillator is used. The performance of the sequential sampling scheme on these problems is evaluated using the CNN to infer parameters from an observed simulation.

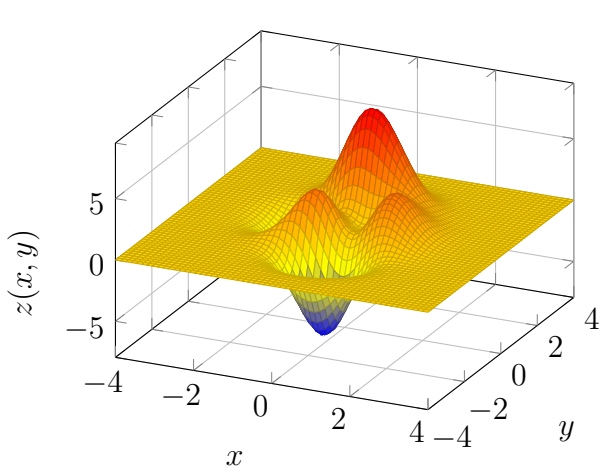
5.1 Peaks Function

As a proof of concept for the Lola-Voronoi sampling implementation, we use the so-called *peaks*-function as a benchmark problem. The function's behaviour can be seen in Figure 1a and is defined as:

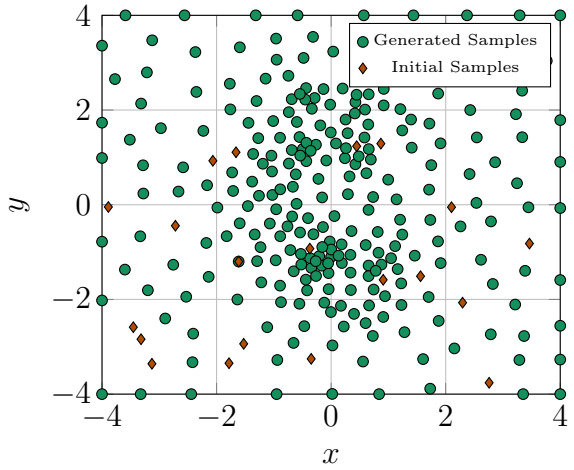
$$z(x,y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10(\frac{x}{5} - x^3 - y^5)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2}. \tag{12}$$

Starting from 20 initial random samples, the implementation of the Lola-Voronoi algorithm iteratively generates new samples until a total of 256 have been drawn within the domain $\Omega : \{x, y : x, y \in [-4, 4]\}$. This is shown in Figure 1b where the red dots represent the initial samples and the green the generated ones. Observe that the samples are denser in places where $z(x, y)$ changes rapidly and where the non-linearity is larger.

A simple neural network with three fully connected 20-neuron layers and one output layer is implemented and trained using samples generated by the algorithm, with 300 epochs using a batch size of 50. The network learns to predict $\hat{z}(x, y)$ using training data



(a) The Peaks Function in the domain $x, y \in [-4, 4]$



(b) 236 samples generated by the Lola-Voronoi algorithm (green) from 20 initial samples (red) for a total of 256 samples

Figure 1: The peaks function and samples generated by the Lola-Voronoi algorithm

MAE of NN on Peaks for Different Sampling Techniques

Sampling Technique	Training ($\mu \pm \sigma$)	Testing ($\mu \pm \sigma$)
Uniform	$\mathbf{0.4333 \pm 0.2891}$	0.6694 ± 0.3144
Equidistant Grid	0.4580 ± 0.2855	0.4994 ± 0.3017
Lola-Voronoi	0.5344 ± 0.4140	$\mathbf{0.4107 \pm 0.2740}$

Table 1: The MAE errors in training and testing for a neural network prediction model using 256 training samples generated by different sampling techniques. Values shown are the mean values $\pm$ the standard deviations from 100 independent experiments.

$\{x_i, y_i, z(x_i, y_i)\}_{i=1}^n$. As a comparison, we also train the model using samples drawn from a uniform distribution over the domain as well as a 16×16 equidistant grid. The models are evaluated using 2025 test samples randomly drawn from Ω. Means and standard deviations of the MAE for training and testing data over 100 independent experiments are shown in Table 1, where it is seen that the Lola-Voronoi algorithm outperforms both random uniform sampling and grid-based sampling techniques for the test data. The higher error in training data for sequential sampling is explained by the fact that the Lola-Voronoi algorithm generates more training data in the non-linear regions, which should be more difficult to predict, leading to larger average errors in the training data. These results are consistent with those in [19]. They indicate sequential sampling allowing us to train more accurate models with the same sample count or equal models with the same sample count.

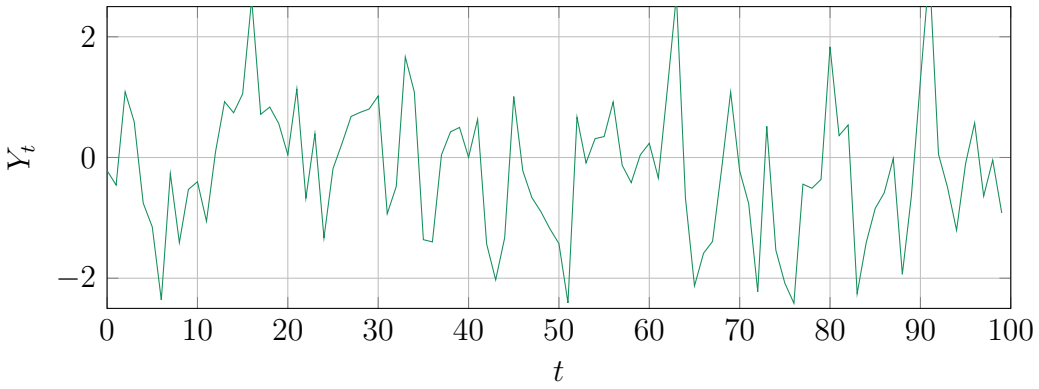


Figure 2: A MA2 simulation with parameters $\theta_1 = 0.6$, $\theta_2 = 0.2$

5.2 The Moving Averages Model of 2nd Order

5.2.1 Model Definition

The Moving averages of order 2 (MA2) model is in this report defined for $T = 30$ time steps as

$$Y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2}, \tag{13}$$

where $\varepsilon_{t>0} \sim \mathcal{N}(0, 1^2)$, $\varepsilon_{t\leq 0} = 0$ are white-noise terms and $t = 1, \ldots, T$. Further, we consider a triangular parameter space defined by

$$\theta_1 \in [-2, 2],\ \theta_2 \in [-1, 1],\ \theta_2 \pm \theta_1 \geq -1. \tag{14}$$

An observed time series is generated using the true parameters $\theta_1^{true} = 0.6$ and $\theta_2^{true} = 0.2$, one example simulation for these parameters is seen in Figure 2.

5.2.2 Comments on Sequential Sampling

The proposed Similarity-Voronoi sequential sampling scheme using summary statistic distance as exploitation metric described in Section 3.2 is used on the MA2 model. In Figure 3, we show a histogram of 5000 samples generated by our scheme, where the red dot represents the true parameters. It can be seen that most samples are drawn in the area around the observed point. Consequently, it can be concluded that the alternative exploitation metric used for the model is a viable option for this problem.

However, while no exact measurements were made, for such a cheap simulation model the sequential sampling step adds a significant computational time for little to no gain.

5.2.3 Parameter Inference

From 200 initial uniformly random samples within the defined parameter space, 100 new samples are generated by the sequential algorithm with $c = 1.4$ in each iteration until 15000 samples are simulated. c was chosen with limited manual testing. Using these samples as well as 15000 random samples to train the CNN, we compare the $\text{MAE}_{\%}$ for parameter inference of a random test set and inference for the observed true series in Table 2. Further, in Figure 4 it is seen how this error changes for testing and observed data with

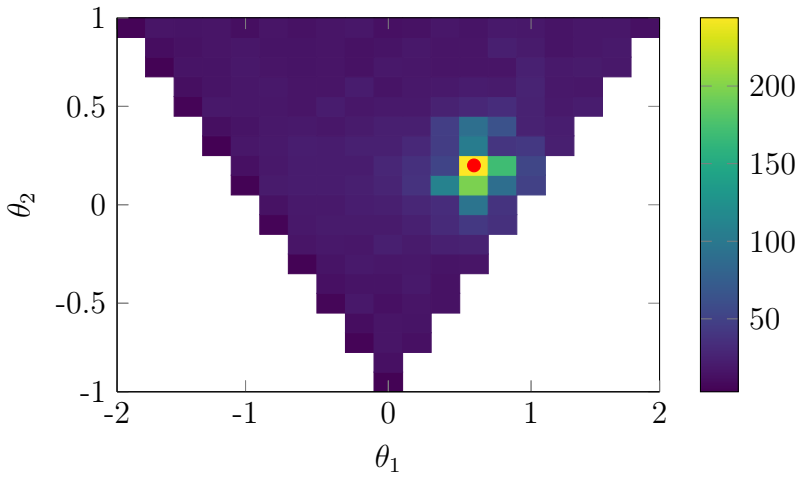


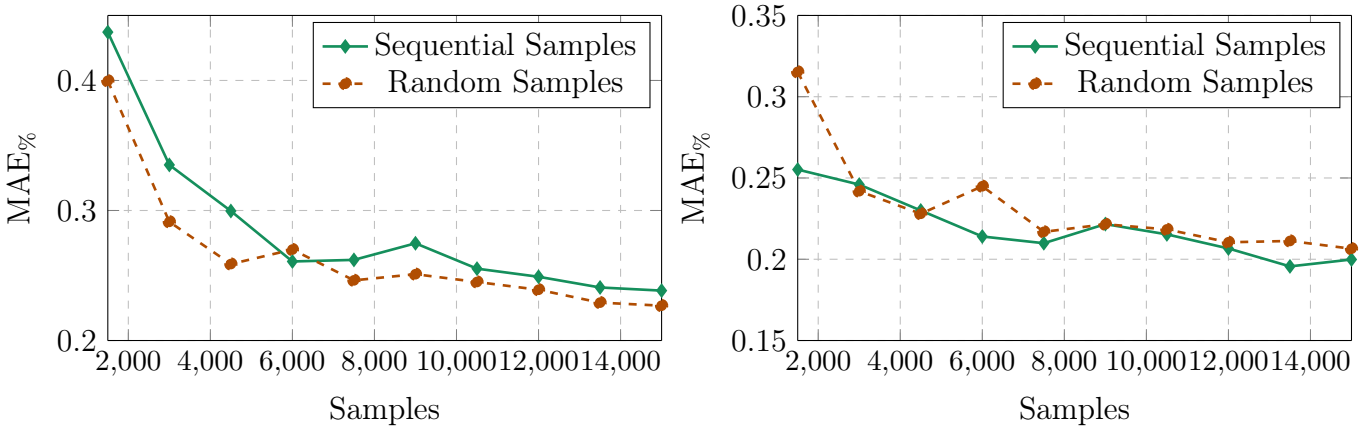
Figure 3: A histogram showing the observed point (red) and sample density for samples generated by the Similarity-Voronoi algorithm for the MA2 model

Parameter Inference $MAE_{\%}$ of CNN on MA2, 15000 samples, N=35

Sampling Technique	Testing ($\mu \pm \sigma$)	Observed ($\mu \pm \sigma$)
Uniform Sampling	$\mathbf{0.2248 \pm 0.0222}$	0.1998 ± 0.0490
Similarity-Voronoi	0.2418 ± 0.0234	$\mathbf{0.1956 \pm 0.0398}$

Table 2: $MAE_{\%}$ for parameter inference of the MA2 problem on random testing data and on an observed time series generated with parameters $\theta_1 = 0.6$, $\theta_2 = 0.2$. The CNN is trained with 15000 samples, and values shown are mean $\pm$ standard deviation from 35 experiments.

the number of samples, averaged over 35 experiments. While the $MAE_{\%}$ on average is lower for sequentially than uniformly generated data for inference on the observed series, the difference is small. The difference was expected to be larger after the results in Figure 3, but both errors seem to have converged by the time we reach 3000 samples.



(a) The $MAE_{\%}$ for random testing data

(b) The $MAE_{\%}$ for observed data

Figure 4: $MAE_{\%}$ for random testing data (a) and the observed time series (b) for parameter inference using the CNN with sequentially and randomly generated samples

5.3 The Lotka-Volterra Model

5.3.1 Model Definition

The Lotka-Volterra model is a prey-predator model used to simulate biological systems where one species of prey and one species of a predators interact. We consider the prey population χ_1 and the predator population χ_2 to be modelled by the differential equations

$$\begin{aligned} \frac{d\chi_1}{dt} &= \theta_1\chi_1 - \theta_2\chi_1\chi_2 \\ \frac{d\chi_2}{dt} &= \theta_2\chi_1\chi_2 - \theta_3\chi_2, \end{aligned} \tag{15}$$

with constant parameters θ_1, θ_2, and θ_3. The implementation is the same as in [6], which uses a stochastic Markov jump process implemented using `Gillespy2` which is part of the `Sciope` toolkit.

We simulate three events between each time-step: the reproduction of prey, the prey consumption by predators, and the death of predators. The rates of these events are governed by θ_1, θ_2, and θ_3 respectively. The simulation risks diverging for parameter combinations of high θ_1 and low θ_2. Thus, we abort simulations that do not finish within an a priori set time limit as they are assumed to have diverged. Such simulations are not kept as training data for the learning model but are kept within the sequential sampling scheme with an assigned exploitation score of zero to deter sampling around those points.

All simulations are performed for times $t = 0, 1, \ldots, 29$, with initial conditions $\chi_1 = 50$, $\chi_2 = 100$, and the parameter space is bounded to $\theta_1, \theta_2, \theta_3 \in [0.005, 6.0]$. For the inference problem, an observed time series was generated using the true parameters $\theta_1^{true} = 1.0$, $\theta_2^{true} = 0.005$, $\theta_3^{true} = 0.6$. As an example, one simulation with those parameters can be seen in Figure 5.

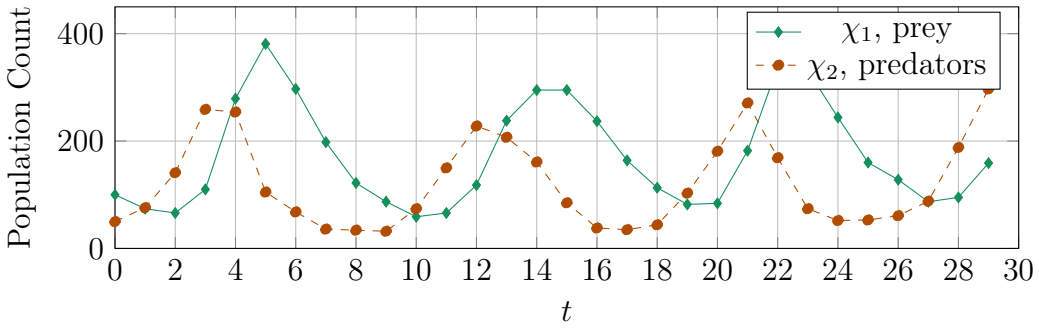


Figure 5: A simulation of Lotka-Volterra equations with parameters $\theta_1 = 1.0$, $\theta_2 = 0.005$, $\theta_3 = 0.6$

5.3.2 Comments on Sequential Sampling

The Similarity-Voronoi sequential sampling scheme is initiated with 200 randomly simulated samples, after which 100 samples are added per iteration until a target of samples

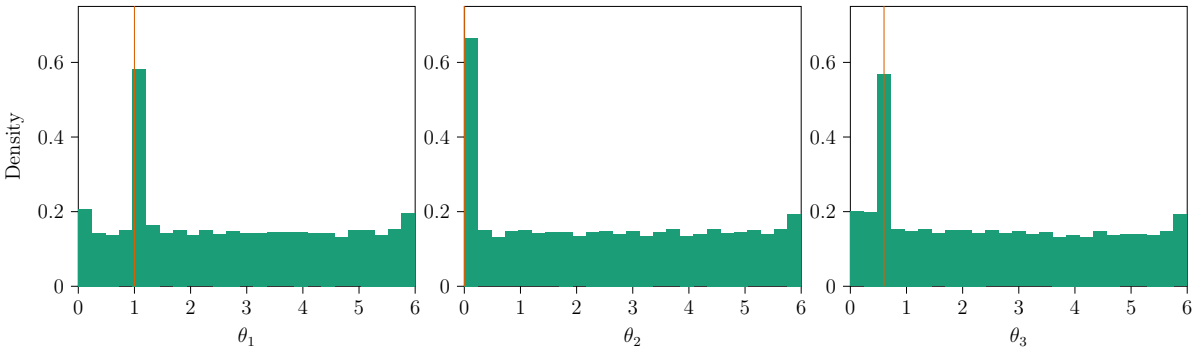


Figure 6: Parameter histograms after 5000 sequentially generated samples. The orange lines represents the true latent parameter values $\theta_1 = 1.0$, $\theta_2 = 0.005$, and $\theta_3 = 0.6$.

is reached. A value of $c = 0.7$ is used for the sampler design for this problem. Compared to the previous MA2 problem, the increased complexity allows for a lower c as discussed in Section 3.2, but too low values results in exploitation before any adequate exploration is completed. Sample entropy as a summary statistic was not considered for the Lotka-Volterra model, as many normal simulations have an infinite sample entropy for non-investigated reasons.

Figure 6 shows parameter-wise histograms after 5000 sequential samples, where we have a significant peak around the true latent parameter values. While the Similarity-Voronoi algorithm manages near perfect sampling in this problem, the non-smoothness of the histograms might be indicative of the exploitation function being too sensitive, and that it might prioritize exploitation too heavily once a similar region has been discovered.

5.3.3 Parameter Inference

The sampled data set is now $\{\theta_1^i, \theta_2^i, \theta_3^i, \chi_1^i(t), \chi_2^i(t)\}_{i=1}^n$. The same CNN model from earlier is trained using these samples, with an input shape of [2 species $\times$ 30 time steps] and outputs the three parameters. An additional 10000 random simulations are generated as a testing data set. For comparison, the inference model is also trained on random samples from a uniform distribution over the parameter domain.

The $\text{MAE}_{\%}$ for different sample counts are presented in Table 3 for parameter inference on the random test set and for the observed time series. It is seen that the sequential sampling algorithm significantly outperforms the uniform sampling for inference on an observed time series for all tested sample counts. We believe this difference mainly stems from the fact that a relatively small volume of the parameter space in the Lotka-Volterra problem generates non-zero or non-diverging solutions allowing the sequential algorithm to focus on parameter regions producing interesting outputs. Additionally, the increase in dimensionality should penalize sequential sampling less than uniform sampling. As expected, the sequential algorithm performs worse on random testing data since it essentially limits itself to one region. If more accurate inference is desired around all oscillatory solutions in general and not just around one specific simulation, our approach could easily be modified to exploit such behaviour in general.

Parameter Inference $\text{MAE}_{\%}$ of CNN on Lotka-Volterra

	Uniform Sampling		Similarity-Voronoi	
Samples	Testing	Observed	Testing	Observed
5000	$\mathbf{1.1015 \pm 0.0800}$	1.4183 ± 0.7694	1.3450 ± 0.2243	$\mathbf{0.5892 \pm 0.4259}$
10000	$\mathbf{1.0984 \pm 0.0955}$	1.2699 ± 0.3997	1.4340 ± 0.2118	$\mathbf{0.3926 \pm 0.3916}$
30000	$\mathbf{1.0003 \pm 0.0992}$	1.2528 ± 0.2271	1.3066 ± 0.2566	$\mathbf{0.3426 \pm 0.3731}$
100000	$\mathbf{0.7426 \pm 0.0174}$	0.5743 ± 0.1618	0.7508 ± 0.0245	$\mathbf{0.0967 \pm 0.1072}$

Table 3: $\text{MAE}_{\%}$ for inference of random testing data, and for inference of the observed time series with $\theta_1 = 1.0$, $\theta_2 = 0.005$, and $\theta_3 = 0.6$. 10^4 testing data points are randomly drawn from a uniform distribution over the whole domain, and 100 time series are generated with the true parameters for error estimation around observed parameters. Values represent mean ± standard deviation over 25 independent experiments.

5.4 Genetic Oscillator Model

5.4.1 Model Definition

For a more demanding and high-dimensional problem, we test a 15-parameter genetic oscillator, observing the population time series for nine interacting species. This model is the oscillatory circadian clock from [26], defined as

$$\begin{aligned}
\frac{dD_A}{dt} &= \theta_A D'_A - \gamma_A D_A A & \frac{dD_R}{dt} &= \theta_R D'_R - \gamma_R D_R A \\
\frac{dD'_A}{dt} &= \gamma_A D'_A A - \theta_A D'_A & \frac{dD'_R}{dt} &= \gamma_R D_R A - \theta_R D'_R \\
\frac{dM_A}{dt} &= \alpha'_A D'_A + \alpha_A D_A - \delta_{M_A} M_A & \frac{dC}{dt} &= \gamma_C AR - \delta_A C \\
\frac{dM_R}{dt} &= \alpha'_R D'_R + \alpha_R D_R - \delta_{M_R} M_R & \frac{dR}{dt} &= \beta_R M_R - \gamma_C AR + \delta_A C - \delta_R R \\
\frac{dA}{dt} &= \beta_A M_A + \theta_A D'_A + \theta_R D'_R - A(\gamma_A D_A + \gamma_R D_R + \gamma_C R + \delta_A),
\end{aligned} \tag{16}$$

with the nine species D_A, D'_A, D_R, D'_R, M_A, M_R, C, A, R and the 15 parameters $\theta_{A,R}$, $\gamma_{R,C,A'}$, $\alpha_{A,R,A',R'}$, $\beta_{A,R}$, δ_{A,R,M_A,M_R}. As initial conditions, we use $D_R, D_A = 1$ with the remaining species at zero. Simulations are performed with the time vector $t = 0, 1, \ldots, 200$.

The high-dimensional problem is an informative test scenario for two main reasons. Firstly, the Voronoi-cell volume estimation for exploration in our implementation of the Similarity-Voronoi algorithm utilizes a KD-tree [27] for nearest neighbour lookup. Unfortunately, the computational performance of a KD-tree has been demonstrated to suffer from "*the curse of dimensionality*" [23], leading to a growing time complexity for the number of dimensions. Secondly, it is expected that the benefit of a sequential sampling algorithm should be more noticeable for higher dimensionalities because of the aforementioned curse, data becomes sparser. However, the sampling risks being largely exploratory

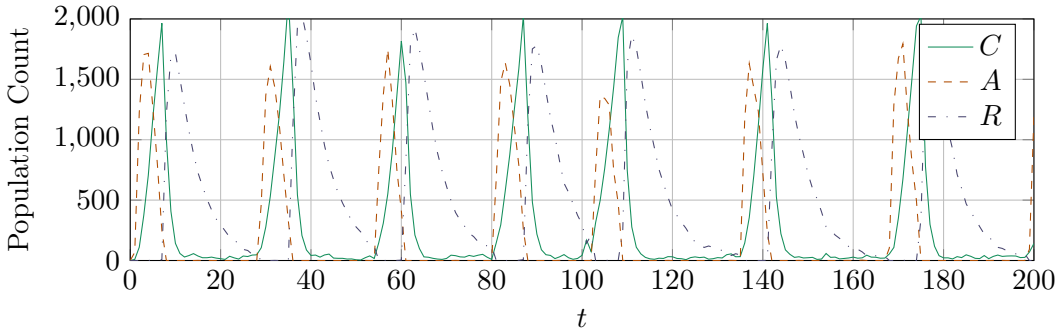


Figure 7: A Genetic Oscillator simulation of the species C, A and R using the true parameters found in Table 4

for longer since the odds of randomly generating a sample close to the observed point decreases rapidly for the same reasons.

5.4.2 Comments on Sequential Sampling

In Figure 8 we show histograms for each parameter after 40000 samples have been sequentially generated. As reference, the true latent parameter values of the observation is included. We note that the Similarity-Voronoi scheme has sampled most parameters more densely in the vicinity of the true value, while some are more uniform and off-target. Additionally, in Figure 9 we show the histogram for when we let $b = 3$ in Equation (7), meaning that we prioritize exploitation thrice as much as exploration. Although the histograms are slightly more peaked in the second figure, they look similar overall.

The exact reasons why some parameters are not sampled densely around the true value is currently unexplored. When exploring the model, [26] found that β_A, β_R, δ_{M_A}, δ_{M_R} have less of an impact on oscillations than other parameters, and that δ_A and δ_R are more sensitive. In the histograms, we see that β_A, β_R, δ_{M_A} are sampled more uniformly, while δ_A and δ_R are centred around the true point. Further, the high-dimensionality of the parameter space means it is difficult to find a point close to the true observations, and that other points are sufficiently similar to the observation for heavy exploitation in those areas. Another reason could be that the chosen summary statistics are not able to distinguish behaviours dependent of e.g. δ_{M_A} well enough, leading to exploitation in the wrong areas. Therefore, a potential improvement of the Similarity-Voronoi scheme is to use automatic summary statistic selection such as approximate sufficiency [16].

Finally, we comment on the additional computational cost introduced by doing sequential sampling. When iterativly adding 2000 samples until $2 \cdot 10^6$ total samples; the time spent on sequential sampling was 23% of that of the simulations, almost entirely being nearest neighbour queries. Due to the logarithmic search times for the nearest neighbour we expect a time complexity of $\mathcal{O}(N \cdot \log N)$ for each iteration where N is the number of current samples. However, these timings are inconclusive as no extensive testing was done and our Similarity-Voronoi implementation ran in serial compared to the simulations which ran on 8 CPU-cores in parallel. The sequential sampling should be parallelizable, and there exists potentially more effective KD-tree implementations [28] than the `SciPy` used here.

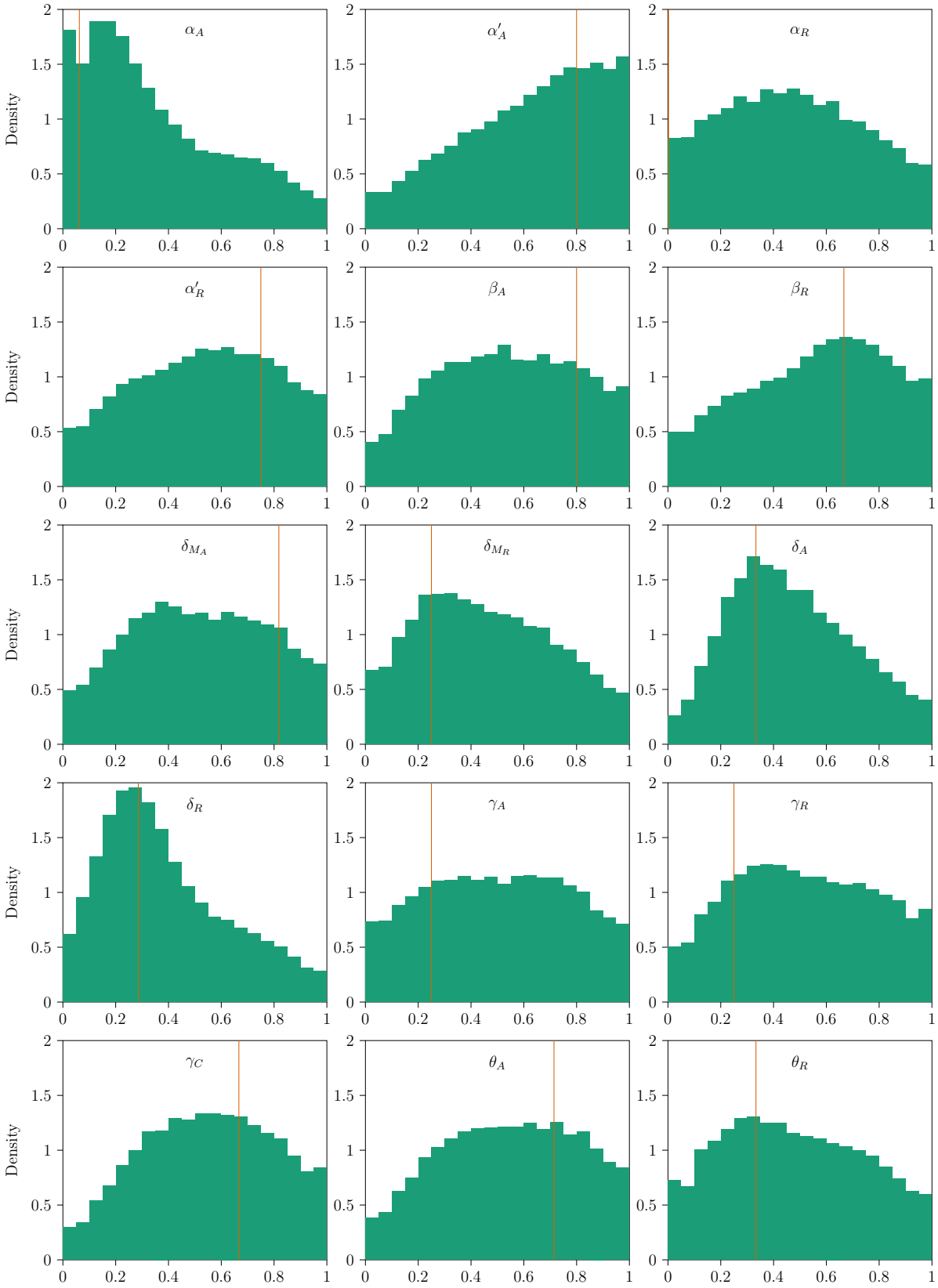


Figure 8: Histograms for all 15 parameters (normalized) in the genetic oscillator for 40000 sequentially generated samples, with equal weights for exploration and exploitation. The orange line represents the true parameter value.

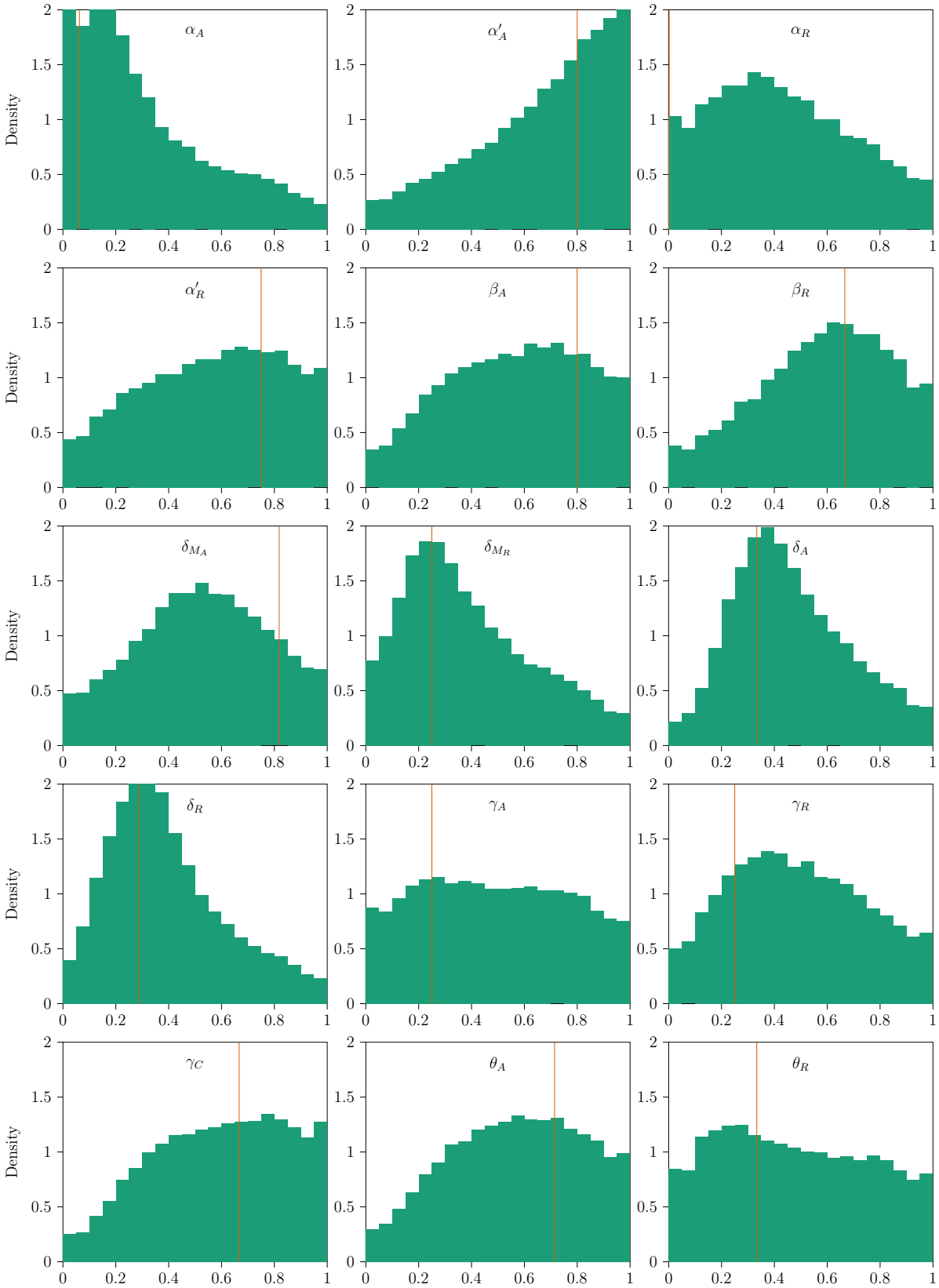


Figure 9: Histograms for all 15 parameters (normalized) in the genetic oscillator for 40000 sequentially generated samples, when exploitation is weighted thrice that of exploration. The orange line represents the true parameter value.

Boundaries of Parameter Search Space and True Parameters for Genetic Oscillator

	α_A	α'_A	α_R	α_R	β_A	β_R	δ_{M_A}	δ_{M_R}	δ_A	δ_R	γ'_A	γ_R	γ_C	θ_A	θ_R
min	0.0	100.0	0.0	20.0	10.0	1.0	1.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0
max	80.0	600.0	4.0	60.0	60.0	7.0	12.0	2.0	3.0	0.7	2.5	4.0	3.0	70.0	300.0
True	5.0	500.0	0.01	50.0	50.0	5.0	10.0	0.5	1.0	0.2	1.0	1.0	2.0	50.0	100.0

Table 4: The lower and upper boundaries of the parameter search space for the 15-dimensional genetic oscillator, as well as the true parameters used to simulate the observed time series.

5.4.3 Parameter Inference

The parameter search space boundaries are given in Table 4, using the same boundaries as [6]. It is noted that the parameter search spaces differ in the order of magnitudes, highlighting the importance of normalizing the dimensions before doing any sequential sampling. True parameter values used to generate the observed time series are also included in the table. We consider only the species C, A, and R for parameter inference. Species C, A, R for a simulation with the true parameters are visualised in Figure 7. We add 2000 samples in each sequential iteration, with 2000 initial samples and using $c = 0.1$. Due to the high complexity of this problem, the probability of observing similar summary statistics is very low, allowing for a lower c without risking exploitation too early. Even lower values of c have been tested without observing problematic behaviour, but settled on the more conservative value in experiments. Five sequential sampling runs are performed, and five networks are trained and tested for each sample pool for a total of 25 tests. The results of these tests are seen in Table 5 for inference of the observed point, and in Table 6 for random test points in the domain. Once again, we compare these results with networks trained on uniformly random samples. For additional sample counts, the mean $\text{MAE}_\%$ is seen in Figure 10.

In Table 5, we notice an overall improvement when training on the sequentially generated training sets, although the errors for some parameters are lower for uniform data. This could be explained by some of the parameters not being sampled around the true points as seen in the histograms in Figure 8, 9. Further, we note that the error between the different sequentially generated data sets differ significantly: some show mean $\text{MAE}_\%$ around 0.16 at 200000 samples, while others show around 0.45. This behaviour was not found for uniform training sets, indicating that the sequential process may require a initial samples in regions of interest to find a good area to exploit. If no point near the observation's latent parameters is generated, the sequential process risks exploiting in unwanted regions. Therefore, a large number of initial samples is recommended.

When testing on random data instead of around the observed point, it is clear in Table 6 that uniform sampling outperforms sequential sampling in each parameter, except for low sample counts. We believe the improvement for low sample counts for the sequential sampling is due to its initial exploratory behaviour, ensuring a good spread of samples. The improvement for uniformly random training data on random testing data is consistent with the results in the previous experiments. It is apparent that uniform sampling or purely exploration-based sequential sampling approaches should be preferred if inference is to be done throughout the entire parameter domain with a larger sample count.

Parameter Inference $\text{MAE}_{\%}$ for Observed Genetic Oscillator Time Series

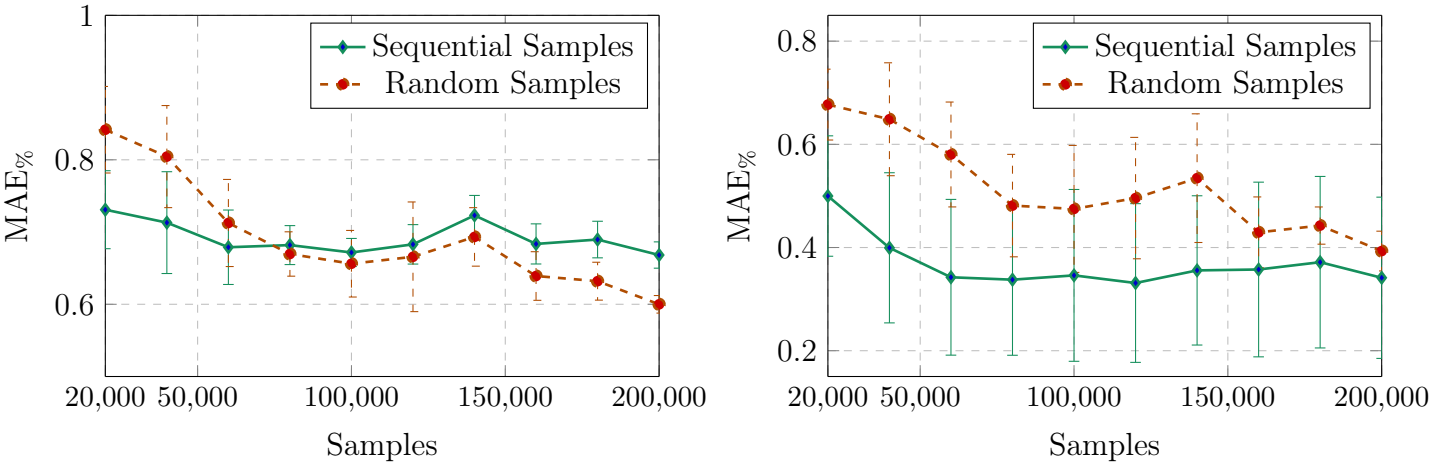
	$N = 2 \cdot 10^4$		$N = 6 \cdot 10^4$		$N = 10^5$		$N = 2 \cdot 10^5$	
Parameter	Uniform	Sequential	Uniform	Sequential	Uniform	Sequential	Uniform	Sequential
α_A	0.215 ± 0.246	0.230 ± 0.120	0.270 ± 0.151	0.145 ± 0.157	0.266 ± 0.268	0.085 ± 0.055	0.207 ± 0.100	$\mathbf{0.065 \pm 0.028}$
α'_A	0.397 ± 0.230	0.330 ± 0.165	0.421 ± 0.138	0.253 ± 0.177	0.309 ± 0.108	0.229 ± 0.152	$\mathbf{0.246 \pm 0.133}$	0.257 ± 0.120
α_R	1.448 ± 0.272	0.620 ± 0.367	1.048 ± 0.479	0.237 ± 0.224	0.667 ± 0.494	0.173 ± 0.060	0.373 ± 0.148	$\mathbf{0.122 \pm 0.052}$
α'_R	0.595 ± 0.311	0.358 ± 0.272	0.337 ± 0.237	0.289 ± 0.136	0.252 ± 0.202	0.441 ± 0.376	$\mathbf{0.226 \pm 0.082}$	0.426 ± 0.360
β_A	0.891 ± 0.275	0.660 ± 0.265	0.830 ± 0.160	0.453 ± 0.233	0.606 ± 0.200	0.380 ± 0.194	$\mathbf{0.341 \pm 0.104}$	0.399 ± 0.235
β_R	0.354 ± 0.248	0.308 ± 0.200	0.306 ± 0.146	0.279 ± 0.187	$\mathbf{0.261 \pm 0.124}$	0.345 ± 0.283	0.387 ± 0.095	0.313 ± 0.231
δ_{M_A}	1.836 ± 0.183	0.920 ± 0.446	1.339 ± 0.366	0.425 ± 0.253	1.006 ± 0.317	0.327 ± 0.142	0.672 ± 0.207	$\mathbf{0.291 \pm 0.129}$
δ_{M_R}	0.212 ± 0.155	0.208 ± 0.072	0.209 ± 0.179	0.136 ± 0.040	0.167 ± 0.063	$\mathbf{0.122 \pm 0.036}$	0.168 ± 0.104	0.096 ± 0.021
δ_A	0.561 ± 0.231	0.218 ± 0.135	0.280 ± 0.173	0.108 ± 0.082	0.167 ± 0.054	$\mathbf{0.061 \pm 0.018}$	0.162 ± 0.068	0.064 ± 0.034
δ_R	0.545 ± 0.315	0.243 ± 0.139	0.426 ± 0.287	0.118 ± 0.065	0.410 ± 0.341	0.094 ± 0.036	0.169 ± 0.066	$\mathbf{0.065 \pm 0.024}$
γ_A	0.658 ± 0.207	0.649 ± 0.340	0.608 ± 0.308	0.520 ± 0.405	0.557 ± 0.221	$\mathbf{0.421 \pm 0.370}$	0.510 ± 0.273	0.445 ± 0.332
γ_R	0.933 ± 0.148	1.182 ± 0.523	0.950 ± 0.220	0.821 ± 0.728	1.037 ± 0.185	0.987 ± 0.839	$\mathbf{0.819 \pm 0.289}$	0.921 ± 0.744
γ_C	0.311 ± 0.205	0.406 ± 0.147	0.502 ± 0.275	0.356 ± 0.151	$\mathbf{0.259 \pm 0.051}$	0.481 ± 0.235	0.367 ± 0.094	0.429 ± 0.186
θ_A	0.570 ± 0.369	0.335 ± 0.260	0.308 ± 0.210	0.351 ± 0.216	0.310 ± 0.187	0.289 ± 0.117	$\mathbf{0.219 \pm 0.071}$	0.343 ± 0.164
θ_R	$\mathbf{0.630 \pm 0.211}$	0.829 ± 0.334	0.871 ± 0.254	0.643 ± 0.494	0.849 ± 0.214	0.756 ± 0.641	1.033 ± 0.113	0.885 ± 0.721
Mean	0.677 ± 0.069	0.500 ± 0.117	0.580 ± 0.102	0.342 ± 0.151	0.475 ± 0.123	0.346 ± 0.167	0.393 ± 0.039	$\mathbf{0.341 \pm 0.156}$

Table 5: $\text{MAE}_{\%}$ for inferring parameters of an observed time series generated with the true parameters specified in Table 4 for varying training set sizes. Values show error in each parameter for sequentially generated and uniformly sampled training data sets, and are means ± standard deviations for five sequentially generated data sets.

Parameter Inference $\text{MAE}_{\%}$ for Random Genetic Oscillator Time Series

	$N = 2 \cdot 10^4$		$N = 6 \cdot 10^4$		$N = 10^5$		$N = 2 \cdot 10^5$	
Parameter	Uniform	Sequential	Uniform	Sequential	Uniform	Sequential	Uniform	Sequential
α_A	0.708 ± 0.098	0.584 ± 0.065	0.560 ± 0.084	0.544 ± 0.076	0.495 ± 0.044	0.528 ± 0.024	$\mathbf{0.449 \pm 0.021}$	0.538 ± 0.038
α'_A	0.811 ± 0.076	0.702 ± 0.080	0.664 ± 0.071	0.641 ± 0.065	0.597 ± 0.061	0.633 ± 0.031	$\mathbf{0.537 \pm 0.033}$	0.631 ± 0.026
α_R	1.001 ± 0.020	0.953 ± 0.023	0.944 ± 0.031	0.903 ± 0.027	0.912 ± 0.030	0.895 ± 0.023	$\mathbf{0.849 \pm 0.021}$	0.891 ± 0.019
α'_R	0.961 ± 0.031	0.908 ± 0.028	0.902 ± 0.031	0.888 ± 0.017	0.874 ± 0.024	0.891 ± 0.013	$\mathbf{0.854 \pm 0.005}$	0.887 ± 0.012
β_A	0.796 ± 0.059	0.716 ± 0.054	0.686 ± 0.060	0.658 ± 0.057	0.629 ± 0.055	0.657 ± 0.030	$\mathbf{0.564 \pm 0.012}$	0.646 ± 0.025
β_R	0.877 ± 0.069	0.755 ± 0.058	0.747 ± 0.091	0.710 ± 0.049	0.689 ± 0.055	0.700 ± 0.021	$\mathbf{0.629 \pm 0.021}$	0.698 ± 0.030
δ_{M_A}	0.827 ± 0.055	0.703 ± 0.049	0.676 ± 0.061	0.660 ± 0.046	0.623 ± 0.041	0.652 ± 0.018	$\mathbf{0.577 \pm 0.015}$	0.643 ± 0.015
δ_{M_R}	0.759 ± 0.057	0.593 ± 0.080	0.576 ± 0.092	0.512 ± 0.072	0.499 ± 0.076	0.511 ± 0.030	$\mathbf{0.430 \pm 0.022}$	0.516 ± 0.029
δ_A	0.584 ± 0.109	0.441 ± 0.077	0.418 ± 0.051	0.386 ± 0.068	0.372 ± 0.054	0.376 ± 0.018	$\mathbf{0.317 \pm 0.012}$	0.380 ± 0.018
δ_R	0.596 ± 0.135	0.426 ± 0.093	0.406 ± 0.083	0.373 ± 0.085	0.335 ± 0.055	0.360 ± 0.032	$\mathbf{0.272 \pm 0.012}$	0.359 ± 0.032
γ_A	0.972 ± 0.031	0.916 ± 0.024	0.904 ± 0.032	0.890 ± 0.023	0.878 ± 0.023	0.893 ± 0.012	$\mathbf{0.842 \pm 0.012}$	0.890 ± 0.013
γ_R	0.990 ± 0.017	0.946 ± 0.023	0.932 ± 0.039	0.896 ± 0.030	0.894 ± 0.037	0.894 ± 0.019	$\mathbf{0.824 \pm 0.016}$	0.889 ± 0.021
γ_C	0.867 ± 0.129	0.643 ± 0.105	0.617 ± 0.117	0.534 ± 0.105	0.502 ± 0.078	0.508 ± 0.053	$\mathbf{0.420 \pm 0.016}$	0.486 ± 0.046
θ_A	0.876 ± 0.075	0.733 ± 0.062	0.718 ± 0.066	0.694 ± 0.049	0.652 ± 0.042	0.687 ± 0.019	$\mathbf{0.606 \pm 0.027}$	0.684 ± 0.019
θ_R	0.999 ± 0.021	0.945 ± 0.026	0.938 ± 0.038	0.896 ± 0.026	0.891 ± 0.039	0.890 ± 0.015	$\mathbf{0.832 \pm 0.019}$	0.883 ± 0.017
Mean	0.842 ± 0.060	0.731 ± 0.054	0.713 ± 0.060	0.679 ± 0.052	0.656 ± 0.046	0.672 ± 0.019	$\mathbf{0.600 \pm 0.012}$	0.668 ± 0.018

Table 6: $\text{MAE}_{\%}$ for inferring parameters of an time series generated with random parameters for varying training set sizes. Values show error in each parameter for sequentially generated and uniformly sampled training data sets, and are means ± standard deviations for five sequentially generated data sets.



(a) The $MAE_{\%}$ for random testing data

(b) The $MAE_{\%}$ for observed data

Figure 10: $MAE_{\%}$ for random testing data (a) and the observed time series (b) for parameter inference using the CNN with sequentially and randomly generated samples for Genetic Oscillator. Error bars show ± standard deviation.

6 Conclusions

This report presents a sequential sampling algorithm designed to improve the inference accuracy and data-efficiency for likelihood-free parameter inference with neural networks. The algorithm has been tested using a CNN based inference model but can also be used with other learning-based inference techniques. For all three test problems, the improved efficiency in the inference stage outweighs the cost of the sequential sampling by reducing the overall samples needed. The effect is amplified when partially unstable problems and computationally expensive simulation problems are considered. Consequently, the efficiency is expected to improve with model complexity, although the technique needs to be tested on additional problems before any such conclusions can be made. While inference for a specific observation is improved, it is important to note that performance is worse for remaining parts of the parameter space, since we focus more on specific regions. If a more general model is desired, the sequential sampling approach used in this report could be tweaked to reward specific behaviours instead of searching for maximal similarity.

Another subject not explored extensively in this report is on which types of problems the model is most effective. To examine this, additional high-dimensional and unstable models need to be considered. Further testing also must be performed to examine how the choice of summary statistics affects or aggregating multiple observations the performance. Rather than using the same manually selected set of summary statistics for each problem, automatic summary statistic selection methods can tailor the choice of summary statistics to the considered problem. With further testing and mentioned improvements, our approach may have the potential of greatly increasing data efficiency and inference accuracy of current state of the art solutions for likelihood-free parameter inference.

7 References

[1] In Jae Myung. Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*, 47(1):90–100, 2003.

[2] Katalin Csilléry, Michael G.B. Blum, Oscar E. Gaggiotti, and Olivier François. Approximate bayesian computation (abc) in practice. *Trends in Ecology & Evolution*, 25(7):410–418, 2010.

[3] Nelson J. R. Fagundes, Nicolas Ray, Mark Beaumont, Samuel Neuenschwander, Francisco M. Salzano, Sandro L. Bonatto, and Laurent Excoffier. Statistical evaluation of alternative models of human evolution. *Proceedings of the National Academy of Sciences*, 104(45):17614–17619, 2007.

[4] Franck Jabot and Jérôme Chave. Inferring the parameters of the neutral theory of biodiversity using phylogenetic information and implications for tropical forests. *Ecology Letters*, 12(3):239–248, 2009.

[5] Joël Akeret, Alexandre Refregier, Adam Amara, Sebastian Seehars, and Caspar Hasner. Approximate bayesian computation for forward modeling in cosmology. *Journal of Cosmology and Astroparticle Physics*, 2015(08):043–043, aug 2015.

[6] Mattias Åkesson, Prashant Singh, Fredrik Wrede, and Andreas Hellander. Convolutional neural networks as summary statistics for approximate bayesian computation, 2021.

[7] Wing Wong, Bai Jiang, Tung-yu Wu, and Charles Zheng. Learning summary statistic for approximate bayesian computation via deep neural network. *Statistica Sinica*, 2018.

[8] Samuel Wiqvist, Pierre-Alexandre Mattei, Umberto Picchini, and Jes Frellsen. Partially exchangeable networks and architectures for learning summary statistics in approximate bayesian computation, 2019.

[9] Paul Fearnhead and Dennis Prangle. Constructing summary statistics for approximate bayesian computation: semi-automatic approximate bayesian computation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(3):419–474, 2012.

[10] Alfonso Delgado-Bonal and Alexander Marshak. Approximate entropy and sample entropy: A comprehensive tutorial. *Entropy*, 21(6), 2019.

[11] Steve Pincus. Approximate entropy as a measure of system complexity. *Proceedings of the National Academy of Sciences of the United States of America*, 88:2297–301, 04 1991.

[12] Joshua S. Richman and J. Randall Moorman. Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology*, 278(6):H2039–H2049, 2000. PMID: 10843903.

[13] John A. Gubner. *Probability and Random Processes for Electrical and Computer Engineers*. Cambridge University Press, 2006.

[14] Bo Hjorth. Eeg analysis based on time domain properties. *Electroencephalography and Clinical Neurophysiology*, 29(3):306–310, 1970.

[15] K.-I. Goh and A.-L. Barabási. Burstiness and memory in complex systems. *EPL (Europhysics Letters)*, 81(4):48002, jan 2008.

[16] Dennis Prangle. Summary statistics in approximate bayesian computation, 2015.

[17] Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, and Thomas B. Schön. *Machine Learning - A First Course for Engineers and Scientists*. Cambridge University Press, 2022.

[18] Mohammad K Sadoughi, Chao Hu, Cameron A MacKenzie, Amin Toghi Eshghi, and Soobum Lee. Sequential exploration-exploitation with dynamic trade-off for efficient reliability analysis of complex engineered systems. *Structural and Multidisciplinary Optimization*, 57(1):235–250, 2018.

[19] Karel Crombecq, Dirk Gorissen, Dirk Deschrijver, and Tom Dhaene. A novel hybrid sequential design strategy for global surrogate modeling of computer experiments. *SIAM J. Scientific Computing*, 33:1948–1974, 01 2011.

[20] Joachim van der Herten, Dirk Deschrijver, and Tom Dhaene. Fuzzy local linear approximation-based sequential design. In *2014 IEEE Symposium on Computational Intelligence for Engineering Solutions (CIES)*, pages 17–21, 2014.

[21] Dennis Prangle. Adapting the abc distance function, 2015.

[22] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[23] Piotr Indyk. Nearest neighbors in high-dimensional spaces. 2004.

[24] Prashant Singh, Fredrik Wrede, and Andreas Hellander. Scalable machine learning-assisted model exploration and inference using Sciope. *Bioinformatics*, 37(2):279–281, 07 2020.

[25] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

[26] José M. G. Vilar, Hao Yuan Kueh, Naama Barkai, and Stanislas Leibler. Mechanisms of noise-resistance in genetic oscillators. *Proceedings of the National Academy of Sciences*, 99(9):5988–5992, 2002.

[27] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, sep 1975.

[28] Thomas Grandits, Alexander Effland, Thomas Pock, Rolf Krause, Gernot Plank, and Simone Pezzuto. GEASI: Geodesic-based Earliest Activation Sites Identification in cardiac models. *arXiv:2102.09962 [cs, math]*, February 2021. arXiv: 2102.09962.