Eötvös Loránd University

Faculty Of Science

Dávid Kovács

Parameter Estimation of Stochastic Processes with Deep Neural Networks

Thesis

Applied Mathematics MSc

Supervisor:
András Lukács
Department of Computer Science

Budapest, 2023

NYILATKOZAT

Név: Kovács Dávid

ELTE Természettudományi Kar, szak: Alkalmazott matematikus MSc

NEPTUN azonosító: B2W30T

Diplomamunka címe:
Parameter Estimation of Stochastic Processes with Deep Neural Networks

A **diplomamunka** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2023. 06. 05.

Kovács Dávid

a hallgató aláírása

Acknowledgements

I would like to express my gratitude to my supervisor, András Lukács for his continued guidance and involvement.
I would also like to thank the members of the ELTE AI Research Group: Bálint Csanády for conducting several experiments and for making plots and tables, Iván Ivkovic and Dániel Boros for implementing the process generators and the baselines, and Lóránt Nagy for designing stress tests and giving theoretical insights into the baselines.
Last but not least, I would like to thank my mother for her love and support.

Contents

Chapter 1

Introduction

Long-range dependence or long memory has critical importance in the scientific modeling of natural or industrial phenomena. On one hand, in natural sciences, it finds applications in climate change (see e.g. [11]), hydrology (see e.g. [22]), DNA sequencing (see e.g. [34]) and networks (see e.g. [39]). On the other hand, research on long memory implies achievements in financial mathematics, see for example [33, 10] or [3] for the application of long memory in volatility modeling. Nevertheless, the presence of long memory in time series data is a common tenet, turning a great deal of attention to models that are capable of capturing this phenomenon. In most stochastic models the impact of past events on future events has a fast decay, and this way, the effect of observations from the distant past, in terms of forecasting ability, is negligible. When long-range dependence is present in a Gaussian system, predictions concerning the future require information from the complete history of the process - in contrast to Markovian environments, when the most recent events already contain all the information that is necessary for an optimal forecast. When one models data with long memory, it is a crucial task to estimate model parameters, and classical inference methods are often not applicable in the case of long memory processes. We focus our attention on two stochastic processes that are frequently utilized in modern applied mathematics: fractional Brownian motion, and the fractional Ornstein-Uhlenbeck process. The parameter called the Hurst exponent controls the roughness, self-similarity, and the long-range dependence of fractional Brownian motion paths, and this way also influences the characteristics of descendant processes such as the fractional Ornstein-Uhlenbeck process. With the spread of the use of the Hurst exponent, we can highlight such new areas of application as EEG signal processing, detection of epilepsy [1] or in cybersecurity detecting anomalies [25]. Therefore, numerical inference targeting this quantity and other parameters of fractional models is essential.

In this thesis, we examine the estimation of parameters belonging to two processes. In the case of fBm, we mainly focus on the estimation of the Hurst parameter, and in the case of fOU, we mainly concentrate on the estimation of the Hurst and drift parameters. As a side note, we also investigate the estimation of the scale and shift parameters in both cases.

The neural network estimation of the Hurst parameter of the fBm overtakes conventional numerical-statistical estimation mechanisms (see Chap. 3 for baselines). In general, the proposed neural network estimators have higher accuracy and this advantage is uniform on the whole range of the Hurst exponent. Even when Whittle's method produces an accuracy that is in close proximity to the results the neural network yields, the network estimator still has a significant advantage in terms of estimation speed. We measured a kind of consistency of the neural network estimation method, i.e. when trained on longer sequences, the method becomes more accurate, even when inferred on sequences of not necessarily the length used in training.

In the case of the fOU process, there is a severe lack of estimators for all of the parameters. For one thing, conventional estimators produce large errors when estimating the Hurst parameter. We gain advantage over these methods by obtaining neural network estimators that produce significantly higher accuracy. Furthermore, we surpass the single existing baseline for the drift parameter as well. The superiority of the neural network-based estimators is twofold in this case. First, the conventional drift parameter estimator assumes the Hurst, the scale and the shift parameters to be known, which is not a viable assumption in practice. The neural network estimators do not have such prerequisites. Secondly, even when giving the baseline the advantage of knowing its required parameters, the neural network estimation achieves higher accuracy.

In both cases, the estimation of the shift and scale parameters is given lower priority. However, there are some interesting ideas resulting from the estimation of these two parameters. Namely, for the scale parameter, a homogeneous neural network is proposed and the idea of training with constant label is explained. Similarly, in the case of the shift parameter, an unbiased linear neural network is proposed, which only needs to be trained with constant labels as well.

The success of the utilized networks (Sec. 4.3) utmostly stems from a large volume of high-quality training data, manifested with the software that was built around the framework of the so-called isonormal processes (see [31] for the mathematical background, and Sec. 4.2 about the implementation). The underlying path-generating methodology includes the circulant embedding of a covariance matrix and fast Fourier transform. In particular, the system produces fractional Brownian motion paths with great efficiency.

Finally, we conduct stress test with various stochastic processes. These test were designed to gain a better understanding of the inner workings of the neural networks. In some of these tests, the models exhibits behaviour that gives us reason to assume that it learns some general characteristics pertaining to long memory and self-similarity.

This thesis is a result of 2 years of continuous research conducted in the ELTE AI Research Group. As such, parts of it are constituted by the contents of a pending publication. Therefore, for certain elements of it, credit must be given to the coauthors.

What I can claim most credit for is for example, serving as a link between the theoretical and practical parts of the project. To name a , putting theoretical results into a more practice-oriented context, designing a training system that properly utilizes unlimited data and constructing neural network architectures that conform to certain properties.

Chapter 2

Background

In this chapter, we present the processes for which we are looking to construct parameter estimators. We only present the properties of the processes in question that are crucial to know to estimate their parameters.

2.1 The fractional Brownian motion

Here, we briefly introduce the fractional Brownian motion (fBm). For a more in-depth analysis, one may consult [27].

Definition 2.1.1. Let $H \in (0,1], \sigma > 0, \mu \in \mathbb{R}$. The fractional Brownian motion $\left(B_t^H\right)_{t\geq 0}$ is a continuous centered Gaussian process with covariance function

$$cov\left(B_t^H, B_s^H\right) = \frac{1}{2}\left(|t|^{2H} + |s|^{2H} - |t-s|^{2H}\right) .$$

Definition 2.1.2. It is customary to consider the scaled and drifted process $\left(\sigma B_t^H + \mu t\right)_{t\geq 0}$. Let $\mathrm{fBm}(H, \mu, \sigma)$ denote the distribution of this process, which is a distirbution on the Borel σ-algebra of the continuous functions.

Proposition 2.1.1. $\forall c > 0$: if $X \sim \mathrm{fBm}(H, \mu, \sigma)$, then $(X_{ct})_{t\geq 0} \sim \mathrm{fBm}(H, c\mu, c^H\sigma)$

Proof. Let $Z_t = \dfrac{X_{ct} - c\mu \cdot t}{c^H \sigma}$. We will show that $Z \sim \mathrm{fBm}(H, 0, 1)$. To see that, note that it is a centered Gaussian process, so we only have to check its covariance function.

$$\begin{aligned} cov(Z_s, Z_t) &= cov(c^{-H} B_{cs}^H, c^{-H} B_{ct}^H) = c^{-2H} cov(B_{cs}^H, B_{ct}^H) \\ &= c^{-2H} \frac{1}{2}\left(|ct|^{2H} + |cs|^{2H} - |ct - cs|^{2H}\right) \\ &= \frac{1}{2}\left(|t|^{2H} + |s|^{2H} - |t-s|^{2H}\right) = cov(B_s^H, B_t^H), \end{aligned}$$

□

Remark 2.1.1. The above property is called self-similarity when $\mu = 0$. This is in reference to the fact that after a change of time-scale, the resulting process is stochastically similar to the original, i.e., it has the same distribution apart from a scaling parameter.

Remark 2.1.2. Proposition 2.1.1 has important practical implications. In practice, we only have discretized observations from a process $X \sim FBM(H, \mu, \sigma)$. In this thesis, we assume that these observations are equidistant. In other words, we can observe $(X_{j\delta})_{j=0}^{n-1}$, where $\delta > 0$ is the time step and n is the number of observations. However, this is the same as saying we can observe $(Z_j)_{j=0}^{n-1}$ with $Z_t := X_{\delta t}\,(t \geq 0)$. By Proposition 2.1.1, $Z \sim \mathrm{fBm}(H, \delta\mu, \delta^H\sigma)$. In other words, we can assume that we have observations from a "canonical" fBm process, with time step 1. To put it into another perspective, by discretizing an fBm process, μ and σ cease to be well-defined parameters. Only H, $\delta\mu$ and $\delta^H\sigma$ are well-defined.

Proposition 2.1.2. Let $X \sim \mathrm{fBm}(H, \mu, \sigma)$. X has strongly stationary increments. As a consequence, for the marginal distributions of the increments, $X_t - X_s \overset{d}{=} X_{t-s}$ holds.

Proof. As X is Gaussian, all we have to prove is that $cov(X_{t+h} - X_{s+h}, X_t - X_s)$ only depends on $t - s$ and h. To check this, we can write

$$\begin{aligned}
&cov(X_{t+h} - X_{s+h}, X_t - X_s) = \\
&cov(X_{t+h}, X_t) - cov(X_{t+h}, X_s) - cov(X_{s+h}, X_t) + cov(X_{s+h}, X_s) = \\
&\frac{1}{2}(|t - s + h| + |t - s - h| - 2|h|).
\end{aligned}$$

□

Remark 2.1.3. As a consequence, in this work, we will always pre-process discretized fBm observations by differencing them, i.e., apply the transformation $(Z_j)_{j=0}^{n-1} \mapsto (Z_j - Z_{j-1})_{j=1}^{n-1}$. This is advantageous because this way we obtain a stacionary sequence. Also, differencing detrends the observations thus transforms the drift parameter μ into a simple shift parameter.

2.2 The fractional Ornstein-Uhlenbeck process

After the fBm, it is a natural next step to consider the more complicated fractional Ornstein-Uhlenbeck (fOU) process. Beyond the scope of this brief section, a more detailed discussion can be found in [7].

Definition 2.2.1. Let $H \in (0, 1), \alpha, \sigma > 0, \eta, \mu \in \mathbb{R}$. The fractional Ornstein-Uhlenbeck process $(Y_t)_{t\geq 0}$ is the solution of the following stochastic differential equation (SDE):

$$\begin{aligned} dY_t &= -\alpha(Y_t - \mu)\, dt + \sigma\, dB_t^H \\ Y_0 &= \eta. \end{aligned}$$

Let $\mathrm{fOU}(\eta, H, \alpha, \mu, \sigma)$ denote the distribution of this process, which is a distribution on the Borel σ-algebra of the continuous functions.

Proposition 2.2.1. In the above distribution, μ and σ are simply scale and shift parameters. Namely, if $Y \sim \mathrm{fOU}(\eta - \mu)/\sigma, H, \alpha, 0, 1)$, then $\sigma Y + \mu \sim \mathrm{fOU}(\eta, H, \alpha, \mu, \sigma)$.

Proof. We will show that $Z = \sigma Y + \mu$ satisfies the SDE of the $\mathrm{fOU}(\eta, H, \alpha, \mu, \sigma)$.

First, we can write

$$\begin{aligned} dZ_t = d(\sigma Y_t + \mu) = \sigma\, dY_t &= \sigma(\alpha Y_t\, dt + dB_t^H) \\ &= \alpha \cdot \sigma Y_t\, dt + \sigma\, dB_t^H = \alpha(Z_t - \mu)\, dt + \sigma\, dB_t^H. \end{aligned}$$

Secondly, $Z_0 = \sigma Y_0 + \mu = \eta$. □

Proposition 2.2.2. $\forall c > 0$: If $Y \sim \mathrm{fOU}(\eta, H, \alpha, \mu, \sigma)$ then $(Y_{ct})_{t\geq 0} \sim FOU(\eta, H, c\alpha, \mu, c^H\sigma)$.

Proof. Let $Z_t = Y_{ct}$. Let B^H be the fractional Brownian motion that propels Y. We will show that Z satisfies the $\mathrm{fOU}(\eta, H, c\alpha, \mu, c^H\sigma)$ SDE with $W = c^{-H} B_{ct}^H \sim FBM(H, 0, 1)$ as the propelling noise. To see that, we can write

$$\begin{aligned} dZ_t = c \cdot (dY)_{ct} &= c \cdot (-\alpha(Y_{ct} - \mu)\, dt + \sigma\, dB_{ct}^H) \\ &= -c\alpha \cdot (Z_t - \mu) + \sigma c^H\, dW_t \end{aligned}$$

Moreover, trivially, $Z_0 = Y_0 = \eta$.

□

Remark 2.2.1. Note that the above statement does not mean that the fOU is self-similar. After a change of time-scale the resulting process is not stochastically similar to the original process, because it has a different drift parameter.

Remark 2.2.2. The practical implications of Proposition 2.2.2 are similar to that of Proposition 2.1.1. Namely, we can assume the time step to be 1 when discretizing fOU processes. Or again from a different point of view, α, σ and μ cease to be well-defined parameters upon discretization, only H, $\delta\alpha$, $\delta\mu$ and $\delta^H\sigma$ are well-defined.

Chapter 3

Baselines

Before we can establish new, DL-based methods to obtain parameter estimators for the processes in question, we need to consider the statistical methods that are available. We will refer to these methods as baselines and surpassing them will be the focal point of this work.

3.1 Baselines for the fBm

3.1.1 Rescaled range analysis

The rescaled range analysis (R/S) is a historically important estimator for the Hurst parameter (see [15, 22]). For a stationary sequence X, let $Y_k = \sum_{j=0}^{k} X_j$.

$$R/S(n) = \frac{\max_{1\leq k\leq n}\left\{X_k - \frac{k}{n}X_n\right\} - \min_{1\leq k\leq n}\left\{X_k - \frac{k}{n}X_n\right\}}{\sqrt{\frac{1}{n}\sum_{k=1}^{n}\left(Y_k - \frac{1}{n}X_n\right)^2}},$$

If $Y \sim FBM(H,\mu,\sigma)$, then $E(R/S(n)) \sim cn^H$ holds. Thus, the slope coefficient of the linear regression $\log(R/S(n)) \sim \log(m)$ yields an estimate for H.

3.1.2 Higuchi's method

Higuchi's method [19] relies on the computation of the fractal dimension by a one-dimensional box counting. For a sliding box size $b \in \mathbb{N}$ and a starting point $i \in \mathbb{N}, i \leq b$, consider $L_b(i) = \frac{1}{\left[\frac{n-i}{b}\right]} \sum_{k=1}^{\left[\frac{n-i}{b}\right]} \left|X_{i+kb} - X_{i+(k-1)m}\right|$. Then let $L_b := \frac{1}{b}\sum_{i=1}^{b} L_b(i)$. If $X \sim \text{fBm}(H,0,\sigma)$ then $E(L_b) = cb^H$ holds. Thus, the slope coefficient of the linear regression $\log(L_b) \sim \log(b)$ yields an estimate for H.

3.1.3 Whittle's method

Whittle's method [28] maximizes the likelihood function

$$L(X) = (2\pi)^{-n/2} |\Gamma_H|^{-1/2} e^{-\frac{1}{2}X^T\Gamma_H^{-1}X},$$

where $|\Gamma_H|$, Γ_H^{-1} denotes the determinant and the inverse of the matrix Γ_H respectively, and X^T denotes the transpose of the vector X. In practice, however, minimizing the above likelihood proves to be too computationally intensive. Therefore one instead uses a Fourier-analysis-based approximation, see e.g. [4].

3.2 Baselines for the fOU

Parameter estimation for the fOU differs significantly from that of the fBm. Contrary to the fBm, there is a severe lack of estimators for the parameters of the fOU.

3.2.1 Variation Estimators

A generalization of the variogram estimator based on the variogram of order p for a stochastic process with stationary increments is utilized in [14]. The variogram of order p is defined as $\gamma_p(t) = \frac{1}{2}\mathbf{E}|X_i - X_{i+t}|^p$. Notably, when $p = 2$, the variogram is obtained, while for $p = 1$, the madogram is obtained. The case of $p = \frac{1}{2}$ corresponds to the rodogram. In this study, we specifically focus on the case of $p = 1$, where the fractal dimension can be estimated. The fractal dimension is determined using the following formula: $\hat{D}_{V;p} = 2 - \frac{1}{p}\frac{\log \hat{V}_p(\frac{2}{n}) - \log \hat{V}_p(\frac{2}{n})}{\log 2}$. By applying the derived fractal dimension, we can calculate the Hurst exponent (H) as $H = 2 - D$ (see [13]). Moreover, an important estimator belonging to this family is the generalized variation (QGV) estimator, as described in [5].

3.2.2 The Hu-Nualart baselines

For the estimation of fOU parameters σ and α, important results are achieved in [21]. In the paper, a method is described to infer σ with known H. Moreover, another method is described to infer α with known H and σ. We will focus on this later estimator, we will investigate how well it infers α with known H and σ. This way we can gauge its performance without dependence on the H and σ estimators.

Chapter 4

Methods

In this chapter, we describe the framework for attaining the desired parameter estimators.

4.1 Training with unlimited data

In this subsection, we propose a general procedure to attempt to obtain parameter estimators by utilizing generators. Let Θ be the set of the possible parameters and let P be the prior distribution on Θ. Then, $\forall a \in \Theta$: let $G^{(a)}$ generate discretized trajectories from the process distribution Q_a. Now, with $\vartheta \sim P$, consider the compound generator $G := G^{(\vartheta)}$. Suppose we have G as input and we would like to estimate ϑ. Formally, an optimal M estimator would minimize the MSE $E(M(G) - \vartheta)^2$. Now, by having independent copies of (G, ϑ), we can consider the training set $(G_1, \vartheta_1), (G_2, \vartheta_2), (G_3, \vartheta_3), \ldots$. Training a neural network $\mathcal{M}$ on this set with the squared loss function would be a heuristic attempt to obtain the above M estimator.

Training on this set can be done iteratively. Let B be the batch size. Then, in the k^{TH} step of the training process we sample the $[kB, (k+1)B - 1]$ slice of the training set and aggregate it into a batch. We perform the iteration step on this batch, then we go on with the next batch and we never reuse any parts of this batch. Stopping this process is at our discretion, as we never run out of data. During the training process, we are using completely unique, independent training data. This means overfitting is not possible, hence training losses can be treated as validation losses. In practice we can organize our batches into virtual epochs, to monitor the models performance metrics epoch to epoch as the model trains.

We may assume that Q is only parametrized by the target parameter a. This can be done without loss of generality, because if Q is parametrized by other parameters beside a, then we can just randomize those parameters and have Q_a be redefined as the resulting compound distribution.

4.2 Generating processes

To carry out the training process described in the last section, it is indispensable to have fast and accurate generators. In the following, we briefly describe the main ideas behind the generator system we used.

The fBm increment process can be characterized as a zero-mean stationary Gaussian noise. As such, it can be generated using a circular matrix embedding method belonging to the Davies-Harte procedure family. However, the original Davies-Harte method is lacking in speed. Therefore, to obtain a faster generator, Kroese's method [24] was implemented in Python. The resulting Python implementation is approximately an order of magnitude faster than the Python package fbm [8].

Using the generator for the fBm increments, the fOU generator can be implemented by adequately discretizing the stochastic integral it is defined by.

4.3 Neural network architecture

4.3.1 Architecture for estimating H and α

An estimator for H and α should be scale and shift invariant. We can obtain such an $\mathcal{M}$ by setting the first layer of $\mathcal{M}$ to be a standardizing layer. This layer applies the transformation $x \mapsto (x - \overline{x})/\hat{\sigma}(x)$ to each sequence x in the batch separately, where $\hat{\sigma}(x) = \left(\frac{1}{n}\sum_{j=0}^{n-1}(x_j - \overline{x})^2\right)^{1/2}$. The next part of $\mathcal{M}$ is a sequential regressor. We constructed the regressor by first transforming the input sequence into a higher dimensional sequence, after which we average out each dimension of the output sequence and thus obtain a vector, and finally we apply an MLP [17] to attain a scalar output. We considered two options for the sequence transformation. In $\mathcal{M}_{\text{conv}}$, it is achieved by a multilayer 1D convolution [23], and in $\mathcal{M}_{\text{LSTM}}$, by an LSTM [20].

4.3.2 Architecture for estimating σ

We want the estimator for σ to be homogeneous, i.e., $M(\lambda x) = \lambda M(x)$ should hold for every input x and every scalar $\lambda > 0$. One way to achieve homogeneity is considering the previously defined $\mathcal{M}_{\text{conv}}$ and turning off the bias everywhere, and setting every activation to PReLU. Moreover, the fist layer should not standardize completely, only center. That is, the transformation $x \mapsto x - \overline{x}$ should constitute the first layer. Denote the resulting neural network by $\mathcal{M}^{*}_{\text{conv}}$.

4.3.3 Architecture for estimating μ

We want the estimator for μ to be linear. One way to achieve linearity is considering the previously defined $\mathcal{M}^{*}_{\text{conv}}$ and removing the centering layer and turning off the activation functions. Denote the resulting neural network by $\mathcal{M}^{+,*}_{\text{conv}}$. However, when estimating μ, on top of the linearity, it is customary to require the estimator to be unbiased. For a linear sequential estimator M, being unbiased is equivalent to the property $M(\mathbf{1}) = 1$, where $\mathbf{1}$ is the constant 1 sequence. This property can be easily enforced by redefining $\mathcal{M}^{+,*}_{\text{conv}}(x)$ as $\dfrac{\mathcal{M}^{+,*}_{\text{conv}}(x)}{\mathcal{M}^{+,*}_{\text{conv}}(\mathbf{1})}$. In the remainder of this thesis, the notation $\mathcal{M}^{*}_{\text{conv}}$ will refer to the unbiased version.

4.3.4 Hyperparameters

We found that unless $\mathcal{M}$ is severely underparametrized, the specific hyperparameter configuration does not have a significant effect on its performance. Slight differences can arise in the speed of convergence, but these are not relevant due to the unlimited data and fast generators. The following are the hyperparameters that we used in our experiments.

In $\mathcal{M}_{\text{conv}}$, we used a 1D convolution with 6 layers. The input has 1 channels, and the convolution layers have output channels sizes of 64, 64, 128, 128, 128, and 128. Every layer has stride 1, kernel size 4 and no padding. The activation function is PReLU after every layer. In M_{LSTM}, we used an unidirectional LSTM with two layers, its input dimension is 1, the dimension of its inner representation is 128. In both models, we use an MLP with 2 layers with input dimension of 128 and output dimensions of 64 and 1. The activation function is PReLU between the two layers. Moreover, $\mathcal{M}^{*}_{\text{conv}}$ and $\mathcal{M}^{+,*}_{\text{conv}}$ have the same structure as $\mathcal{M}_{\text{conv}}$ except for the properties already discussed in the previous subsection.

For training the models, we used AdamW optimization for the MSE loss function with default parameters as described in [26]. We used a train (and validation) batch size of 32.

4.4 Technical details

The process generators were implemented in Python [37], using Numpy [16] and SciPy [38]. We imported Higuchi's method from the package AntroPy [36]. The R/S method was imported from the package hurst [29]. The framework responsible for the training process was implemented in Pytorch [32]. Every neural module we used was readily available in Pytorch. We managed our experiments using the experiment tracker neptune.ai [30].

The neural models were trained on Nvidia RTX 2080 Ti graphics cards. Training took approximately one GPU hour to one GPU day per model, depending on the type of process, the applied architecture, and on the length of sequences used for training. A shorter training time can mostly be expected from the acceleration (parallelization) of the sequence generation.

Chapter 5

Experiments

With the necessary foundation being made, we can now present the training and experiments we conducted to obtain and evaluate neural network-based parameter estimators.

5.1 Metrics

First, we briefly describe the metrics used throughout this chapter. Notation from Section 4.1 will be used. Moreover, let F be the cumulative distribution function and F^{-1} be the quantile function of P. Let N be the number of realizations.

Definition 5.1.1. Let the empirical mean squared error (MSE) be defined as $\sum_{j=0}^{N-1}(M(G_j) - \vartheta_j)^2$.

Definition 5.1.2. Let $m_\epsilon(t) = \dfrac{\sum_{j=1}^{N} G_j \mathbb{1}_{[t-\epsilon,t+\epsilon]}(F(\vartheta_j))}{\sum_{j=1}^{N} \mathbb{1}_{[t-\epsilon,t+\epsilon]]}(F(\vartheta_j))}$. Then, let the empirical bias function $\hat{b}_\epsilon : [0,1] \to \mathbb{R}$ be defined as $\hat{b}_\epsilon(t) = m_\epsilon(t) - F^{-1}(t)$.

In other words, we average out $M(G_j) - \vartheta_j$ for $F(\vartheta_j) \in [t-\epsilon, t+\epsilon]$, where $\epsilon > 0$ is the sliding window radius.

Definition 5.1.3. Let the empirical standard deviation function $\hat{s}_\epsilon : [0,1] \to \mathbb{R}$ be defined as $\hat{s}_\epsilon(t) = \left(\dfrac{\sum_{j=1}^{N} \left(G_j - m_\epsilon(t)\right)^2 \mathbb{1}_{[t-\epsilon,t+\epsilon]}(F(\vartheta_j))}{\sum_{j=1}^{N} \mathbb{1}_{[t-\epsilon,t+\epsilon]]}(F(\vartheta_j))} \right)^{1/2}$.

Remark 5.1.1. The above use of F is convenient because we can plot the metrics in question as functions of the quantiles. This is especially useful when P is not uniform. However, when P is uniform, we will usually just plot as functions of the parameter values itself.

Let K be a suitably large number. We may assume $K = \lceil 1/\epsilon \rceil$, unless otherwise specified.

Definition 5.1.4. Let the absolute area under the empirical bias curve be defined as $\dfrac{|\hat{b}_\epsilon(0)| + |\hat{b}_\epsilon(1)| + 2\sum_{j=0}^{K} |\hat{b}_\epsilon(j/K)|}{2K}$.

Definition 5.1.5. Let the absolute area under the empirical standard deviation curve be defined as $\dfrac{|\hat{s}_\epsilon(0)| + |\hat{s}_\epsilon(1)| + 2\sum_{j=0}^{K} |\hat{s}_\epsilon(j/K)|}{2K}$.

5.2 Results for the fBm

5.2.1 Results for fBM H

In the following, one epoch will simply refer to a block of 10^5 unique training input. We trained $\mathcal{M}_{\text{conv}}$ and $\mathcal{M}_{\text{LSTM}}$ on different sequence lengths then compared their performance to that of the baselines. In every case, the training and evaluation was done on fBm$(H, 0, 1)$ increments with $H \sim U(0,1)$. Which means we fixed $\mu = 0$ and $\sigma = 1$. The reason for this is $\mathcal{M}_{\text{conv}}$ and $\mathcal{M}_{\text{LSTM}}$ are shift and scale invariant, as such they are indifferent to μ and σ.

The first training phase lasted for 200 epochs on sequences of length $n = 100$. At this point, the models were evaluated on this later sequence length. Then fine-tuning ensued on sequence lengths of $n = 200, 400, \ldots, 12800$ for 10 epochs each. The models were evaluated on each one of the sequence lengths right after being trained on the same length. The comparison between these fine-tuned best-case neural networks and the baselines are seen in Tables 5.1, 5.2, 5.3. However, as these tables only contain aggregate metrics, they do not contain information on Hurst parameter specific performance. For this reason the results for $n = 12800$ are visualized in Figures 5.2, 5.1. These show that the models are uniformly superior to the baselines on the entire Hurst parameter set $(0,1)$.

However, some may find using different models for different sequence lengths inconvenient. Therefore, it might be important to construct an estimator that can be used on every length. A candidate for such an estimator could be the neural network that was fine-tuned on the longest sequences, i.e., on $n = 12800$. The evaluation of this model can be seen in Table 5.4. Moreover, after the fine-tuning experiments, we continued to train $\mathcal{M}_{\text{LSTM}}$ on random $n \sim U(\{100, 101, \ldots, 12800\})$ sequence length for 50 epochs. This was another attempt to obtain a single neural network that could be used for every sequence length. The evaluation of this model can be seen in 5.5. As one would expect, in both cases, the MSE is slightly worse for every length than the length specific best-case scenario. However, what is interesting is that this slightly inferior precision is mainly caused by the bias rather than the variance. All in all, the model that was fine-tuned for $n = 12800$ performs better than the one trained on random length due to it having smaller absolute bias. We conducted furher experiments to gain a better understanding on the relation between the training and evaluation sequence length. The results of these experiments can be seen in Table 5.6 and they are visualized in Figures 5.3, 5.4.

Table 5.1: MSE losses of different fBm Hurst-estimators by sequence length.

	MSE loss ($\times 10^{-3}$)					
seq. len.	R/S	variogram	Higuchi	Whittle	M_{conv}	M_{LSTM}
100	27.6	9.30	10.6	4.33	4.27	**4.07**
200	18.9	5.05	4.21	2.00	1.99	**1.91**
400	13.9	2.92	1.99	1.00	0.959	**0.917**
800	10.8	1.75	1.05	0.540	0.476	**0.453**
1600	8.62	1.09	0.593	0.324	0.240	**0.224**
3200	6.74	0.724	0.360	0.225	0.122	**0.114**
6400	5.57	0.502	0.229	0.179	0.0628	**0.0579**
12800	4.70	0.365	0.155	0.157	0.0333	**0.0297**

Table 5.2: Absolute area under Hurst-bias curve of different fBm Hurst-estimators by sequence length.

	$\hat{b}_{0.025}$ ($\times 10^{-3}$)					
seq. len.	R/S	variogram	Higuchi	Whittle	M_{conv}	M_{LSTM}
100	116	34.9	58.1	**10.7**	12.2	11.3
200	98.0	26.5	27.7	6.84	5.54	**5.24**
400	87.1	20.7	14.5	5.79	2.63	**2.58**
800	79.1	17.0	7.82	5.03	1.40	**1.32**
1600	71.4	13.9	4.63	4.90	0.765	**0.656**
3200	62.6	11.8	2.89	4.88	0.411	**0.403**
6400	58.0	9.79	1.74	5.03	0.241	**0.211**
12800	53.7	8.34	1.23	5.04	0.140	**0.131**

Table 5.3: Absolute area under Hurst - empirical standard deviation curve of different fBm Hurst-estimators by sequence length.

	$\hat{s}_{0.025}$ ($\times 10^{-2}$)					
seq. len.	R/S	variogram	Higuchi	Whittle	M_{conv}	M_{LSTM}
100	9.62	8.67	8.25	6.25	6.16	**6.01**
200	7.35	6.33	5.65	4.30	4.27	**4.15**
400	5.69	4.78	4.04	3.01	3.00	**2.91**
800	4.58	3.61	2.97	2.11	2.12	**2.05**
1600	3.71	2.74	2.22	1.50	1.51	**1.46**
3200	3.34	2.14	1.67	1.09	1.08	**1.04**
6400	2.76	1.67	1.28	0.791	0.775	**0.743**
12800	2.34	1.34	0.998	0.590	0.564	**0.534**

Table 5.4: Metrics of $\mathcal{M}_{\text{LSTM}}$ on sequences of different length after being fine-tuned on length $n = 12800$.

	Metrics of M_{LSTM} ($\times 10^{-3}$)		
seq. len.	MSE	$\hat{b}_{0.025}$	$\hat{s}_{0.025}$
100	4.4	17	62
200	2.0	7.3	43
400	0.94	3.1	30
800	0.46	3.9	21
1600	0.22	1.4	15
3200	0.11	1.1	10
6400	0.059	1.4	7.3
12800	0.030	1.6	5.2

Table 5.5: Metrics of $\mathcal{M}_{\text{LSTM}}$ on sequences of different length after being trained on random length.

	Metrics of M_{LSTM} ($\times 10^{-3}$)		
seq. len.	MSE	$\hat{b}_{0.025}$	$\hat{s}_{0.025}$
100	4.4	16	62
200	2.0	8.1	42
400	0.96	5.1	30
800	0.47	3.9	21
1600	0.24	3.7	15
3200	0.13	3.6	10
6400	0.069	3.3	7.2
12800	0.039	3.1	5.2

Table 5.6: MSE losses of LSTM-based models trained on different sequence lengths.

	MSE loss by validation seq. len. ($\times 10^{-3}$)						
train seq. len.	100	200	400	800	1600	3200	6400
100	4.14	2.17	1.41	1.09	0.962	0.918	0.915
200	4.21	1.88	0.947	0.528	0.344	0.264	0.231
400	4.78	2.02	0.940	0.477	0.281	0.196	0.161
800	4.80	2.00	0.913	0.443	0.230	0.134	0.0888
1600	5.01	2.11	0.952	0.447	0.220	0.113	0.0617
3200	5.44	2.23	0.972	0.454	0.221	0.111	0.0608
6400	5.59	2.30	1.01	0.471	0.229	0.121	0.0692

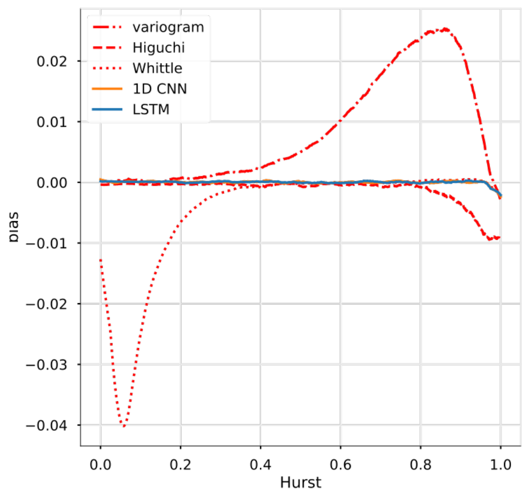

Figure 5.1: Empirical bias $b_{0.025}$ of the fBm estimators by Hurst value. Measured on sequences of length 12800. The estimator R/S has a bias from 0.125 to -0.075 roughly linearly when the Hurst exponent changes from 0 to 1.

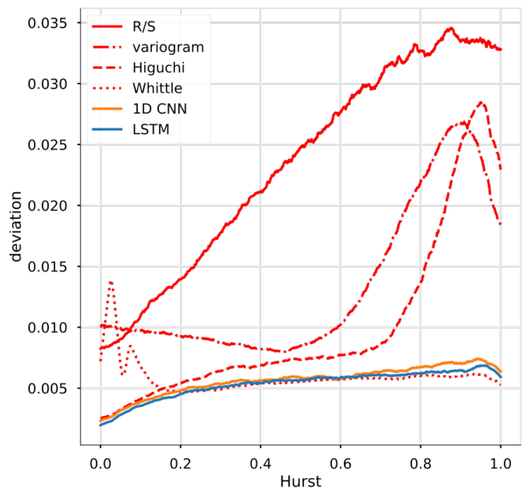

Figure 5.2: Empirical standard deviation $\hat{s}_{0.025}$ of the fBm estimators by Hurst value. Measured on sequences of length 12800.

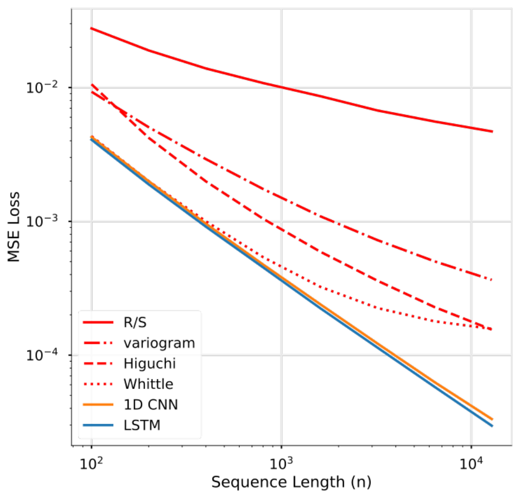

Figure 5.3: Empirical consistency of the baseline estimators and the fine-tuned neural models.

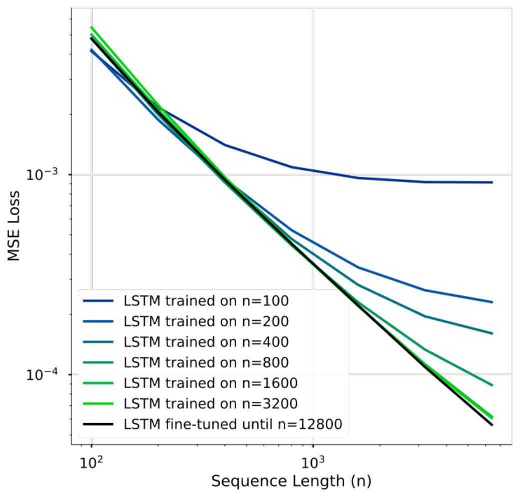

Figure 5.4: The empirical consistency of the different dedicated LSTM models.

5.2.2 Results for fBM σ

We briefly present the results achieved for the estimation of the scale parameter of the fBm. We used the incremental fine-tuning approach described in the previous subsection to train $\mathcal{M}^*_{\text{conv}}$ for the estimation of σ. It should be noted that just as in the previous subsection, the inputs are fBm$(H, 0, 1)$ increments with $H \sim U(0, 1)$ being random. This means that during the training process every label is $\sigma = 1$. However, $\mathcal{M}^*_{\text{conv}}$ cannot abuse this fact, i.e., it cannot "cheat" and just learn the constant 1 function, because it is homogeneous. Therefore, by also taking in account the shift invariance of $\mathcal{M}^*_{\text{conv}}$, when it learns to infer 1 on fBm$(H,0,1)$ inputs, it actually learns to infer σ on fBm(H,μ,σ), $(\sigma > 0,\ \mu > 0)$ inputs. As another consequence of the the homogeneity of $\mathcal{M}^{*}_{\text{conv}}$, there is no need for the metrics described in Subsection 5.1. The MSE, bias and standard deviation for $\sigma = 1$ determine these metrics for every $\sigma > 0$. Therefore, in Table 5.7, we only present the regular bias $\hat{b}$ and standard deviation $\hat{s}$ for $\sigma = 1$. The MSE can be calculated from these. As a baseline, we use the simple $\hat{\sigma}(x) = \left(\frac{1}{n}\sum_{j=0}^{n-1}(x_j - \overline{x})^2\right)^{1/2}$. As we can see, the model significantly outperforms $\hat{\sigma}$. Finding more sophisticated baseline methods for σ might be subject of further research.

Table 5.7: Performance metrics of M^{*}_{conv} and $\hat{\sigma}$ by sequence length for $\sigma = 1$.

	$\hat{b}$ $(\times 10^{-3})$		$\hat{s}$ $(\times 10^{-2})$	
seq. len.	$\hat{\sigma}$	M^{*}_{conv}	$\hat{\sigma}$	M^{*}_{conv}
100	70	24	16	15.4
200	59	15.8	15	12.5
400	54	11.8	14	12.7
800	45	8.8	12.2	9.3
1600	43	6.1	11.8	7.8
3200	37	4.7	11.6	7.7
6400	35	3.5	10.8	5.9
12800	33	3.1	10.6	5.6

5.2.3 Results for fBM μ

We also briefly present the results achieved for the estimation of the shape parameter of the fBm. Again, we used the previous incremental fine-tuning approach. The input is still fBm$(H,0,1)$ with $H \sim U(0,1)$ being random. This means that during the training process every label is $\mu = 0$. However, $\mathcal{M}^{+;*}_{\text{conv}}$ cannot just learn the constant 0 function, because it is enforced to be unbiased. This, and the linearity together make it so that learning to infer 0 on fBm$(H,0,1)$ is equivalent to learning to infer μ on fBm(H,μ,σ), $(\mu > 0, \sigma > 0)$. Keeping all this in mind, the only performance metric needed to evaluate $\mathcal{M}^{+;*}_{\text{conv}}$ is the MSE for $\mu = 0$ and $\sigma = 1$. This is also true for the baseline $\hat{\mu}(x) = \overline{x}$. The comparison between the baseline and the neural network can be seen in Table 5.8. As we can see, the model does not have a clear edge over $\overline{x}$. Therefore $\overline{x}$ might be preferable in this case, because it is simple and well-understood. Further inquiry could be carried out into relaxing the linearity and unbiasedness restrictions of $\mathcal{M}^{+;*}_{\text{conv}}$ and finding more sophisticated baselines.

Table 5.8: MSE losses of different fBm μ estimators by sequence length.

	MSE loss ($\times 10^{-2}$)	
seq. len.	$\overline{x}$	$M_{\text{conv}}^{+,*}$
100	11.4	11.1
200	9.44	9.45
400	8.25	8.28
800	7.54	7.41
1600	6.76	6.93
3200	6.15	6.16
6400	5.93	5.60
12800	5.53	5.34

5.3 Results for the fOU

5.3.1 Results for fOU H

We trained M_{LSTM} with the incremental training technique on fOU$(\eta, H, \alpha, 0, 1)$ inputs with $H \sim U(0,1), \alpha \sim Exp(100), \eta \sim N(0,1)$. The reason for fixing $\mu = 0$ and $\sigma = 1$ is the same as in the case of the fBm. However, additional consideration is needed for η. By setting $\eta \sim N(0,1)$ in the standard $\mu = 0$, $\sigma = 1$ case we assume that $\eta \sim N(\mu, \sigma)$ holds in the scaled and shifted case. Heuristically, this is equivalent to assuming that the sequnce starts at a "typical value". If one finds this assumption impractical, then η can be sampled from a heavy-tailed distribution.

With parameters yielded by these same distributions, we compared the performance of the model and the QGV baseline. As specified above, an exponential distribution with high rate is yielding the parameter α. The reason for such a high rate (thus small scale) is that the QGV method is only consistent for small values of α and we wanted to compare the model to the baseline on equal grounds. This comparison can be seen in Table 5.9. As we can see, M_{LSTM} has a very siginificant advantage over the QGV method.

Table 5.9: Performance metrics of different fOU Hurst estimators by sequence length.

	MSE loss ($\times 10^{-3}$)		$\hat{b}_{0.025}$ ($\times 10^{-3}$)		$\hat{s}_{0.025}$ ($\times 10^{-2}$)	
seq. len.	QGV	M_{LSTM}	QGV	M_{LSTM}	QGV	M_{LSTM}
100	41.0	3.38	106	9.26	9.86	5.53
200	34.2	1.74	97.1	4.84	8.07	4.00
400	29.4	0.919	86.4	2.59	6.92	2.92
800	25.0	0.494	76.2	1.52	5.88	2.15
1600	20.6	0.269	65.1	0.827	5.09	1.59
3200	16.3	0.149	53.8	0.575	4.37	1.18
6400	12.6	0.0810	43.7	1.95	3.68	0.842

5.3.2 Results for fOU α

Estimating parameter α is the most challenging part of this work. The biggest challenge lies in designing an experiment that can compare M_{LSTM} to the Hu-Nualart baseline on equal grounds. One has to adequately choose a distribution of α for the training process of the model and another distribution for the comparison of the model and the baseline. For the sake of simplicity, restrict ourselves to uniform distributions $U(0,a)$ and $U(0,b)$ for the training and evaluation respectively. Naturally, $a \geq b > 0$ should hold. However, if $a >> b$ then the model has to learn to estimate large α values, which are much harder to learn than smaller values, thus it might be forced to learn the average $a/2$ and it would only slightly deviate from that to avoid large errors. In this case, the model would be at a severe disadvantage if we were to evaluate it with $\alpha \sim U(0,b)$. On the other hand, if $a = b$, then the model has an advantage. As a middle ground, we chose $a = 0.2$ and $b = 0.1$. All in all, the model was trained and evaluated on $\text{fOU}(\eta, H, \alpha, 0, 1)$ inputs with $\eta \sim N(0,1), H \sim U(0,1)$, $\alpha \sim (0, 0.2)$ during the training and $\eta \sim N(0,1), H \sim U(0.1,1)$, $\alpha \sim (0, 0.1)$ during the evaluation. The change in the Hurst interval was done because smaller values of H caused numerical instability in the baseline.

Moreover, it is important to note that the baseline had the additional advantage of knowing H, σ and μ. With these in mind, the comparison between the model and the baseline can be seen in Table 5.10. The baseline seems to have smaller bias and the model smaller standard deviation. However, in this case, the standard deviation is the metric that dominates, therefore the model performs better.

Table 5.10: Performance metrics of different fOU α estimators by sequence length. The model was trained with $\alpha \sim U(0, 0.2)$, the evaluation was done with $\alpha \sim U(0, 0.1)$.

	MSE loss ($\times 10^{-4}$)		$\hat{b}_{0.025}$ ($\times 10^{-4}$)		$\hat{s}_{0.025}$ ($\times 10^{-4}$)	
seq. len.	Hu-Nualart	M_{LSTM}	Hu-Nualart	M_{LSTM}	Hu-Nualart	M_{LSTM}
100	43.9	18.8	26.5	28.35	58.9	29.62
200	16.8	13.5	12.7	20.4	37.25	30.07
400	9.43	8.10	5.40	14.0	27.0	24.1
800	5.90	4.23	2.61	7.31	21.6	18.19
1600	3.56	2.17	2.00	4.10	16.8	12.9
3200	3.34	1.03	1.98	3.35	15.8	8.89
6400	2.36	0.440	2.16	0.722	13.4	6.24

We also plotted the MSE curve of the estimators for sequence length $n = 1600$. This can be seen in Figures 5.5, which shows us that the model produces a smaller MSE for the vast majority of values of α.

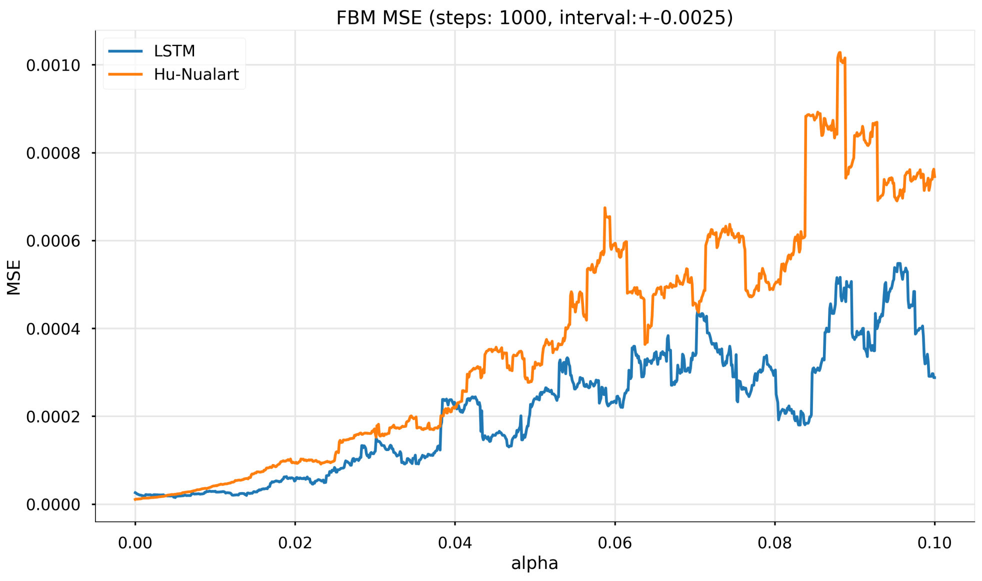

Figure 5.5: Empirical MSE curve of fOU α estimators. Measured on sequences of length 1600.

There are several experiments that could be subject of further research. Most importantly, a better understanding is needed in determining the effect of α on the performance of the estimators.

5.3.3 Results for fOU σ

We can obtain an estimator for fOU σ analogously to the case of fBm σ. Namely, we can train neural network $\mathcal{M}^{*}_{\text{conv}}$ to attain a homogeneous estimator. The evaluation and training is done on fOU$(\eta, H, \alpha, 0, 1)$ processes, with $\eta \sim N(0,1)$, $H \sim U(0,1)$ and $\alpha \sim U(0,1)$. By setting this distribution of α, we have chosen to allow larger values of α compared to the previous sections. Here, we do not compare the model to a baseline, although a potential candidate could be the Hu-Nualart baseline for σ. However, due to time restraints, we did not have enough time to properly implement and test the baseline in question.

With the usual incremental training and evaluation, the performance metrics of M^{*}_{conv} can be seen in Table 5.11. Note that just as in the case of the fBm σ, the evaluation and training only need to be done for $\sigma = 1$.

Table 5.11: Performance metrics of fOU σ estimator M^{*}_{conv} by sequence length.

seq. len.	$\hat{b}$ $(\times - 10^{-2})$	$\hat{s}$ $(\times 10^{-1})$
100	3.26	1.78
200	2.89	1.66
400	2.43	1.52
800	2.25	1.46
1600	1.97	1.36
3200	1.76	1.30

Due to time restrictions, we did not have enough time to fully train the model. For the initial length $n = 100$, it was trained on $1.2 * 10^6$ input sequences. Then, it was incrementally fine-tuned on 10^5 sequences for each of the lengths displayed in the table. Still, the results are promising. Also, it should be kept in mind that here the neural network was further burdened with large α values that would cause numerical instability for conventional estimators.

5.3.4 Results for fOU μ

For the sake of completeness, we also include results for the parameter μ of the fOU. Note that this is the easiest parameter estimation problem discussed in this work. The reason for this is that the fOU process is mean-reverting, and the mean of the process is μ. Therefore even simply averaging the input sequence constitutes a very viable estimation for μ.

We can obtain a neural network-based estimator for this parameter analogously to the case of fBm μ. Namely, we can train neural network $\mathcal{M}^{+;*}_{\text{conv}}$ to attain an unbiased linear estimator. The evaluation and training is done on fOU$(\eta, H, \alpha, 0, 1)$ processes, with $\eta \sim N(0,1)$, $H \sim U(0,1)$ and $\alpha \sim U(0,1)$. With the usual incremental training and evaluation, the performance metrics of $M_{\text{conv}}^{+,*}$ can be seen in Table 5.12. Note that just as in the case of the fBm μ, the evaluation and trainin only needs to be done for $\mu = 0$.

Table 5.12: MSE losses of different fOU μ estimators by sequence length.

	MSE loss ($\times 10^{-4}$)	
seq. len.	$\overline{x}$	$M_{\text{conv}}^{+,*}$
100	21.1	18.5
200	8.61	7.54
400	3.01	3.41
800	1.72	1.45
1600	0.377	0.412
3200	0.0751	0.289

It seems that for larger sequence lengths, $\overline{x}$ becomes superior, whereas for smaller sequence lengths, $M_{\text{conv}}^{+,*}$ has a slight edge.

Chapter 6

Stress-testing the neural estimators

Suppose we have a neural network M that was trained to estimate the Hurst exponent of the fBm. With $\vartheta \sim U(0,1)$ being the random Hurst parameter and $G := G^{(\vartheta)}$ being the corresponding compound generator, our heuristic idea is that $M(x) = E(\vartheta | G = x)$ holds. However, this conditional expectation is difficult to analyze. What happens when the input x is yielded by a process distribution other than the fBm? There are at least three different characteristics of the input distribution that M could approximate: the box dimension of the paths, properties associated with memory, or it could potentially quantify self-similarity. Answering this question will be the focal point of our stress-testing experiments.

6.1 Sum of two fBM: box dimension versus memory

To differentiate between fractal properties of paths (box dimension) and the decay of auto-covariance (memory), let $H_1 < H_2$ be two real numbers in the set $(0, 1/2) \cup (1/2, 1)$, and let $X \sim \mathrm{fBm}(H_1, 0, 1)$ and $Y \sim \mathrm{fBm}(H_2, 0, 1)$ be two independent processes. Consider the process defined by $Z = X + Y$. On one hand, if M captures the asymptotic decay of the autocovariance of the given input, then we have $M(Z) \approx H_2$. On the other hand, contrary to the above case, if M captures the box dimension of the given input, then $M(Z) \approx H_1$. This way the above heuristic reasoning gives a possible method, with which one can test the otherwise unknown behavior of the estimator M, regarding its hidden estimation procedure.

Driven by these motivations, after training M_{LSTM} for the parameter estimation of the fBm, we tested it on the above fBm sums. We considered cases where H_1 was fixed and $H_2 \sim U(0,1)$ was random. The resulting scatter plot can be seen in Figure 6.1. Apparently, M_{LSTM} tends to infer values $\hat{H} \in (H_1, H_2)$. Therefore, it does not seem to learn the box dimension nor the memory but an "in-between" quantity.

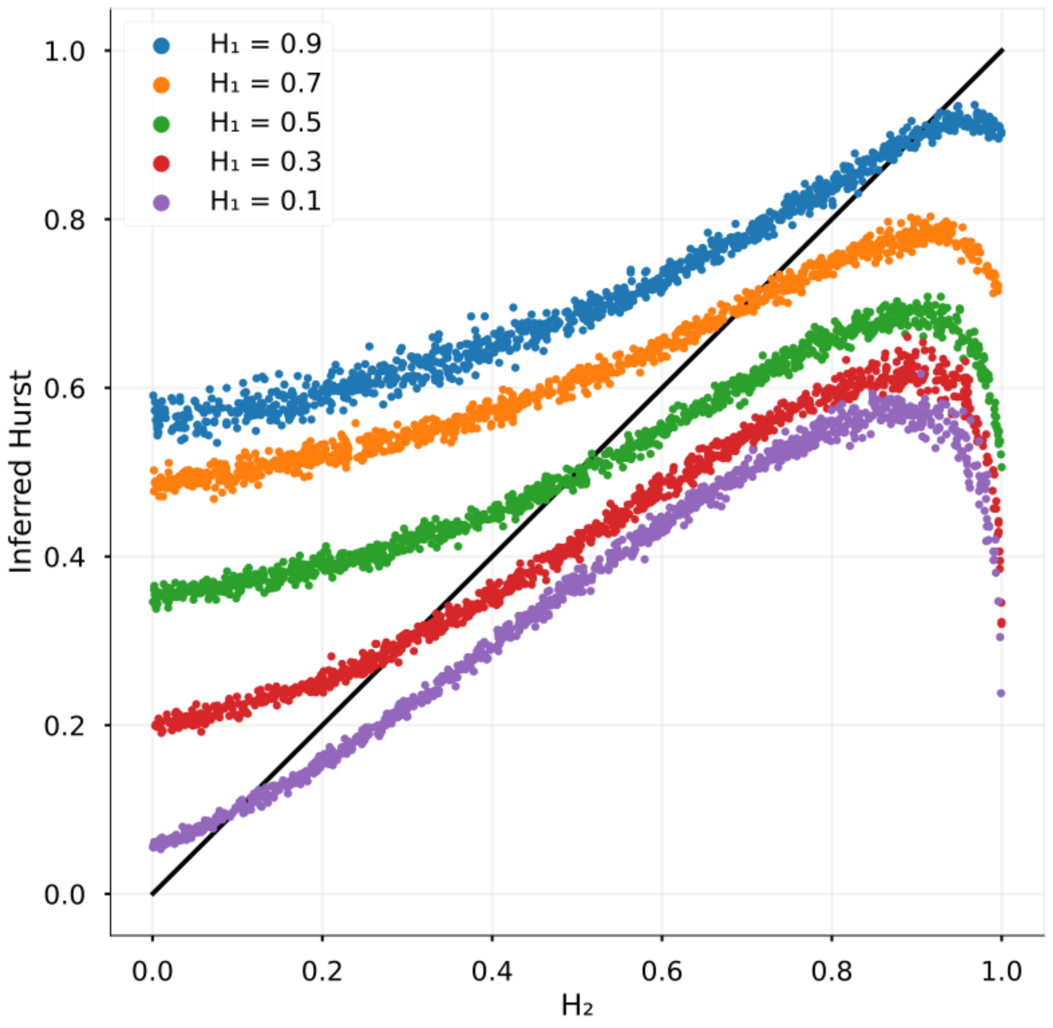

Figure 6.1: Scatterplot of M_{LSTM} on fBm sum inputs and different fixed H_1 values. $n = 6400$. x: H_2, y: H value estimated by M_{LSTM}

6.2 Lévy processes: box dimension versus self-similarity

On one hand, it is a well-known fact that symmetric α-stable Levy processes, for $\alpha \in (0, 2]$, are self-similar with self-similarity exponent $1/\alpha$, and according to [35] the box dimension of such processes, for $\alpha > 1/2$, can be given by the formula $2 - 1/\alpha$. On the other hand, they do not have memory in the sense that increments are independent. This way one can assess if the estimator M, in case of an α-stable Lévy process as input, denoted by D_α, produces inferred values according to the law $M(D_\alpha) \approx \frac{1}{\alpha}$ corresponding to self-similarity of the underlying, or contrary to this, it rather follows a logic that supports evidence that it infers according to the law $M(D_\alpha) \approx 2 - \frac{1}{\alpha}$ corresponding to box dimension.

We tested several models, calibrated to fractional Brownian motion, on α-stable Levy processes. The results can be seen in Figure 6.2. In the particular case, when $\alpha = 2$, the law of an α-stable Levy process coincides with that of a standard Brownian motion, and also with the law of a fractional Brownian motion with Hurst parameter $H = 1/2$. This way the inferred value at $\alpha = 2$, that is $1/2$, is not at all unforeseen. However, when moving away from $\alpha = 2$, the model first displays heavy variance, and when moving close to $\alpha = 0$, we see the concentration of inferred values around some levels — that is, as we anticipate, specific to the unique learning phase of the model used.

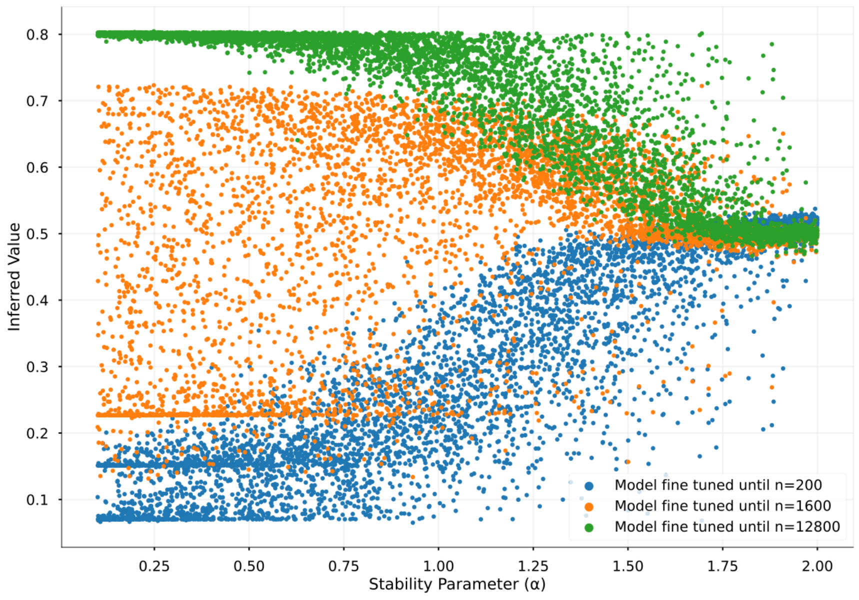

Figure 6.2: Scatterplot of M_{LSTM} models finetuned up to different lengths of fBm sequences inferring on Lévy processes of length 6400 and stability parameter α.

6.3 ARFIMA(0,d,0): an alternative with similar memory

For $d > -1$ we define the fractional difference operator $\nabla^d = \sum_{k=0}^{\infty} \binom{d}{k}(-B)^{-k}$, where B is the backward shift operator, that is $BX_j = X_{j-1}$, and $\binom{d}{k} = \frac{d!}{k!(d-k)!}$. For $d \in (-1/2, 1/2)$ the ARFIMA$(0, d, 0)$ process is defined as the solution of the difference equation $\nabla^d X_j = \zeta_j$, where $\zeta_j,\ j \in \mathbb{Z}$ is a white-noise sequence. It is known that when $\zeta_j,\ j \in \mathbb{Z}$ is ergodic, and $d \neq 0$, there is a unique stationary solution to this differential equation (see Theorem 7.2.2 in [12]). Moreover, according to Theorem 7.2.1 in [12], for the autocovariance function R, we have $R(k) \approx c_d k^{2d-1}$. Thus, in terms of the decay of autocovariance, and memory properties (see Definition 3.1.2 in [12]), the ARFIMA$(0, d, 0)$ process corresponds to a fractional noise with Hurst parameter $H = d + 1/2$, offering a potential way for testing estimators calibrated to fractional Brownian motion or its first order difference.

We tested M_{LSTM} on ARFIMA$(0, d, 0)$ trajectories generated by the Python package arfima [2], and as Figure 6.3 shows, the model performs remarkably — with minor asymmetric bias with respect to the parameter range. This phenomenon attracts a logic that the model either captures the decay rate of autocovariance of fractional noise or some fractal property of sample paths. One might also fine-tune M_{LSTM} on ARFIMA inputs and see how this scatter plot changes. As it can be seen in Figure 6.4, even fine-tuning on 10^4 sequences drastically improves the accuracy of the model.

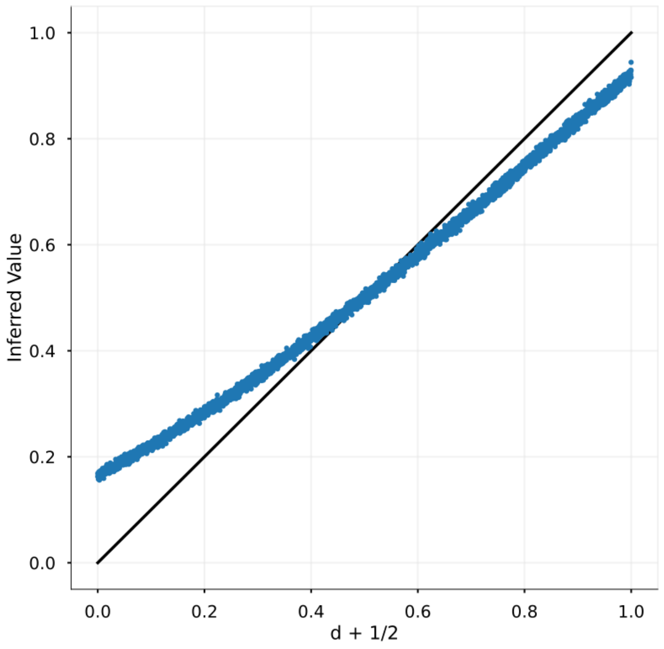

Figure 6.3: Scatterplot of M_{LSTM} model fine-tuned up to 12800 length fBm sequences, inferring on ARFIMA processes of $d \in (0, 1)$ and length 12800.

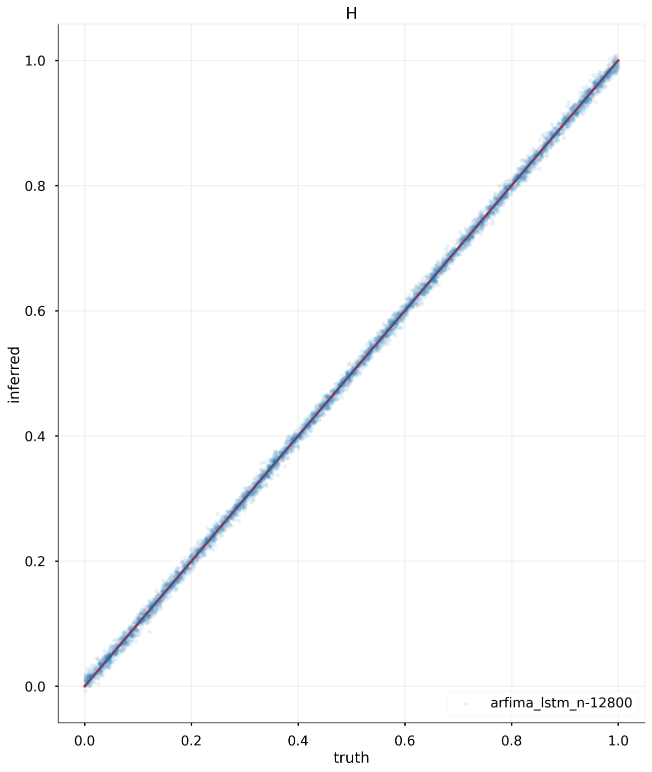

Figure 6.4: ARFIMA $H = d + 1/2$ scatterplot of M_{LSTM} fine-tuned up to 12800 length fBm sequences, then fine-tuned on 10^4 ARFIMA sequences.

6.4 Autoregressive processes

We tested M_{LSTM} on autoregressive dynamics of order 1, results are shown in Figure 6.5. There are two parameter values that can be explained with high confidence. On one hand, when the speed of mean reversion vanishes, inferred values do so too: this corresponds to the fact that, in some sense, as we approach zero with the Hurst parameter H, that is when $H \to 0$, increments of fractional Brownian motion display a behavior that is comparable to that of white-noise. On the other hand, when the speed of mean reversion is close to 1, then the autoregressive process coincides with a random walk driven by the underlying noise, and as such corresponds to standard Brownian motion, that is a fractional Brownian motion with Hurst parameter $H = 1/2$, which explains the inferred value. The regularity of the inference curve can probably be explained by the continuous nature of neural networks.

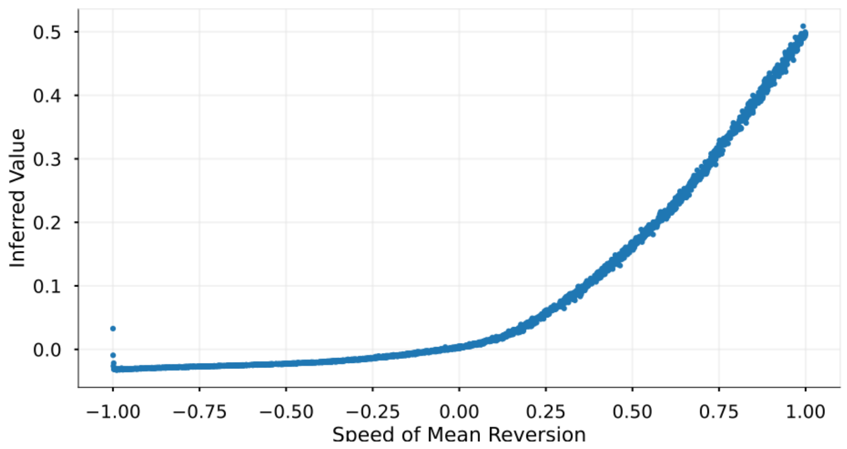

Figure 6.5: Scatterplot of M_{LSTM} model fine-tuned up to 12800 length fBm sequences, inferring on autoregressive processes of order 1 and length 12800.

6.4.1 Ornstein–Uhlenbeck processes

We tested M_{LSTM}, which was trained on fBm inputs, on standard Ornstein-Uhlenbeck processes. The resulting scatter plot can be seen in Figure 6.6. The inferred value at $\alpha = 0$ is 1/2 as expected — since the model receives an input that it already encountered in the learning phase. When the parameter of the input deviates from zero, we see a decreasing convex curve. A possible explanation is that when $\alpha \neq 0$, then the autocovariance shows exponential decay - contrary to the power decay associated with the data the model perceived when it was calibrated.

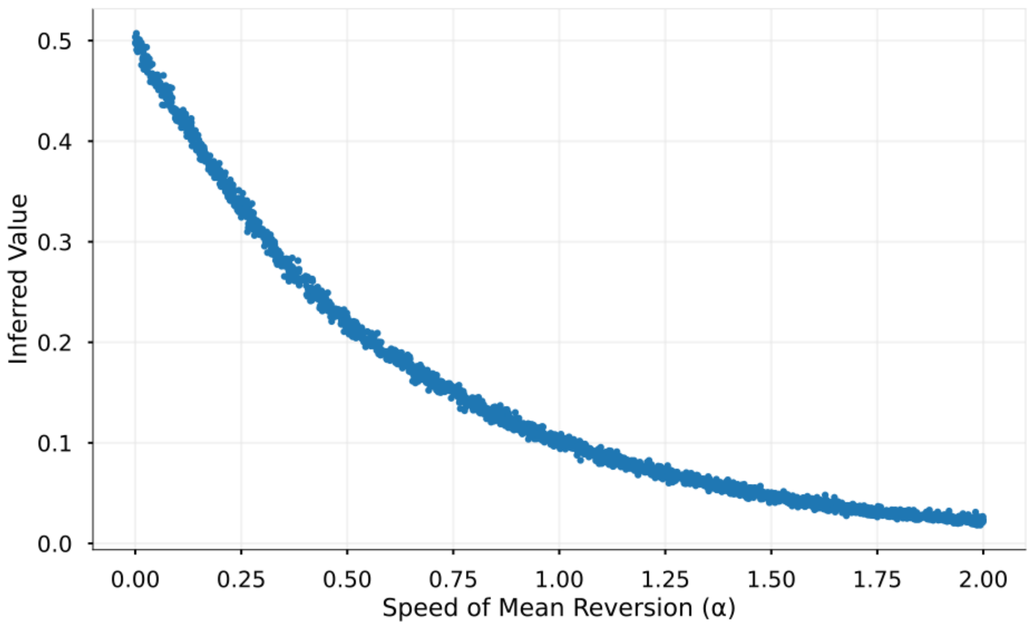

Figure 6.6: Scatterplot of M_{LSTM} model finetuned up to 12800 length fBm sequences, inferring on Ornstein–Uhlenbeck processes of length 12800.

6.5 Tests on sequences generated by the YUIMA R package

On a final note, we evaluated some of our models on sequences generated by the R package YUIMA [6]. This was also an indirect way of validating the process generators we used. In all of the cases, the evaluated models were fine-tuned for the longest sequence length in the respective tables, as such they do not yield the best possible metrics for other sequence lengths. In every case, 10^5 realizations were generated. In the case of the fOU and fBm, the prior distribution on the parameters was exactly as specified in Chapter 5. The experiments we conducted with these processes are also similar to those of Chapter 5. The results of these experiments can be seen in Tables 6.1 and 6.2.

Table 6.1: Performance metrics of fBm Hurst estimators M_{LSTM} and M_{conv} by sequence length on sequences generated by the YUIMA package.

	MSE loss ($\times 10^{-3}$)		$\hat{b}_{0.025}$ ($\times 10^{-3}$)		$\hat{s}_{0.025}$ ($\times 10^{-2}$)	
seq. len.	M_{LSTM}	M_{conv}	M_{LSTM}	M_{conv}	M_{LSTM}	M_{conv}
100	4.78	5.16	25.7	19.3	6.12	6.61
200	2.07	2.19	12.9	9.73	4.21	4.41
400	0.947	1.00	6.10	4.79	2.92	3.03
800	0.449	0.470	3.33	2.47	2.03	2.10
1600	0.222	0.236	2.04	1.84	1.43	1.49
3200	0.111	0.121	1.46	1.88	1.011	1.06
6400	0.0565	0.0668	1.35	2.68	0.715	0.748
12800	0.0300	0.0412	1.40	3.07	0.511	0.535

Table 6.2: Performance metrics of fOU Hurst estimator M_{LSTM} by sequence length on sequences generated by the YUIMA package.

seq. len.	MSE ($\times 10^{-3}$)	$\hat{b}_{0.025}$ ($\times 10^{-3}$)	$\hat{s}_{0.025}$ ($\times 10^{-2}$)
100	8.19	16.0	8.37
200	2.85	7.23	5.08
400	1.22	4.13	3.35
800	0.577	2.79	2.32
1600	0.293	2.19	1.65
3200	0.151	1.94	1.17
6400	0.0815	2.25	0.839

Chapter 7

Conclusion and future work

In this work, we have demonstrated the utility of sequence-processing neural networks as an effective estimation method for estimating parameters of several stochastic processes. In the context of the fractional Brownian motion, we constructed an estimator for the important Hurst exponent. Additionally, we employed special neural network architecture to estimate the shift parameter μ and scale parameter σ. In the case of the fractional Ornstein-Uhlenbeck process, in addition to parameters H, μ and σ, we also obtained an estimator for the drift parameter α.

In most cases, we compared the performance and consistency of these neural network-based methods with commonly used baseline methods. These comparisons overwhelmingly favored our neural network-based approach. We believe that the proposed parameter estimators of stochastic processes, based on neural networks, form a good basis for the estimation methods that can be used for more intricate processes.

Obtaining neural network-based estimators for other processes, such as the CIR process [9] might be the subject of further research. The parameter estimation of the CIR process is important when modeling stochastic volatility (see Heston model [18]). Therefore, we could apply the resulting parameter estimators in option pricing, or we could even try to train neural networks directly for option pricing, thus bypassing the parameter estimation.

Moreover, the ARFIMA(p, d, q) process could be studied, which imposes an order selection problem as well, something we did not have to consider when discussing the ARFIMA$(0, d, 0)$ case.

Bibliography

[1] U Rajendra Acharya et al. "Application of non-linear and wavelet based features for the automated identification of epileptic EEG signals". In: *International journal of neural systems* 22.02 (2012), p. 1250002.

[2] Aleksejus Kononovicius. *arfima: Python implementation of ARFIMA process with an aim to simulate series.* https://github.com/akononovicius/arfima. 2021.

[3] Richard T. Baillie. "Long memory processes and fractional integration in econometrics." In: *Journal of Econometrics* 73.1 (1994), pp. 5–59.

[4] Jan Beran. *Statistics for Long-Memory Processes.* Nov. 2017, pp. 1–315. ISBN: 9780203738481. DOI: `10.1201/9780203738481`.

[5] Alexandre Brouste and Stefano M. Iacus. *Parameter estimation for the discretely observed fractional Ornstein-Uhlenbeck process and the Yuima R package.* 2011. arXiv: `1112.3777 [stat.CO]`.

[6] Alexandre Brouste et al. "The yuima project: A computational framework for simulation and inference of stochastic differential equations". In: *Journal of Statistical Software* 57 (2014), pp. 1–51.

[7] Patrick Cheridito, Hideyuki Kawaguchi, and Makoto Maejima. "Fractional Ornstein-Uhlenbeck processes". In: *Electronic Journal of Probability* 8.none (2003), pp. 1–14. DOI: `10.1214/EJP.v8-125`. URL: `https://doi.org/10.1214/EJP.v8-125`.

[8] Christopher Flynn. *fbm: Exact methods for simulating fractional Brownian motion and fractional Gaussian noise in python.* https://github.com/crflynn/fbm. 2020.

[9] John C Cox, Jonathan E Ingersoll Jr, and Stephen A Ross. "A theory of the term structure of interest rates". In: *Theory of valuation.* World Scientific, 2005, pp. 129–164.

[10] Zoltan Eisler and Janos Kertesz. "Size matters: some stylized facts of the stock market revisited". In: *The European Physical Journal B-Condensed Matter and Complex Systems* 51 (2006), pp. 145–154.

[11] Christian LE Franzke et al. "A dynamical systems explanation of the Hurst effect and atmospheric low-frequency variability". In: *Scientific reports* 5.1 (2015), pp. 1–6.

[12] Liudas Giraitis, Hira Koul, and Donatas Surgailis. "Large Sample Inference For Long Memory Processes". In: (Apr. 2012). DOI: 10.1142/p591.

[13] Tilmann Gneiting and Martin Schlather. "Stochastic Models That Separate Fractal Dimension and the Hurst Effect". In: *SIAM Review* 46.2 (2004), pp. 269–282. DOI: 10.1137/S0036144501394387. eprint: https://doi.org/10.1137/S0036144501394387. URL: https://doi.org/10.1137/S0036144501394387.

[14] Tilmann Gneiting, Hana Ševčíková, and Donald B Percival. "Estimators of fractal dimension: Assessing the roughness of time series and spatial data". In: *Statistical Science* (2012), pp. 247–277.

[15] Timothy Graves et al. "A brief history of long memory: Hurst, Mandelbrot and the road to ARFIMA, 1951–1980". In: *Entropy* 19.9 (2017), p. 437.

[16] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[17] Simon Haykin. *Neural networks: a comprehensive foundation.* Prentice Hall PTR, 1994.

[18] Steven L Heston. "A closed-form solution for options with stochastic volatility with applications to bond and currency options". In: *The review of financial studies* 6.2 (1993), pp. 327–343.

[19] Tomoyuki Higuchi. "Approach to an irregular time series on the basis of the fractal theory". In: *Physica D: Nonlinear Phenomena* 31.2 (1988), pp. 277–283.

[20] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[21] Yaozhong Hu and David Nualart. "Parameter estimation for fractional Ornstein–Uhlenbeck processes". In: *Statistics & probability letters* 80.11-12 (2010), pp. 1030–1038.

[22] Harold Edwin Hurst. "Methods of using long-term storage in reservoirs." In: *Proceedings of the institution of civil engineers* 5.5 (1956), pp. 519–543.

[23] Serkan Kiranyaz et al. "1D convolutional neural networks and applications: A survey". In: *Mechanical systems and signal processing* 151 (2021), p. 107398.

[24] Dirk P Kroese and Zdravko I Botev. "Spatial process generation". In: *arXiv preprint arXiv:1308.0399* (2013).

[25] Ming Li. "Change trend of averaged Hurst parameter of traffic under DDOS flood attacks". In: *Computers & security* 25.3 (2006), pp. 213–220.

[26] Ilya Loshchilov and Frank Hutter. "Decoupled weight decay regularization". In: *arXiv preprint arXiv:1711.05101* (2017).

[27] Benoit B Mandelbrot and John W Van Ness. “Fractional Brownian motions, fractional noises and applications”. In: *SIAM review* 10.4 (1968), pp. 422–437.

[28] Percy Moran and Peter Whittle. *Hypothesis Testing in Time Series Analysis.* 1951.

[29] Dmitry Mottl. *hurst: Hurst exponent evaluation and R/S-analysis in Python.* `https://github.com/Mottl/hurst`. 2019.

[30] *neptune.ai: experiment tracking and model registry.* https://neptune.ai. 2022.

[31] D. Nualart and E Nualart. *Introduction to Malliavin Calculus.* 1th. Cambridge University Press, 2018. DOI: `doi:10.1017/9781139856485`.

[32] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems.* Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: `https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf`.

[33] Bo Qian and Khaled Rasheed. “Hurst exponent and financial market predictability”. In: *IASTED conference on Financial Engineering and Applications.* Proceedings of the IASTED International Conference Cambridge, MA. 2004, pp. 203–209.

[34] M.A. Nunes R.C. Lopes. “Long memory analysis in DNA sequences.” In: *Physica A* 361.2 (2006), pp. 569–588.

[35] V. Seshadri and Bruce J. West. “Fractal Dimensionality of Levy Processes.” In: *Proceedings of the National Academy of Sciences of the United States of America* 79.14 (1982), pp. 4501–5.

[36] Raphael Vallat. *AntroPy: entropy and complexity of (EEG) time-series in Python.* `https://github.com/raphaelvallat/antropy`. 2022.

[37] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[38] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: `10.1038/s41592-019-0686-2`.

[39] Walter Willinger et al. *Long-Range Dependence and Data Network Traffic.* 2001.