Simulation and Estimation of Operator scaling stable random Fields

DISSERTATION

zur Erlangung des Grades eines Doktors
der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von

Dipl.–Math. Tobias Kegel

Eingereicht bei der Naturwissenschaftlich-Technischen Fakultät
der Universität Siegen

Siegen 2011

GUTACHTER:

Prof. Dr. Hans-Peter Scheffler, Universität Siegen
Prof. Dr. Peter Kern, Heinrich-Heine-Universität Düsseldorf

TAG DER MÜNDLICHEN PRÜFUNG: 14. Oktober 2011

Kurzzusammenfassung

Operator-skalierende stabile Zufallsfelder (engl. *Operator-scaling stable random fields*, kurz: OSSRFs) sind stochastische Modelle, die räumliche Abhängigkeiten beschreiben können. Dabei sind Abhängigkeiten unterschiedlicher Stärke und in verschiedenen, nicht notwendigerweise zueinander senkrechten, Richtungen zugelassen. Die resultierenden anisotropen Felder werden z.B. zur Beschreibung poröser Medien in der Hydrologie oder fraktaler Oberflächen in der Physik verwendet. In [1] präsentierten H. Biermé, M. M. Meerschaert und H.-P. Scheffler Modelle für operator-skalierende stabile Zufallsfelder in *harmonizable* und in *moving-average* Darstellungen, und zeigten einige wichtige Eigenschaften dieser Felder.

Um diese OSSRFs in praktischen Anwendungen einsetzen zu können, werden Methoden für die numerische Simulation und für die statistische Analyse (z.B. Parameterschätzung) solcher Felder benötigt. In der vorliegenden Arbeit werden numerische Approximationen von OSSRFs präsentiert und ihre Abweichungen von den ursprünglichen Feldern untersucht. Algorithmen für die Berechnung dieser Approximationen wurden ebenfalls entwickelt und in dieser Arbeit vorgestellt. Für die in der Praxis relevanten Fälle von zwei- und drei-dimensionalen Feldern wurden diese Algorithmen in den Programmiersprachen Matlab und Java implementiert. Schliesslich stellen wir auch eine Methode für die Schätzung mehrerer Parameter eines *harmonizable* OSSRF sowie ihre Implementierung in Matlab vor.

Abstract

Operator-scaling stable random fields are stochastic models which can describe spacial dependencies. Thereby dependencies of different intensities and in different, not necessarily orthogonal, directions are allowed, resulting in anisotropic fields which are used, e.g. in hydrology to represent porous media, or to describe fractal surfaces in physics. In [1], Bierme, Meerschaert and Scheffler presented models for operator-scaling stable random fields in harmonizable and in moving average representation, and showed some important properties of these fields.

In order to use these fields for practical application, procedures for their numeric simulation are needed, and also methods for the statistical analysis (e.g. parameter estimation) of observed realizations of OSSRFs. The present thesis presents numeric approximations of OSSRFs and examines their deviation from the original OSSRFs. Algorithms for the calculation of these approximations have been also developed and are described in the thesis. For the cases of two- and three-dimensional fields, which are relevant for practical applications, these algorithms for the simulation of OSSRFs have been implemented in the programming languages Matlab and Java. Finally, we present also a method for the estimation of several parameters of a two-dimensional harmonizable OSSRF, and its implementation in Matlab.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Operator-scaling stable random fields are stochastic models which describe spacial dependencies. Thereby dependecies of different intensities and in different, not necessarily orthogonal, directions are allowed, resulting in anisotropic fields which are used, e.g. in hydrology to represent porous media, or to describe fractal surfaces in physics. In [1], Bierme, Meerschaert and Scheffler presented models for operator-scaling stable random fields in harmonizable and in moving average representation, which can be anisotropic, and showed some important properties of these fields.

In order to use these fields for practical application, procedures for their numeric simulation are needed, and also methods for the statistical analysis (e.g. parameter estimation) of observed realizations of OSSRFs. The present thesis presents numeric approximations of OSSRFs and examines their deviation from the original OSSRFs. Algorithms for the calculation of these approximations have been also developed and are described in the thesis. For the cases of two- and three-dimensional fields, which are relevant for practical applications, these algorithms for the simulation of OSSRFs have been implemented in the programming languages Matlab and Java. Finally, we present also a method for the estimation of several parameters of a two-dimensional harmonizable OSSRF, and its implementation in Matlab.

This thesis is organized as follows: In Chapter 2, we quote the basic definitions and results from [1], which are used in this thesis, e.g. the definitions of E-homogeneous functions and of operator-scaling stable random fields.

In Chapter 3, we present an approximation of the harmonizable integral representation of OSSRFs by a discrete model, and calculate error bounds for this approximation. In Section 3.1, the approximation is described. It is obtained in two steps: In the first step the domain of integration is truncated from $\mathbb{R}^d$ to a finite subset, and in the second step this finite integral is approximated by a sum with a finite number of summands. After the presentation of this approximation, we examine the resulting error between the approximation and the original model, and divide this task again in two parts: First, in Section

3.2, we estimate the approximation error resulting from the truncation. We give exact error bounds for isotropic OSSRFs, which is defined with a norm as the E-homogeneous function, in Theorem 3.5, and for harmonizable OSSRFs in general in Theorem 3.10. Then we consider the approximation error due to the discretisation in Section 3.3. We use two similar variants of discretisation, the first one being a special case of the second. For both cases, the approximation error is estimated, like the approximation error for the truncation, first for the special case of OSSRFs with norms as E-homogeneous functions, and then for the general class of OSSRFs in harmonizable representation. Thus, we show that using the second variant of discretisation, the approximation error can be reduced below any given threshold $\varepsilon > 0$ by choosing suitable parameters for the approximation (Theorem 3.34, Corollary 3.35 and Remark 3.36), and give exact error bounds in the case of the E-homogeneous function being a norm (Theorem 3.20 and Corollary 3.21).

Chapter 4 contains similar considerations for OSSRFs in moving-average representation. Like the previous chapter, it consists of three sections: Section 4.1 contains, analogous to Section 3.1, a description of the approximation of these OSSRFs. Again, the approximation is obtained by truncating the domain of integration in the integral occuring in the definition of the OSSRF, and subsequently approximating the truncated integral by a finite sum. For OSSRFs for which the E-homogeneous function is a norm, the approximation error of the truncation is estimated in Section 4.2, and the error due to discretisation in Section 4.3. For both error estimates, exact error bounds are obtained.

In Chapter 5 we develop algorithms for the numeric calculation of the approximations which were presented in the Chapters 3 and 4. First, we describe in Section 5.1, how the α-stable random variables, which are needed for the approximation of the OSSRFs, can be simulated using the algorithm published by Chambers, Mallows and Stuck in [4]. Then we develop an algorithm for the simulation of two-dimensional harmonizable OSSRFs in Section 5.2, which is generalized for the simulation of d-dimensional OSSRFs with $d \geq 2$ (e.g. three-dimensional fields) in Section 5.3. In the Sections 5.4 and 5.5, an algorithm for the simulation of OSSRFs in moving-average representation is presented. Because the algorithm for the simulation of harmonizable OSSRFs includes the calculation of a discrete Fourier transform, the basic principles of an efficient, fast Fourier transform are explained in Section 5.6. Finally, we show in Section 5.7, how the convolution of two arrays, which is a key element of the algorithm for the simulation of moving-average OSSRFs, can be calculated efficiently with the help of the fast Fourier transform.

These algorithms for the simulation of OSSRFs were implemented in the programming languages Matlab and Java for the important cases of two- and three-dimensional random fields. The complete source codes of these implementations can be found on the accompanying CD. In Chapter 6, we analyze and compare the required system resources (memory space and computation time) which are needed by the different implementation variants of the simulation algorithms, and show some images of examples of OSSRFs that have been generated by these programs. We also present the relevant source codes of the Matlab implementations in Section 6.1, while the Java sources are not quoted in the thesis because of their size. However, an user manual for the Java program is included in Appendix A.

Finally, we turn from the topic of the approximation and simulation of OSSRFs to the estimation of their parameters. In Chapter 7, we discuss a method for the estimation of several parameters of an OSSRF, presenting an algorithm for this task in Section 7.1 and implementing it in Matlab for the case of five estimated parameters of a two-dimensional harmonizable OSSRF (see Section 7.2). In Section 7.3, the results of a numerical case study are presented which indicate that the described algorithm is indeed an useful and effective means for the estimation of parameters. In particular, it gives empirical evidence to the assumption that the estimator is consistent, i.e. that the estimated parameter values converge to the underlying, original values if the sizes of the simulated fields (and thus the sample size) are increased. The results for the considered sample indicate that the mean squared error is proportional to the inverse of the number of elements in the simulated field, and some other statistics, like the median of absolute deviations, are proportional to its square root, i.e. to the inverse of the width and height of the field. In this context, further research is needed in order to prove the consistency of this estimator mathematically.

Chapter 2

Definition of OSSRFs

The *Operator Scaling Stable Random Fields* (OSSRFs) in *harmonizable* or *moving average* representation, which will be considered in this thesis, have been introduced in the paper [1]. In this chapter we quote some definitions and results from that paper which will be used in the thesis. The proofs of these results are not given here, but can be found in the paper.

In section 2 of [1], a polar representation of vectors $x \in \mathbb{R}^d \backslash \{0\}$, depending on a matrix E, is defined:

Let E be a real $d \times d$ matrix with positive real parts of the eigenvalues $0 < a_1 < \ldots < a_p$ for $p \leq d$. Let us define $\Gamma = \mathbb{R}^d \backslash \{0\}$. It follows from Lemma 6.1.5 of [8] that there exists a norm $||\cdot||_0$ on $\mathbb{R}^d$ such that for the unit sphere $S_0 = \{x \in \mathbb{R}^d : ||x||_0 = 1\}$ the mapping $\Psi : (0, \infty) \times S_0 \to \Gamma, \Psi(r, \theta) = r^E \theta$ is a homomorphism. Moreover for any $x \in \Gamma$ the function $t \mapsto ||t^E x||_0$ is strictly increasing. Hence we can write any $x \in \Gamma$ uniquely as $x = \tau(x)^E l(x)$ for some *radial part* $\tau(x) > 0$ and some direction $l(x) \in S_0$ such that $x \mapsto \tau(x)$ and $x \mapsto l(x)$ are continuous. Observe that $S_0 = \{x \in \mathbb{R}^d : \tau(x) = 1\}$ is compact. Moreover we know that $\tau(x) \to \infty$ as $x \to \infty$ and $\tau(x) \to 0$ as $x \to 0$. Hence we can extend $\tau(\cdot)$ continuously by setting $\tau(0) = 0$. Note that further $\tau(-x) = \tau(x)$ and $l(-x) = -l(x)$. The following result gives bounds on the growth rate of $\tau(x)$ in terms of the real parts of the eigenvalues of E.

Lemma 2.1. *(Lemma 2.1 in [1])*
For any (small) $\delta > 0$ there exist constants $C_1, \ldots, C_4 > 0$ such that for all $||x||_0 \leq 1$ or all $\tau(x) \leq 1$,

$$C_1 ||x||_0^{1/a_1 + \delta} \leq \tau(x) \leq C_2 ||x||_0^{1/a_p - \delta},$$

and, for all $||x||_0 \geq 1$ or all $\tau(x) \geq 1$,

$$C_3 ||x||_0^{1/a_p - \delta} \leq \tau(x) \leq C_4 ||x||_0^{1/a_1 + \delta}.$$

The following proposition provides an integration in *polar coordinates* formula (with $q = \text{trace}(E)$).

Proposition 2.2. *(Proposition 2.3 in [1])*
There exists a unique finite Radon measure σ on S_0 such that for all $f \in L^1(\mathbb{R}^d, dx)$ we have

$$\int_{\mathbb{R}^d} f(x)dx = \int_0^\infty \int_{S_0} f(r^E\theta)\sigma(d\theta)r^{q-1}dr.$$

Definition 2.3. (Definition 2.6 in [1])
Let $\varphi : \mathbb{R}^d \to \mathbb{C}$ be any function. We say that φ is *E-homogeneous* if $\varphi(c^E x) = c\varphi(x)$ for all $c > 0$ and $x \in \Gamma$.

Remark 2.4. (from section 2 in [1], following after Definition 2.6)
An E-homogeneous function φ is completely determined by its values on S_0, since $\varphi(x) = \varphi(\tau(x)^E l(x)) = \tau(x)\varphi(l(x))$. Observe that if φ is E-homogeneous and continuous with positive values on Γ, then

$$M_\varphi = \max_{\theta \in S_0} \varphi(\theta) > 0 \quad \text{and} \quad m_\varphi = \min_{\theta \in S_0} \varphi(\theta) > 0. \tag{2.1}$$

Moreover by continuity we necessarily have $\varphi(0) = 0$.

Definition 2.5. (Definition 2.7 in [1])
Let $\beta > 0$. A continuous function $\varphi : \mathbb{R}^d \to [0, \infty)$ is called (β, E)-*admissible*, if $\varphi(x) > 0$ for all $x \neq 0$ and for any $0 < A < B$ there exists a positive constant $C > 0$ such that, for $A \leq ||y|| \leq B$,

$$\tau(x) \leq 1 \Rightarrow |\varphi(x + y) - \varphi(y)| \leq C\tau(x)^\beta.$$

In the following corollary, a family of E-homogenous, (β, E)-admissible functions in a certain parametric representation is defined. In this thesis, we usually use (if not stated otherwise) E-homogeneous and (β, E)-admissible functions which can be written in this representation.

Corollary 2.6. *(Corollary 2.12 in [1])*
Let $\theta_1, \ldots, \theta_d$ be any basis of $\mathbb{R}^d$, let $0 < \lambda_1 \leq \ldots \leq \lambda_d$ and $C_1, \ldots, C_d > 0$. Choose a $d \times d$ matrix E such that $E^T\theta_j = \lambda_j\theta_j$ for $j = 1, \ldots, d$. Then for any $\rho > 0$, if $\rho < 2\lambda_1$ the function

$$\varphi(x) = \left(\sum_{j=1}^d C_j | < x, \theta_j > |^{\rho/\lambda_j} \right)^{1/\rho} \tag{2.2}$$

is a continuous E-homogeneous and (β, E)-admissible function for $\beta < \min\left(\lambda_1, \rho\frac{\lambda_1}{\lambda_d}\right)$ if $\lambda_1 \leq \rho$ and $\beta = \rho$ if $\lambda_1 > \rho$.

Definition 2.7. (see section 1 in [1])
A scalar valued random field $\{X(x)\}_{x \in \mathbb{R}^d}$ is called *operator-scaling* if for some $d \times d$ matrix E with positive real parts of the eigenvalues and some $H > 0$ we have

$$\{X(x^E)\}_{x \in \mathbb{R}^d} \overset{f.d.}{=} \{c^H X(x)\}_{x \in \mathbb{R}^d} \text{ for all } c > 0, \tag{2.3}$$

where $\overset{f.d.}{=}$ denotes equality of all finite-dimensional marginal distributions, and $c^E = \exp(E\log(c))$ where $\exp(A) = \sum_{k=0}^{\infty} \frac{A^k}{k!}$ is the matrix exponential.

Furthermore, a scalar valued random field is called *stable*, if all its finite dimensional marginal distributions are stable (see [10], Def. 3.1.1), i.e. α-stable for an $\alpha \in (0, 2]$. A random field, which is both operator scaling and stable, is called *operator scaling stable random field*, or abbreviated: *OSSRF*. In this thesis, two classes of OSSRF will be considered, which are the OSSRFs in *harmonizable representation* and the OSSRFs in *moving average representation.*

Theorem 2.8. *(Theorem 4.1 and Corollary 4.2 in [1])*
For $0 < \alpha \leq 2$, be $W_\alpha(d\xi)$ a complex isotropic α-stable random measure with Lebesgue control measure (see [10], p. 281). Be E a real $d \times d$ matrix with $0 < a_1 < \ldots < a_p$ denoting the real parts of the eigenvalues of E, and let $q = trace(E)$. Let $\psi : \mathbb{R}^d \to [0, \infty)$ be a continuous, E^T-homogeneous function such that $\psi(x) \neq 0$ for $x \neq 0$. Then for any $0 < \alpha \leq 2$ the random field

$$X_\psi(x) = Re \int_{\mathbb{R}^d} \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-q/\alpha}\, W_\alpha(d\xi), \quad x \in \mathbb{R}^d \tag{2.4}$$

exists and is stochastically continuous if and only if $H \in (0, a_1)$. It is operator-scaling and has stationary increments.

Definition 2.9. The representation (2.4) of the OSSRF $\{X_\psi(x)\}_{x\in\mathbb{R}^d}$ in Theorem 2.8 is called *harmonizable representation.*

Theorem 2.10. *(Theorem 3.1 and Corollary 3.2 in [1])*
For $0 < \alpha \leq 2$ be $Z_\alpha(dy)$ an independently scattered, symmetric α-stable random measure on $\mathbb{R}^d$ with Lebesgue control measure λ^d. Be E a real $d \times d$ matrix with $0 < a_1 < \ldots < a_p$ denoting the real parts of the eigenvalues of E, and let $q = trace(E)$. Be $\beta > 0$. Let $\varphi : \mathbb{R}^d \to [0, \infty)$ be an E-homogeneous, (β, E)-admissible function. Then for any $0 < \alpha \leq 2$ and any $0 < H < \beta$ the random field

$$X_\varphi(x) = \int_{\mathbb{R}^d} \left(\varphi(x - y)^{H-q/\alpha} - \varphi(-y)^{H-q/\alpha}\right)\, Z_\alpha(dy), \quad x \in \mathbb{R}^d \tag{2.5}$$

exists and is stochastically continuous. It is operator-scaling and has stationary increments.

Definition 2.11. The representation (2.5) of the OSSRF $\{X_\varphi(x)\}_{x\in\mathbb{R}^d}$ in Theorem 2.10 is called *moving average representation.*

Chapter 3

Approximation of OSSRFs in harmonizable representation

3.1 Approximation

Be X_ψ an OSSRF in harmonizable representation on the $\mathbb{R}^d$. In order to simulate this random field numerically, the integral which occurs in its definition (see (2.4)) has to be approximated by a finite sum. This approximation is performed in two steps:

(a) Truncation of the domain of integration from $\mathbb{R}^d$ to $[-A, A]^d$: The scope of integration is reduced from the complete space $\mathbb{R}^d$ to the finite area $[-A, A]^d$ (for a "large" positive real number A; in this thesis it is assumed that $A > 1$). Thus, $X_\psi(x)$ is approximated by

$$X_\psi^A(x) = \operatorname{Re} \int_{[-A,A]^d} \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}}\, W_\alpha(d\xi). \tag{3.1}$$

(b) Discretisation: The integral over $[-A, A]^d$ is approximated by a finite sum, by dividing $[-A, A]^d$ into $(2M)^d$ small hypercubes $\Delta_{k_1,\ldots,k_d}$ of size (length of side) $D := \frac{A}{M}$ (note: in this thesis, the d-dimensional vector of indices $(k_1, \ldots, k_d)^T$ will be written as $\vec{\mathbf{k}}$), and approximating the function $\left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}}$ on each of these small hypercubes by a constant value $g_{\vec{\mathbf{k}}} \in [0, \infty)$. Thus, $X_\psi^A(x)$ is approximated by the sum

$$X_\psi^{A,M}(x) = \operatorname{Re} \sum_{\vec{\mathbf{k}} \in \{-M,\ldots,M-1\}^d} g_{\vec{\mathbf{k}}} \cdot W_\alpha(\Delta_{\vec{\mathbf{k}}}) \tag{3.2}$$

with $\Delta_{\vec{\mathbf{k}}} := [k_1 \cdot D, (k_1 + 1) \cdot D) \times \ldots \times [k_d \cdot D, (k_d + 1) \cdot D)$ for each $\vec{\mathbf{k}} = (k_1, \ldots, k_d) \in \{-M, \ldots, M - 1\}^d$

In this chapter, two slightly different versions of the discretisation are considered: First, $(g_{\vec{\mathbf{k}}})_{\vec{\mathbf{k}}\in J}$ is set to

$$g_{\vec{\mathbf{k}}} = \begin{cases} 0, & \vec{\mathbf{k}} \in \{-1,0\}^d \\ \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>} - 1\right) \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}, & \text{else} \end{cases} \tag{3.3}$$

with $\xi_{\vec{\mathbf{k}}} := (k_1 \cdot D, \dots, k_d \cdot D)^T$. Then it is compared with a discretisation in which the approximation $g_{\vec{\mathbf{k}}}$ is set to zero not only for the indices $\vec{\mathbf{k}} \in \{-1,0\}^d$, but for the indices $\vec{\mathbf{k}} \in \{-N,\dots,N-1\}^d$, for an additional integer parameter $N > 0$, i.e. $g_{\vec{\mathbf{k}}}$ is defined as

$$g_{\vec{\mathbf{k}}} = \begin{cases} 0, & \vec{\mathbf{k}} \in \{-N,\dots,N-1\}^d \\ \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>} - 1\right) \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}, & \text{else} \end{cases} \tag{3.4}$$

Thereby, the "hole" near the origin, for which the $g_{\vec{\mathbf{k}}}$ are set to zero, doesn't have to shrink proportional to $D = \frac{A}{M}$ for growing M, but can be kept at about the same size $2B = 2N \cdot D = 2A \cdot \frac{N}{M}$ by choosing N approximately proportional to M (Obviously B doesn't change if A and the quotient $\frac{N}{M}$ are kept constant). The second version of the discretisation is a generalization of the first one: The first version is equivalent to the second with parameter $N = 1$.

3.2 Approximation error due to the truncation

It is desired to estimate the errors of the approximations ($X_\psi^A(x) - X_\psi(x)$ and $X_\psi^{A,M}(x) - X_\psi^A(x)$) for a given $x \in \mathbb{R}^d$, i.e. to calculate an upper bound for the absolute values of these differences. Because $X_\psi(x)$, $X_\psi^A(x)$ and $X_\psi^{A,M}(x)$ are symmetric α-stable ($S\alpha S$) random variables, their differences are $S\alpha S$ random variables, too. Therefore, as a measure of the approximation errors, the scale parameters of their distributions ($||X_\psi^A(x) - X_\psi(x)||_\alpha$ and $||X_\psi^{A,M}(x) - X_\psi^A(x)||_\alpha$) have to be estimated. According to equation (3.4.4) in [10] (p. 122), such a norm for an α-stable integral is defined by $|| \int_{\mathbb{R}^d} f(\xi) W_\alpha(d\xi)||_\alpha = \left(\int_{\mathbb{R}^d} |f(\xi)|^\alpha d\xi\right)^{1/\alpha}$ (this means that $|| \int_{\mathbb{R}^d} f(\xi) W_\alpha(d\xi)||_\alpha^\alpha = \int_{\mathbb{R}^d} |f(\xi)|^\alpha d\xi$). Therefore, the scale parameter can be estimated by an estimation of this non-random integral.

Lemma 3.1. *The scale parameter of the approximation error due to the truncation can be bounded by the following integral:*

$$\left|\left|X_\psi(x) - X_\psi^A(x)\right|\right|_\alpha^\alpha \le \int_{\mathbb{R}^d \setminus [-A,A]^d} \left|e^{i<x,\xi>} - 1\right|^\alpha \psi(\xi)^{-\alpha H - q} d\xi$$

Proof.

$$\begin{aligned}\left|\left|X_\psi(x) - X_\psi^A(x)\right|\right|_\alpha^\alpha &= \Bigg|\Bigg|\mathrm{Re}\int_{\mathbb{R}^d} \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}}\, W_\alpha(d\xi) \\ &\quad - \mathrm{Re}\int_{[-A,A]^d} \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}}\, W_\alpha(d\xi)\Bigg|\Bigg|_\alpha^\alpha \\ &\underset{\mathrm{Re}}{\leq} \left|\left|\int_{\mathbb{R}^d\setminus[-A,A]^d} \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}}\, W_\alpha(d\xi)\right|\right|_\alpha^\alpha \\ &= \int_{\mathbb{R}^d\setminus[-A,A]^d} \left|\left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}}\right|^\alpha d\xi \\ &= \int_{\mathbb{R}^d\setminus[-A,A]^d} \left|\left(e^{i<x,\xi>} - 1\right)\right|^\alpha \psi(\xi)^{-\alpha H-q}\, d\xi\end{aligned}$$

□

The error of approximation by truncation of the domain of integration can be estimated with the help of Lemma 2.1, Proposition 2.2 and Definition 2.3 as follows:

Theorem 3.2. *There are positive real numbers C and δ', which depend on the approximated random field X_ψ, (i.e. on the function ψ and on the parameters α and H), but not on A, so that*

$$\left|\left|X_\psi(x) - X_\psi^A(x)\right|\right|_\alpha^\alpha \leq C \cdot A^{-\frac{\alpha H}{\lambda_d}+\delta'}$$

and therefore (with $\widetilde{C} := C^{1/\alpha}$ and $\delta'' = \frac{\delta'}{\alpha}$)

$$\left|\left|X_\psi(x) - X_\psi^A(x)\right|\right|_\alpha \leq \widetilde{C} \cdot A^{-\frac{H}{\lambda_d}+\delta''}.$$

This estimation is valid for all dimensions $d \geq 2$ and for all E^T-homogeneous functions ψ, even if ψ can't be represented in the special form of (2.2) *(in the more general case - if ψ is not in this special form - the value of λ_d in this estimation is the largest real part of eigenvalues of E).*

Proof. According to Lemma 3.1,

$$\left|\left|X_\psi(x) - X_\psi^A(x)\right|\right|_\alpha^\alpha \leq \int_{\mathbb{R}^d\setminus[-A,A]^d} \left|e^{i<x,\xi>} - 1\right|^\alpha \psi(\xi)^{-\alpha H-q}\, d\xi.$$

Using the fact that $\left|e^{i<x,\xi>} - 1\right| \leq 2$, it follows that

$$\left|\left|X_\psi(x) - X_\psi^A(x)\right|\right|_\alpha^\alpha \leq 2^\alpha \int_{\mathbb{R}^d\setminus[-A,A]^d} \psi(\xi)^{-\alpha H-q}\, d\xi.$$

In the polar representation depending on E^T (see chapter 2), ξ can be represented uniquely in the form $\xi = \tau(\xi)^{E^T} l(\xi)$. According to Lemma 2.1, there exists a $\widetilde{C}_3 > 0$ so that $\tau(\xi) \geq \widetilde{C}_3 ||\xi||_0^{1/\lambda_d - \delta}$ if $||\xi||_0 \geq 1$. Because all norms on $\mathbb{R}^d$ are equivalent, there is a $\widetilde{C}_2 > 0$ with $||\xi||_0 \geq \widetilde{C}_2 \cdot ||\xi||_\infty$ for all $\xi \in \mathbb{R}^d$, so that

$$\tau(\xi) \geq \widetilde{C}_3 ||\xi||_0^{1/\lambda_d - \delta} \geq \widetilde{C}_3 \left(\widetilde{C}_2 \cdot ||\xi||_\infty\right)^{1/\lambda_d - \delta} \geq \widetilde{C}_3 \left(\widetilde{C}_2 \cdot A\right)^{1/\lambda_d - \delta} = \widetilde{C}_4 \cdot A^{1/\lambda_d - \delta}$$

if $||\xi||_\infty \geq A$ and $||\xi||_0 \geq 1$. Therefore, if A is sufficiently large, $||\xi||_\infty \geq A \Rightarrow \tau(\xi) \geq g(A)$ with $g(A) := \widetilde{C}_4 \cdot A^{1/\lambda_d - \delta}$. According to Proposition 2.2, there exists a unique finite Radon measure σ on S_0 such that

$$\int_{\mathbb{R}^d} 1_{\mathbb{R}^d \setminus [-A,A]^d}(\xi) \psi(\xi)^{-\alpha H - q} d\xi = \int_0^\infty \int_{S_0} 1_{\mathbb{R}^d \setminus [-A,A]^d}(r^{E^T}\theta) \psi(r^{E^T}\theta)^{-\alpha H - q} \, \sigma(d\theta) r^{q-1} \, dr.$$

The inequality for $\tau(\xi)$ implies that $||r^{E^T}\theta||_\infty \geq A \Rightarrow \tau(r^{E^T}\theta) \geq g(A)$ and therefore $1_{\mathbb{R}^d \setminus [-A,A]^d}(r^{E^T}\theta) \leq 1_{(g(A),\infty)}\left(\tau(r^{E^T}\theta)\right) = 1_{(g(A),\infty)}(r)$. Thus

$$\begin{aligned}
\left|\left|X_\psi(x) - X_\psi^A(x)\right|\right|_\alpha^\alpha &\leq 2^\alpha \int_{\mathbb{R}^d} 1_{\mathbb{R}^d \setminus [-A,A]^d}(\xi) \psi(\xi)^{-\alpha H - q} \, d\xi \\
&= 2^\alpha \int_0^\infty \int_{S_0} 1_{\mathbb{R}^d \setminus [-A,A]^d}(r^{E^T}\theta) \psi(r^{E^T}\theta)^{-\alpha H - q} \, \sigma(d\theta) r^{q-1} \, dr \\
&\leq 2^\alpha \int_0^\infty \int_{S_0} 1_{(g(A),\infty)}(r) \psi(r^{E^T}\theta)^{-\alpha H - q} \, \sigma(d\theta) r^{q-1} \, dr \\
&= 2^\alpha \int_{g(A)}^\infty \int_{S_0} (r \cdot \psi(\theta))^{-\alpha H - q} \, \sigma(d\theta) r^{q-1} \, dr \\
&= 2^\alpha \int_{g(A)}^\infty r^{-\alpha H - 1} \cdot \int_{S_0} \psi(\theta)^{-\alpha H - q} \, \sigma(d\theta) \, dr \\
&= 2^\alpha \cdot \left[\frac{1}{-\alpha H} \cdot r^{-\alpha H}\right]_{g(A)}^\infty \cdot \underbrace{\int_{S_0} \psi(\theta)^{-\alpha H - q} \, \sigma(d\theta)}_{=:\widetilde{C}_1} \\
&= 2^\alpha \cdot \widetilde{C}_1 \cdot \frac{1}{\alpha H} \cdot g(A)^{-\alpha H} \\
&= 2^\alpha \cdot \widetilde{C}_1 \cdot \frac{1}{\alpha H} \cdot \widetilde{C}_4^{-\alpha H} \left(A^{1/\lambda_d - \delta}\right)^{-\alpha H} \\
&= C \cdot A^{\frac{-\alpha H}{\lambda_d} + \delta'}
\end{aligned}$$

with $\delta' := \delta \alpha H$ and $C = 2^\alpha \cdot \widetilde{C}_1 \cdot \frac{1}{\alpha H} \cdot \widetilde{C}_4^{-\alpha H}$. □

This estimation has the drawback that it contains unknown constants (C and δ'), so that it can't be used to calculate a concrete value for the error. However, it shows that the error of estimation decreases with growing values of the parameter A, and converges to zero if $A \to \infty$, and it gives an approximate estimate for the rate of convergence.

In a special case, if ψ is a ρ-norm, then an estimation can be found, which only contains calculable numbers instead of the unknown constants of the previous estimation, and thereby allows to easily calculate an upper bound for $||X_\psi(x) - X_\psi^A(x)||_\alpha^\alpha$. This calculation uses the ($d-1$-dimensional) Lebesgue measure of the $||\cdot||_\infty$-unit-sphere, i.e. of $\{x \in \mathbb{R}^d : ||x||_\infty = 1\}$. Therefore this integral is first considered in a lemma:

Lemma 3.3. *The $d-1$-dimensional Lebesgue measure of $S_\infty := \{x \in \mathbb{R}^d : ||x||_\infty = 1\}$ is*

$$\int_{S_\infty} 1 \, dx = d \cdot 2^d. \tag{3.5}$$

Proof. The set $\{x \in \mathbb{R}^d : ||x||_\infty \leq 1\} = [-1,1]^d$ is a d-dimensional hypercube whose edges have the length 2. Its surface $\{x \in \mathbb{R}^d : ||x||_\infty = 1\}$ consists of $2d$ $d-1$-dimensional hypercubes whose edges have length 2, too. Each of this $d-1$-dimensional hypercubes has a Lebesgue-measure of 2^{d-1}. Therefore the whole surface $\{x \in \mathbb{R}^d : ||x||_\infty = 1\}$ (1-sphere according to the $||\cdot||_\infty$ - norm) has the measure $2d \cdot 2^{d-1} = d \cdot 2^d$. □

Remark 3.4. In the special cases $d = 2$ and $d = 3$, which are particularly relevant for real data and for simulation, the measure of S_∞ is $2 \cdot 2^2 = 8$ if $d = 2$, and $3 \cdot 2^3 = 24$ if $d = 3$. This can also be confirmed by a visualisation of this set in these cases: In the two-dimensional case, S_∞ is the perimeter of the square $[-1,1]^2$ and therefore consists of four edges of length 2 each, their sum being 8, and in the three-dimensional case, it is the surface of the cube $[-1,1]^3$, consisting of six squares with side length 2. Each of this squares has an area of 4, thus their total sum is 24.

This values of the Lebesgue measure of S_∞ are used in the following estimation of the approximation error in the case of ψ being a norm:

Theorem 3.5. *If $\psi = ||\cdot||_\rho$ with $1 \leq \rho$ (i.e. if in the representation of equation* (2.2) *the parameters are choosen as $\lambda_1 = \ldots = \lambda_d = C_1 = \ldots = C_d = 1$, and the vectors $\theta_1, \ldots, \theta_d$ are the standard unit vectors of the $\mathbb{R}^d$), then*

$$\left|\left|X_\psi(x) - X_\psi^A(x)\right|\right|_\alpha^\alpha \leq \frac{2^{\alpha+d} \cdot d}{\alpha H} \cdot A^{-\alpha H}$$

which implies

$$\left|\left|X_\psi(x) - X_\psi^A(x)\right|\right|_\alpha \leq 2 \cdot \left(\frac{2^d \cdot d}{\alpha H}\right)^{\frac{1}{\alpha}} \cdot A^{-H}.$$

Proof. According to Lemma 3.1,

$$\left|\left|X_\psi(x) - X_\psi^A(x)\right|\right|_\alpha^\alpha \leq \int_{\mathbb{R}^d\setminus[-A,A]^d} \left|e^{i<x,\xi>} - 1\right|^\alpha \psi(\xi)^{-\alpha H - q} d\xi$$

which implies for $\psi = ||\cdot||_\rho$ (and $q = \lambda_1 + \ldots + \lambda_d = 1 + \ldots + 1 = d$):

$$\left|\left|X_\psi(x) - X_\psi^A(x)\right|\right|_\alpha^\alpha \leq \int_{\mathbb{R}^d\setminus[-A,A]^d} \left|e^{i<x,\xi>} - 1\right|^\alpha ||\xi||_\rho^{-\alpha H - d} d\xi$$

This integral is estimated as follows:

$$\begin{aligned}
&\int_{\mathbb{R}^d\setminus[-A,A]^d} \underbrace{\left|e^{i<x,\xi>} - 1\right|^\alpha}_{\leq 2^\alpha} ||\xi||_\rho^{-\alpha H - d} d\xi \\
&\leq 2^\alpha \int_{\mathbb{R}^d\setminus[-A,A]^d} ||\xi||_\rho^{-\alpha H - d} d\xi \\
&\leq 2^\alpha \int_{\mathbb{R}^d\setminus[-A,A]^d} ||\xi||_\infty^{-\alpha H - d} d\xi \\
&= 2^\alpha \int_A^\infty \int_{S_\infty} ||r\cdot\zeta||_\infty^{-\alpha H - d} d\zeta \cdot r^{d-1}\, dr \\
&= 2^\alpha \int_A^\infty \int_{S_\infty} 1\, d\zeta \cdot r^{-\alpha H - d} \cdot r^{d-1}\, dr \\
&= 2^\alpha \cdot \int_{S_\infty} 1\, d\zeta \cdot \int_A^\infty r^{-\alpha H - 1}\, dr \\
&\underset{L.3.3}{\leq} 2^\alpha \cdot d \cdot 2^d \cdot \left[\frac{1}{-\alpha H} \cdot r^{-\alpha H}\right]_A^\infty \\
&= \frac{2^{\alpha+d}\cdot d}{\alpha H} \cdot A^{-\alpha H}.
\end{aligned}$$

□

Corollary 3.6. *Particularly, if $d = 2$ then $||X_\psi(x) - X_\psi^A(x)||_\alpha^\alpha \leq \frac{8\cdot 2^\alpha}{\alpha H} \cdot A^{-\alpha H}$ and if $d = 3$ then $||X_\psi(x) - X_\psi^A(x)||_\alpha^\alpha \leq \frac{24\cdot 2^\alpha}{\alpha H} \cdot A^{-\alpha H}$.*

Remark 3.7. As the exponent of A is negative (in Theorem 3.5), the estimated error of truncation converges to zero for $A \to \infty$. For any $\varepsilon > 0$, a value for A can easily be found so that $||X_\psi(x) - X_\psi^A(x)||_\alpha^\alpha \leq \varepsilon$: Be $A := \left(\frac{\alpha H}{2^{\alpha+d}\cdot d}\cdot\varepsilon\right)^{-1/\alpha H}$. Then

$$\left|\left|X_\psi(x) - X_\psi^A(x)\right|\right|_\alpha^\alpha \leq \frac{2^{\alpha+d}\cdot d}{\alpha H} \cdot A^{-\alpha H} = \frac{2^{\alpha+d}\cdot d}{\alpha H} \cdot \left(\left(\frac{\alpha H}{2^{\alpha+d}\cdot d}\cdot\varepsilon\right)^{-\frac{1}{\alpha H}}\right)^{-\alpha H} = \varepsilon.$$

Not only in the special case of a norm, but also in the general case of an E^T-homogeneous function ψ in the representation of equation (2.2), an estimation of the error of approximation can be found without using the inequalities of Lemma 2.1 or Definition 2.5. Be the E^T-homogeneous function ψ defined as in equation (2.2), i.e.

$$\psi(x) = \left(\sum_{k=1}^{d} C_k \, |< x, \theta_k >|^{\rho/\lambda_k} \right)^{1/\rho}$$

with linear independent vectors $\theta_1, \ldots, \theta_d \in \mathbb{R}^d$. Be assumed that $0 < \lambda_1 \leq \ldots \leq \lambda_d$ (this is no limitation, as the λ_i have to be positive anyway, and the indices can be chosen according to the ordering of the λ_i), and that $A > 1$.

Without limitation of the possible functions ψ, we may assume that the vectors θ_k, $1 \leq k \leq d$ in this representation of ψ are of length 1 (i.e. $||\theta_k||_2 = 1$), as we show in the following lemma:

Lemma 3.8. *If not $||\theta_k||_2 = 1$ for all $1 \leq k \leq d$, then ψ can be transformed in order to have this property, i.e. ψ can always be represented in such a form, by choosing $C_1, \ldots, C_d$ accordingly. Therefore it can be assumed that $||\theta_k||_2 = 1$ for all $1 \leq k \leq d$.*

Proof. Be $\tilde{\theta}_k := \frac{\theta_k}{||\theta_k||_2}$, so that $\theta_k = ||\theta_k||_2 \cdot \tilde{\theta}_k$ and $||\tilde{\theta}_k||_2 = 1$. Then

$$\begin{aligned}
\psi(x) &= \left(\sum_{k=1}^{d} C_k \, |< x, \theta_k >|^{\rho/\lambda_k} \right)^{1/\rho} \\
&= \left(\sum_{k=1}^{d} C_k \left| < x, ||\theta_k||_2 \cdot \tilde{\theta}_k > \right|^{\rho/\lambda_k} \right)^{1/\rho} \\
&= \left(\sum_{k=1}^{d} C_k ||\theta_k||_2^{\rho/\lambda_k} \cdot \left| < x, \tilde{\theta}_k > \right|^{\rho/\lambda_k} \right)^{1/\rho} \\
&= \left(\sum_{k=1}^{d} \tilde{C}_k \cdot \left| < x, \tilde{\theta}_k > \right|^{\rho/\lambda_k} \right)^{1/\rho}
\end{aligned}$$

with $\tilde{C}_k := C_k ||\theta_k||_2^{\rho/\lambda_k}$, and $||\tilde{\theta}_k||_2 = 1$ for all $k \in \{1, \ldots, d\}$. □

Before estimating an upper bound for the approximation error, we estimate a lower bound for the values of the function ψ (Since now, be $C_{min} := \min\{C_1, \ldots, C_d\}$ and $C_{max} := \max\{C_1, \ldots, C_d\}$).

Lemma 3.9. *Be $C_{min} := \min\{C_1, \dots, C_d\}$, $C_{max} := \max\{C_1, \dots, C_d\}$ and $\theta_\xi := \frac{\xi}{||\xi||_2} \in S_2 := \{\xi \in \mathbb{R}^d : ||\xi||_2 = 1\}$. Then*

(a) If $||\xi||_2 \geq 1$, then $\psi(\xi) \geq C_{min}^{1/\rho} \cdot ||\xi||_2^{1/\lambda_d} \cdot \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\rho/\lambda_1}\right)^{1/\rho}$.

(b) If $||\xi||_2 \leq 1$, then $\psi(\xi) \geq C_{min}^{1/\rho} \cdot ||\xi||_2^{1/\lambda_1} \cdot \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\rho/\lambda_1}\right)^{1/\rho}$.

(Remark: The only difference between the two cases is in the exponent of $||\xi||_2$.)

Proof. From the assumption $\lambda_1 \leq \ldots \leq \lambda_d$ follows $\frac{\rho}{\lambda_1} \geq \frac{\rho}{\lambda_k} \geq \frac{\rho}{\lambda_d}$ for all $1 \leq k \leq d$, and therefore $a^{\rho/\lambda_k} \geq a^{\rho/\lambda_1}$ for any $0 \leq a \leq 1$, and $b^{\rho/\lambda_k} \geq b^{\rho/\lambda_d}$ for $b \geq 1$.

If $||\xi||_2 \geq 1$, then

$$\begin{aligned}
\psi(\xi) &= \left(\sum_{k=1}^d C_k \, |< \xi, \theta_k >|^{\rho/\lambda_k}\right)^{1/\rho} \\
&\geq \left(C_{min} \sum_{k=1}^d |< \xi, \theta_k >|^{\rho/\lambda_k}\right)^{1/\rho} \\
&= C_{min}^{1/\rho} \left(\sum_{k=1}^d |< \theta_\xi \cdot ||\xi||_2, \theta_k >|^{\rho/\lambda_k}\right)^{1/\rho} \\
&= C_{min}^{1/\rho} \left(\sum_{k=1}^d ||\xi||_2^{\rho/\lambda_k} \cdot |< \theta_\xi, \theta_k >|^{\rho/\lambda_k}\right)^{1/\rho} \\
&\geq C_{min}^{1/\rho} \left(\sum_{k=1}^d ||\xi||_2^{\rho/\lambda_d} \cdot |< \theta_\xi, \theta_k >|^{\rho/\lambda_1}\right)^{1/\rho} \\
&= C_{min}^{1/\rho} \cdot ||\xi||_2^{1/\lambda_d} \cdot \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\rho/\lambda_1}\right)^{1/\rho}.
\end{aligned}$$

If $||\xi||_2 \leq 1$, then $\psi(\xi)$ can be estimated analogously:

$$\begin{aligned}
\psi(\xi) &\geq C_{min}^{1/\rho} \left(\sum_{k=1}^d ||\xi||_2^{\rho/\lambda_k} \cdot |< \theta_\xi, \theta_k >|^{\rho/\lambda_k}\right)^{1/\rho} \\
&\geq C_{min}^{1/\rho} \left(\sum_{k=1}^d ||\xi||_2^{\rho/\lambda_1} \cdot |< \theta_\xi, \theta_k >|^{\rho/\lambda_1}\right)^{1/\rho} \\
&= C_{min}^{1/\rho} \cdot ||\xi||_2^{1/\lambda_1} \cdot \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\rho/\lambda_1}\right)^{1/\rho}.
\end{aligned}$$

If $||\xi||_\infty \geq A$, then the first case is relevant, because $||\xi||_2 \geq ||\xi||_\infty \geq A \geq 1$. $\square$

Using these lemmas, the error of truncation can be estimated as follows:

Theorem 3.10. *Be the E^T-homogeneous function ψ defined as in equation (2.2), i.e.*

$$\psi(x) = \left(\sum_{k=1}^{d} C_k \, |< x, \theta_k >|^{\frac{\rho}{\lambda_k}} \right)^{\frac{1}{\rho}}$$

with linear independent vectors $\theta_1, \ldots, \theta_d \in \mathbb{R}^d$. Be assumed that $0 < \lambda_1 \leq \ldots \leq \lambda_d$ (this is no limitation, as the λ_i have to be positive anyway, and the indices can be chosen according to the ordering of the λ_i), that $A > 1$ (which we may assume as we are only interested in error estimates for large values of A), and that $||\theta_k||_2 = 1, \quad 1 \leq k \leq d$ (compare Lemma 3.8). Then

$$\left|\left| X_\psi(x) - X_\psi^A(x) \right|\right|_\alpha^\alpha \leq \widetilde{C} \cdot A^{\frac{-\alpha H - q + d\lambda_d}{\lambda_d}}$$

with

$$\widetilde{C} := 2^\alpha \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \cdot \tilde{I} \cdot \frac{\lambda_d}{\alpha H + q - d\lambda_d}$$

if $d\lambda_d - q < \alpha H$ (using the notation $\tilde{I} := \int_{S_2} \left(\sum_{k=1}^{d} |< \theta_\xi, \theta_k >|^{-\frac{\rho}{\lambda_1}} \right)^{\frac{-H\alpha - q}{\rho}} d\theta_\xi$ with $S_2 := \{\xi \in \mathbb{R}^d : ||\xi||_2 = 1\}$).

Proof. According to Lemma 3.1,

$$\left|\left| X_\psi(x) - X_\psi^A(x) \right|\right|_\alpha^\alpha \leq \int_{\mathbb{R}^d \setminus [-A,A]^d} \left| e^{i<x,\xi>} - 1 \right|^\alpha \psi(\xi)^{-\alpha H - q} \, d\xi$$

which is not more than

$$\begin{aligned}
&\int_{\mathbb{R}^d \setminus [-A,A]^d} \left| e^{i<x,\xi>} - 1 \right|^\alpha \psi(\xi)^{-\alpha H - q} \, d\xi \\
\leq & 2^\alpha \cdot \int_{\mathbb{R}^d \setminus [-A,A]^d} \psi(\xi)^{-\alpha H - q} \, d\xi \\
\leq & 2^\alpha \cdot \int_{\mathbb{R}^d \setminus [-A,A]^d} \left(C_{min}^{\frac{1}{\rho}} \cdot ||\xi||_2^{\frac{1}{\lambda_d}} \cdot \left(\sum_{k=1}^{d} |< \theta_\xi, \theta_k >|^{\frac{\rho}{\lambda_1}} \right)^{\frac{1}{\rho}} \right)^{-\alpha H - q} d\xi \\
\leq & 2^\alpha \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \cdot \int_{\{\xi \in \mathbb{R}^d : ||\xi||_2 \geq A\}} \left(||\xi||_2^{\frac{-\alpha H - q}{\lambda_d}} \cdot \left(\sum_{k=1}^{d} |< \theta_\xi, \theta_k >|^{\frac{\rho}{\lambda_1}} \right)^{\frac{-\alpha H - q}{\rho}} \right) d\xi
\end{aligned}$$

$$
\begin{aligned}
&=2^\alpha \cdot C_{min}^{\frac{-\alpha H-q}{\rho}} \cdot \int_A^\infty \int_{S_2} r^{\frac{-\alpha H-q}{\lambda_d}} \cdot \left(\sum_{k=1}^d |<\theta_\xi,\theta_k>|^{\frac{\rho}{\lambda_1}}\right)^{\frac{-\alpha H-q}{\rho}} d\theta_\xi \cdot r^{d-1}\, dr \\
&=2^\alpha \cdot C_{min}^{\frac{-\alpha H-q}{\rho}} \cdot \int_A^\infty r^{\frac{-\alpha H-q}{\lambda_d}+d-1} dr \cdot \underbrace{\int_{S_2} \left(\sum_{k=1}^d |<\theta_\xi,\theta_k>|^{\frac{\rho}{\lambda_1}}\right)^{\frac{-\alpha H-q}{\rho}} d\theta_\xi}_{:=\tilde I} \\
&=2^\alpha \cdot C_{min}^{\frac{-\alpha H-q}{\rho}} \cdot \frac{\lambda_d}{-\alpha H-q+d\lambda_d} \left[r^{\frac{-\alpha H-q+d\lambda_d}{\lambda_d}} \right]_A^\infty \cdot \tilde I \\
&=2^\alpha \cdot C_{min}^{\frac{-\alpha H-q}{\rho}} \cdot \frac{\lambda_d}{-\alpha H-q+d\lambda_d} \cdot \left(-A^{\frac{-\alpha H-q+d\lambda_d}{\lambda_d}}\right) \cdot \tilde I \\
&=2^\alpha \cdot C_{min}^{\frac{-\alpha H-q}{\rho}} \cdot \tilde I \cdot \frac{\lambda_d}{\alpha H+q-d\lambda_d} \cdot A^{\frac{-\alpha H-q+d\lambda_d}{\lambda_d}}
\end{aligned}
$$

if the exponent $\frac{-\alpha H-q+d\lambda_d}{\lambda_d}$ is negative, i.e. if $-\alpha H-q+d\lambda_d<0 \Leftrightarrow d\lambda_d-q<\alpha H$. □

Remark 3.11. Because the exponent of A is assumed to be negative, the error estimate converges to zero for $A \to \infty$.

Corollary 3.12. *If all eigenvalues of the matrix E are equal, i.e. $\lambda_1 = \ldots = \lambda_d$, then $q = d\lambda_1$, and the term of the estimation is reduced to*

$$
\left|\left|X_\psi(x)-X_\psi^A(x)\right|\right|_\alpha^\alpha \le 2^\alpha \cdot C_{min}^{\frac{-\alpha H-d\lambda_1}{\rho}} \cdot \tilde I \cdot \frac{\lambda_1}{\alpha H} \cdot A^{\frac{-\alpha H}{\lambda_1}}.
$$

The integral $\tilde I = \int_{S_2} \left(\sum_{k=1}^d |<\theta_\xi,\theta_k>|^{\frac{\rho}{\lambda_1}}\right)^{\frac{-\alpha H-q}{\rho}} d\theta_\xi$ is dependent only on the parameters of the approximated random field X (like H and the parameters of the function ψ), but independent of the approximation parameter A. As no estimation in a simpler form was found for if $d \ge 3$, it has to be approximated numerically in each concrete case (which can be done in a sufficient precision on a modern computer within a fraction of a second). For the case $d = 2$, the integral can also be estimated as follows (however, the numerical approximation of the integral gives a much better result also in this case):

Corollary 3.13. *In the two-dimensional case (if $d = 2$), the Integral $\tilde I$ can be estimated by $\tilde I \le 2\pi \cdot 2^{\frac{\alpha H+q}{\lambda_1}} \cdot (1-|<\theta_1,\theta_2>|^2)^{\frac{-\alpha H-q}{2\lambda_1}}$, so that the inequality*

$$
\begin{aligned}
&||X_\psi(x)-X_\psi^A(x)||_\alpha^\alpha \\
&\le C_{min}^{\frac{-\alpha H-q}{\rho}} \cdot 2^{\frac{\alpha H+q}{\lambda_1}} \cdot \left(1-|<\theta_1,\theta_2>|^2\right)^{\frac{-\alpha H-q}{2\lambda_1}} \cdot \frac{2^\alpha \cdot 2\pi \cdot \lambda_2}{\alpha H+\lambda_1-\lambda_2} A^{\frac{\alpha H-\lambda_1+\lambda_2}{\lambda_2}}
\end{aligned}
$$

follows (provided that the exponent of A is negative).

Proof. Be α_1 the angle between the vectors θ_ξ and θ_1, α_2 the angle between the vectors θ_ξ and θ_2, and α_{12} the angle between θ_1 and θ_2. Then (using the fact that $||\theta_1||_2 = ||\theta_2||_2 = ||\theta_\xi||_2 = 1$) it follows that

$$| < \theta_\xi, \theta_1 > | = |\cos(\alpha_1)|, \qquad | < \theta_\xi, \theta_2 > | = |\cos(\alpha_2)|, \qquad | < \theta_1, \theta_2 > | = |\cos(\alpha_{12})|$$

Then

$$\begin{aligned}
&\left(|< \theta_\xi, \theta_1 >|^{\rho/\lambda_1} + |< \theta_\xi, \theta_2 >|^{\rho/\lambda_1}\right)^{1/\rho} \\
\geq &\left(\max\{|< \theta_\xi, \theta_1 >|^{\rho/\lambda_1}, |< \theta_\xi, \theta_2 >|^{\rho/\lambda_1}\}\right)^{1/\rho} \\
= &\left((\max\{|< \theta_\xi, \theta_1 >|, |< \theta_\xi, \theta_2 >|\})^{\rho/\lambda_1}\right)^{1/\rho} \\
= &(\max\{|\cos(\alpha_1)|, |\cos(\alpha_2)|\})^{1/\lambda_1} \\
= &(\max\{|\cos(\alpha_1)|, |\cos(\alpha_1 + \alpha_{12})|\})^{1/\lambda_1} \\
\geq &\min_{\beta \in \mathbb{R}} \left\{(\max\{|\cos(\beta)|, |\cos(\beta + \alpha_{12})|\})^{1/\lambda_1}\right\} \\
= &\min_{\beta \in \mathbb{R}} \left\{(\max\{|\cos(\beta - \pi/2)|, |\cos(\beta + \alpha_{12} - \pi/2)|\})^{1/\lambda_1}\right\} \\
= &\min_{\beta \in \mathbb{R}} \left\{(\max\{|\sin(\beta)|, |\sin(\beta + \alpha_{12})|\})^{1/\lambda_1}\right\} \\
= &(\max\{|\sin(-\alpha_{12}/2)|, |\sin(\alpha_{12}/2)|\})^{1/\lambda_1} \\
= &\left|\sin\left(\frac{\alpha_{12}}{2}\right)\right|^{\frac{1}{\lambda_1}} \geq \left(\frac{|\sin(\alpha_{12})|}{2}\right)^{\frac{1}{\lambda_1}} = 2^{-\frac{1}{\lambda_1}} |\sin(\alpha_{12})|^{\frac{1}{\lambda_1}} \\
= &2^{-\frac{1}{\lambda_1}} \cdot \left(\sqrt{1 - \cos^2(\alpha_{12})}\right)^{\frac{1}{\lambda_1}} = 2^{-\frac{1}{\lambda_1}} \cdot \left(1 - | < \theta_1, \theta_2 > |^2\right)^{\frac{1}{2\lambda_1}}
\end{aligned}$$

which implies

$$\begin{aligned}
\tilde{I} &= \int_{S_2} \left(|< \theta_\xi, \theta_1 >|^{\frac{\rho}{\lambda_1}} + |< \theta_\xi, \theta_2 >|^{\frac{\rho}{\lambda_1}}\right)^{\frac{-\alpha H - q}{\rho}} \, d\theta_\xi \\
&\leq \int_{S_2} \left(2^{-\frac{1}{\lambda_1}} \cdot \left(1 - | < \theta_1, \theta_2 > |^2\right)^{\frac{1}{2\lambda_1}}\right)^{-\alpha H - q} \, d\theta_\xi \\
&\leq 2^{\frac{\alpha H + q}{\lambda_1}} \cdot \left(1 - | < \theta_1, \theta_2 > |^2\right)^{\frac{-\alpha H - q}{2\lambda_1}} \cdot \int_{S_2} 1 \, d\theta_\xi \\
&\leq 2^{\frac{\alpha H + q}{\lambda_1}} \cdot \left(1 - | < \theta_1, \theta_2 > |^2\right)^{\frac{-\alpha H - q}{2\lambda_1}} \cdot 2\pi
\end{aligned}$$

□

For a numerical approximation of the integral $\tilde{I}$, be $\nu = \frac{\rho}{\lambda_1}$ and $\gamma = \frac{-H\alpha - q}{\rho}$. Then

$$\tilde{I} = \int_{S_2} \left(|< \theta_\xi, \theta_1 >|^\nu + |< \theta_\xi, \theta_2 >|^\nu\right)^\gamma \; d\theta_\xi$$

which can be approximated (with n support points) by the sum

$$\tilde{S}_n := \frac{2\pi}{n} \cdot \sum_{k=1}^{n} \left(\left|< (\cos(\zeta_k), \sin(\zeta_k))^T, \theta_1 >\right|^\nu + \left|< (\cos(\zeta_k), \sin(\zeta_k))^T, \theta_2 >\right|^\nu\right)^\gamma$$

with $\zeta_k := \frac{2\pi}{n} \cdot k$.

Table 3.1 shows approximated values of $\tilde{I}$ for the parameter values $\nu = 1.0$ and $\gamma = -3.0$. One of the vectors which are parameters to the integral is set to $\theta_1 = (1, 0)^T$, and the other is $\theta_2 = (\cos(v_2), \sin(v_2))^T$ for different values of v_2 ($v_2 \in \{\frac{1}{20}\pi, \ldots, \frac{19}{20}\pi\}$). Thereby, $\hat{I}$ is the estimation of $\tilde{I}$ according to Corollary 3.13, while $\tilde{S}_{4k}$ and $\tilde{S}_{60k}$ are the numerical approximations of $\tilde{I}$ with 4000 and 60000 points of support.

From this table, several conclusions can be drawn (for the used sample of parameter values):

(a) The difference of the approximations $\tilde{S}_{4k}$ and $\tilde{S}_{60k}$ are relatively small. Thus it may be assumed that only a few thousand (e.g. 4000) points of support are necessary to obtain a sufficient approximation of the integral for most purposes.

(b) The estimation of $\tilde{I}$ in Corollary 3.13 overestimates the integral by a large factor.

(c) The values of $\tilde{S}_{60k} \cdot \sin(v_2)^2$ didn't show large differences for different values of v_2. They were all in the interval $[3.0, 3.25]$. Therefore, the term $3.25 \cdot \sin(v_2)^{-2}$ seems to be a relatively good approximation for $\tilde{I}$ with the parameters $\nu = 1.0$ and $\gamma = -3.0$. A comparison with similar tables of numerical approximations for other values of γ and ν suggests that, generally, $\tilde{I}$ may be estimated quite well by $\tilde{I} \leq C_{\gamma,\nu} \cdot \sin(v_2)^{\gamma\nu+1}$, where $C_{\gamma,\nu}$ is a number which depends on γ and ν (but not on v_2).

v_2	$\hat{I}$	$\tilde{S}_{4k}$	$\tilde{S}_{60k}$	$\tilde{S}_{60k} \cdot \sin(v_2)^2$
$0.05 \cdot \pi$	13130.2360	123.2474	123.2392	3.0159
$0.10 \cdot \pi$	1703.4240	31.9048	31.9043	3.0466
$0.15 \cdot \pi$	537.1920	14.9577	14.9576	3.0829
$0.20 \cdot \pi$	247.5220	9.0309	9.0309	3.1201
$0.25 \cdot \pi$	142.1723	6.3106	6.3106	3.1553
$0.30 \cdot \pi$	94.9286	4.8684	4.8684	3.1864
$0.35 \cdot \pi$	71.0603	4.0458	4.0458	3.2120
$0.40 \cdot \pi$	58.4320	3.5720	3.5720	3.2309
$0.45 \cdot \pi$	52.1687	3.3239	3.3239	3.2425
$0.50 \cdot \pi$	50.2655	3.2465	3.2465	3.2465
$0.55 \cdot \pi$	52.1687	3.3239	3.3239	3.2425
$0.60 \cdot \pi$	58.4320	3.5720	3.5720	3.2309
$0.65 \cdot \pi$	71.0603	4.0458	4.0458	3.2120
$0.70 \cdot \pi$	94.9286	4.8684	4.8684	3.1864
$0.75 \cdot \pi$	142.1723	6.3106	6.3106	3.1553
$0.80 \cdot \pi$	247.5220	9.0309	9.0309	3.1201
$0.85 \cdot \pi$	537.1920	14.9577	14.9576	3.0829
$0.90 \cdot \pi$	1703.4240	31.9048	31.9043	3.0466
$0.95 \cdot \pi$	13130.2360	123.2474	123.2392	3.0159

Table 3.1: Estimation and numerical approximation of $\tilde{I}$ with $\nu = 1.0$ and $\gamma = -3.0$, where v_2 is the angle between the vectors θ_1 and θ_2. $\hat{I}$ is the estimation of $\tilde{I}$ according to Corollary 3.13, while $\tilde{S}_{4k}$ and $\tilde{S}_{60k}$ are the numerical approximations of $\tilde{I}$ with 4000 and 60000 points of support.

3.3 Approximation error due to the discretisation

3.3.1 Approximation error if ψ is a ρ-norm

First version of the discretisation

In the first step of approximation, the random field $X_\psi(x)$ was approximated by

$$X_\psi^A(x) = \operatorname{Re} \int_{[-A,A]^d} \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}}\, W_\alpha(d\xi). \tag{3.6}$$

Now, this is approximated by the sum

$$X_\psi^{A,M}(x) = \operatorname{Re} \sum_{\vec{\mathbf{k}} \in J} \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>} - 1\right) \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\, W_\alpha(\Delta_{\vec{\mathbf{k}}}) \tag{3.7}$$

with $J := \{-M, \dots, M-1\}^d \backslash \{-1, 0\}^d$, $\xi_{\vec{\mathbf{k}}} := D \cdot \vec{\mathbf{k}} = (D \cdot k_1, \dots, D \cdot k_d)^T$ and $\Delta_{\vec{\mathbf{k}}} := [k_1 \cdot D, (k_1+1) \cdot D) \times \dots \times [k_d \cdot D, (k_d+1) \cdot D)$ for each $\vec{\mathbf{k}} = (k_1, \dots, k_d)^T \in J$. Written a bit different, this means that

$$X_\psi^{A,M}(x) = \operatorname{Re} \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} g_{\vec{\mathbf{k}}}\, W_\alpha(\Delta_{\vec{\mathbf{k}}}) \tag{3.8}$$

with

$$g_{\vec{\mathbf{k}}} = \begin{cases} 0, & \vec{\mathbf{k}} \in \{-1, 0\}^d \\ \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>} - 1\right) \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}, & \text{else} \end{cases}. \tag{3.9}$$

Like the error of approximation which is caused by the truncation of the domain of integration, the error of approximation due to the discretisation of the integrand function $(X_\psi^A(x) - X_\psi^{A,M}(x))$ is a symmetric α-stable random variable, too. In this section, an estimation of the scale parameter $||X_\psi^A(x) - X_\psi^{A,M}(x)||_\alpha$ will be calculated:

Lemma 3.14. *For $X_\psi^{A,M}(x)$ defined as in equation* (3.8)*, the scale parameter of the error random variable can be estimated by*

$$||X_\psi^A(x) - X_\psi^{A,M}(x)||_\alpha^\alpha \leq \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}} \right|^\alpha d\xi \tag{3.10}$$

Proof.

$$
\begin{aligned}
&||X_\psi^A(x) - X_\psi^{A,M}(x)||_\alpha^\alpha \\
&= \left|\left| \mathrm{Re} \int_{[-A,A]^d} \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}} W_\alpha(d\xi) - \mathrm{Re} \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} g_{\vec{\mathbf{k}}} W_\alpha(\Delta_{\vec{\mathbf{k}}}) \right|\right|_\alpha^\alpha \\
&\underset{\mathrm{Re}}{\le} \left|\left| \int_{[-A,A]^d} \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(\xi) \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}} W_\alpha(d\xi) \right.\right. \\
&\qquad \left.\left. - \int_{[-A,A]^d} \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(\xi) g_{\vec{\mathbf{k}}} W_\alpha(d\xi) \right|\right|_\alpha^\alpha \\
&= \left|\left| \int_{[-A,A]^d} \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(\xi) \left(\left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}} \right) W_\alpha(d\xi) \right|\right|_\alpha^\alpha \\
&= \int_{[-A,A]^d} \left| \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(\xi) \left(\left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}} \right) \right|^\alpha d\xi \\
&= \int_{[-A,A]^d} \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(\xi) \left| \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}} \right|^\alpha d\xi \\
&= \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}} \right|^\alpha d\xi
\end{aligned}
$$

□

Lemma 3.15. *With $g_{\vec{\mathbf{k}}}$ being defined according to* (3.9), *the term on the right side of* (3.10) *in the previous lemma can be estimated further by*

$$
\sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}} \right|^\alpha d\xi \le I_0 + I_1 + I_2 \tag{3.11}
$$

with

$$
\begin{aligned}
I_0 &= ||x||_2^\alpha \cdot \int_{[-D,D]^d} ||\xi||_2^\alpha \cdot \psi(\xi)^{-\alpha H - q} d\xi \\
I_1 &= 2 \cdot ||x||_2^\alpha \cdot D^\alpha d^{\alpha/2} \cdot \int_{[-A,A]^d \setminus [-D,D]^d} \psi(\xi)^{-\alpha H - q} d\xi \\
I_2 &= 2^{1+\alpha} \cdot \sum_{\vec{\mathbf{k}} \in J} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}} \right|^\alpha d\xi
\end{aligned}
$$

Proof. Using the definition of $g_{\vec{\mathbf{k}}}$ in (3.9), the right side of (3.10) is equal to

$$
\begin{aligned}
&\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}}\right|^\alpha d\xi \\
=&\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right)\psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\cdot 1_J(\vec{\mathbf{k}})\right|^\alpha d\xi \\
=&\sum_{\vec{\mathbf{k}}\in\{-1,0\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}}\right|^\alpha d\xi \\
&\quad +\sum_{\vec{\mathbf{k}}\in J} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right)\psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right|^\alpha d\xi
\end{aligned}
$$

The absolute value in the second line can be transformed and estimated with the triangle inequality as follows:

$$
\begin{aligned}
&\left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right)\psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right| \\
=&\Big|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} \\
&\quad + \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right)\psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\Big| \\
\leq&\left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}}\right| \\
&\quad + \left|\left(e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right)\psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right| \\
=&\left|\left(e^{i<x,\xi>}-e^{i<x,\xi_{\vec{\mathbf{k}}}>}\right)\psi(\xi)^{-H-\frac{q}{\alpha}}\right| + \left|\left(e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right)\left(\psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right)\right|
\end{aligned}
$$

As $\alpha \in (0,2]$, this implies:

$$
\begin{aligned}
&\left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right)\psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right|^\alpha \\
\leq&\left(\left|\left(e^{i<x,\xi>}-e^{i<x,\xi_{\vec{\mathbf{k}}}>}\right)\psi(\xi)^{-H-\frac{q}{\alpha}}\right| + \left|\left(e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right)\left(\psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right)\right|\right)^\alpha \\
\leq&2\cdot\left(\left|\left(e^{i<x,\xi>}-e^{i<x,\xi_{\vec{\mathbf{k}}}>}\right)\psi(\xi)^{-H-\frac{q}{\alpha}}\right|^\alpha + \left|\left(e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right)\left(\psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right)\right|^\alpha\right) \\
=&2\cdot\left|e^{i<x,\xi>}-e^{i<x,\xi_{\vec{\mathbf{k}}}>}\right|^\alpha \psi(\xi)^{-\alpha H-q} + 2\cdot\left|e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right|^\alpha \left|\psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right|^\alpha
\end{aligned}
$$

Therefore

$$\begin{aligned}
&\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}}\right|^\alpha d\xi \\
=&\sum_{\vec{\mathbf{k}}\in\{-1,0\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}}\right|^\alpha d\xi \\
&\quad+\sum_{\vec{\mathbf{k}}\in J} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right)\psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right|^\alpha d\xi \\
\leq&\int_{[-D,D]^d} \left|e^{i<x,\xi>}-1\right|^\alpha \psi(\xi)^{-\alpha H-q}\, d\xi \\
&\quad+2\cdot\sum_{\vec{\mathbf{k}}\in J} \int_{\Delta_{\vec{\mathbf{k}}}} \left|e^{i<x,\xi>}-e^{i<x,\xi_{\vec{\mathbf{k}}}>}\right|^\alpha \psi(\xi)^{-\alpha H-q}\, d\xi \\
&\quad+2\cdot\sum_{\vec{\mathbf{k}}\in J} \int_{\Delta_{\vec{\mathbf{k}}}} \left|e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1\right|^\alpha \left|\psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right|^\alpha d\xi
\end{aligned}$$

Because $|e^{it}-1|\leq 2$ and $|e^{it}-1|\leq |t|$ for all $t\in\mathbb{R}$, the inequalities

$$\begin{aligned}
|e^{i<x,\xi>}-1| &\leq |<x,\xi>| \leq ||x||_2\cdot||\xi||_2, \\
|e^{i<x,\xi>}-e^{i<x,\xi_{\vec{\mathbf{k}}}>}| = |e^{i<x,\xi-\xi_{\vec{\mathbf{k}}}>}-1| &\leq ||x||_2\cdot||\xi-\xi_{\vec{\mathbf{k}}}||_2 \leq ||x||_2\cdot D\sqrt{d}, \\
|e^{i<x,\xi_{\vec{\mathbf{k}}}>}-1| &\leq 2
\end{aligned}$$

hold, so that

$$\begin{aligned}
&\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}}\right|^\alpha d\xi \\
\leq&||x||_2^\alpha\cdot\int_{[-D,D]^d} ||\xi||_2^\alpha\cdot\psi(\xi)^{-\alpha H-q}d\xi \\
&+2\cdot||x||_2^\alpha\cdot D^\alpha d^{\frac{\alpha}{2}}\cdot\sum_{\vec{\mathbf{k}}\in J}\int_{\Delta_{\vec{\mathbf{k}}}} \psi(\xi)^{-\alpha H-q}d\xi \\
&+2\cdot 2^\alpha\cdot\sum_{\vec{\mathbf{k}}\in J}\int_{\Delta_{\vec{\mathbf{k}}}} \left|\psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right|^\alpha d\xi
\end{aligned}$$

Because the $\Delta_{\vec{\mathbf{k}}}$ are disjoint, and $\sum_{\vec{\mathbf{k}}\in J}\Delta_{\vec{\mathbf{k}}} = [-A,A]^d\backslash[-D,D]^d$, the second term can be written also as

$$2\cdot||x||_2^\alpha\cdot D^\alpha d^{\frac{\alpha}{2}}\cdot\int_{[-A,A]^d\backslash[-D,D]^d} \psi(\xi)^{-\alpha H-q}d\xi$$

□

Theorem 3.16. *Be ψ the ρ-norm (in the representation* (2.2) *of ψ, the parameters are chosen as $C_1 = \ldots = C_d = 1, \lambda_1 = \ldots = \lambda_d = 1$, and the vectors $\theta_1, \ldots, \theta_d$ form the standard base of the $\mathbb{R}^d$). Be also assumed that $0 < H < 1$. Then the approximation error due to the discretisation can be estimated by*

$$||X_\psi^A(x) - X_\psi^{A,M}(x)||_\alpha^\alpha \leq \left(\widetilde{C}_{0,x} + \widetilde{C}_{1,x}\right) \cdot D^{\alpha(1-H)} + \widetilde{C}_2 \cdot D^{-\alpha H}$$

with

$$\begin{aligned}\widetilde{C}_{0,x} &= ||x||_2^\alpha \cdot \frac{d^{1+\frac{\alpha}{2}} \cdot 2^d}{\alpha(1-H)} \\ \widetilde{C}_{1,x} &= ||x||_2^\alpha \cdot \frac{d^{1+\frac{\alpha}{2}} \cdot 2^{d+1}}{\alpha H} \\ \widetilde{C}_2 &= 2^{\alpha+2} \cdot d^{1+\alpha} \cdot \left(H + \frac{d}{\alpha}\right)^\alpha \cdot \left(4^{d-1} + \frac{3^{d-1}}{\alpha(1+H)}\right)\end{aligned}$$

Proof. Lemma 3.15 implies for this specific choice of ψ, that

$$\sum_{\vec{\mathbf{k}} \in \{-M,\ldots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}} \right|^\alpha d\xi \leq I_0 + I_1 + I_2$$

with

$$\begin{aligned}I_0 &= ||x||_2^\alpha \cdot \int_{[-D,D]^d} ||\xi||_2^\alpha \cdot ||\xi||_\rho^{-\alpha H - d} d\xi \\ I_1 &= 2 \cdot ||x||_2^\alpha \cdot D^\alpha d^{\frac{\alpha}{2}} \cdot \int_{[-A,A]^d \setminus [-D,D]^d} ||\xi||_\rho^{-\alpha H - d} d\xi \\ I_2 &= 2^{1+\alpha} \cdot \sum_{\vec{\mathbf{k}} \in J} \int_{\Delta_{\vec{\mathbf{k}}}} \left| ||\xi||_\rho^{-H-\frac{d}{\alpha}} - ||\xi_{\vec{\mathbf{k}}}||_\rho^{-H-\frac{d}{\alpha}} \right|^\alpha d\xi\end{aligned}$$

(the identity $q = d$ follows from the choice of the parameters $\lambda_1, \ldots, \lambda_d$).

The term I_0 can be estimated as follows (under the assumption $0 < H < 1$):

$$\begin{aligned}
I_0 &= ||x||_2^\alpha \cdot \int_{[-D,D]^d} ||\xi||_2^\alpha \cdot ||\xi||_\rho^{-\alpha H-d} d\xi \\
&\leq ||x||_2^\alpha \cdot \int_{[-D,D]^d} \left(\sqrt{d} \cdot ||\xi||_\infty\right)^\alpha ||\xi||_\infty^{-\alpha H-d} d\xi \\
&= ||x||_2^\alpha \cdot d^{\frac{\alpha}{2}} \cdot \int_{[-D,D]^d} ||\xi||_\infty^{\alpha-\alpha H-d} d\xi \\
&= ||x||_2^\alpha \cdot d^{\frac{\alpha}{2}} \cdot \int_0^D \int_{S_\infty} ||r \cdot \zeta||_\infty^{\alpha-\alpha H-d}\, d\zeta \cdot r^{d-1}\, dr \\
&= ||x||_2^\alpha \cdot d^{\frac{\alpha}{2}} \cdot \int_0^D \int_{S_\infty} 1\, d\zeta \cdot\; r^{\alpha-\alpha H-d+d-1}\, dr \\
&\underset{L.3.3}{=} ||x||_2^\alpha \cdot d^{\frac{\alpha}{2}} \cdot d \cdot 2^d \cdot \int_0^D r^{\alpha-\alpha H-1}\, dr \\
&= ||x||_2^\alpha \cdot d^{1+\frac{\alpha}{2}} \cdot 2^d \cdot \left[\frac{1}{\alpha-\alpha H} \cdot r^{\alpha-\alpha H}\right]_0^D \\
&= ||x||_2^\alpha \cdot \frac{d^{1+\frac{\alpha}{2}} \cdot 2^d}{\alpha(1-H)} \cdot D^{\alpha(1-H)} = \widetilde{C}_{0,x} \cdot D^{\alpha(1-H)}
\end{aligned}$$

with $\widetilde{C}_{0,x} := ||x||_2^\alpha \cdot \frac{d^{1+\alpha/2} \cdot 2^d}{\alpha(1-H)}$.

The term I_1 is estimated similarly: The integral is

$$\begin{aligned}
&\int_{[-A,A]^d \setminus [-D,D]^d} ||\xi||_\rho^{-\alpha H-d}\, d\xi \\
&\leq \int_{[-A,A]^d \setminus [-D,D]^d} ||\xi||_\infty^{-\alpha H-d}\, d\xi \quad = \quad \int_D^A \int_{S_\infty} ||r \cdot \zeta||_\infty^{-\alpha H-d} d\zeta \cdot r^{d-1}\, dr \\
&= \int_D^A \int_{S_\infty} 1\, d\zeta \cdot r^{-\alpha H-d+d-1}\, dr \quad = \quad d \cdot 2^d \cdot \int_D^A r^{-\alpha H-1}\, dr \\
&= d \cdot 2^d \cdot \left[\frac{1}{-\alpha H} \cdot r^{-\alpha H}\right]_D^A \quad = \quad \frac{d \cdot 2^d}{\alpha H}\left(D^{-\alpha H} - A^{-\alpha H}\right)
\end{aligned}$$

Therefore

$$\begin{aligned}
I_1 &= 2 \cdot ||x||_2^\alpha \cdot D^\alpha d^{\frac{\alpha}{2}} \cdot \int_{[-A,A]^d \setminus [-D,D]^d} ||\xi||_\rho^{-\alpha H - d} d\xi \\
&\leq 2 \cdot ||x||_2^\alpha \cdot D^\alpha d^{\frac{\alpha}{2}} \cdot \frac{d \cdot 2^d}{\alpha H} \left(D^{-\alpha H} - A^{-\alpha H} \right) \\
&= ||x||_2^\alpha \cdot D^\alpha \cdot \frac{d^{1+\frac{\alpha}{2}} \cdot 2^{d+1}}{\alpha H} \left(D^{-\alpha H} - A^{-\alpha H} \right) \\
&\leq ||x||_2^\alpha \cdot \frac{d^{1+\frac{\alpha}{2}} \cdot 2^{d+1}}{\alpha H} \cdot D^{\alpha - \alpha H} \\
&= \widetilde{C}_{1,x} \cdot D^{\alpha(1-H)}
\end{aligned}$$

with $\widetilde{C}_{1,x} := ||x||_2^\alpha \cdot \frac{d^{1+\alpha/2} \cdot 2^{d+1}}{\alpha H}$.

Finally, the term $I_2 = 2^{1+\alpha} \cdot \sum_{\vec{\mathbf{k}} \in J} \int_{\Delta_{\vec{\mathbf{k}}}} \left| ||\xi||_\rho^{-H-\frac{d}{\alpha}} - ||\xi_{\vec{\mathbf{k}}}||_\rho^{-H-\frac{d}{\alpha}} \right|^\alpha d\xi$ is estimated. The Mean Value Theorem implies for $a, b > 0$ that

$$\left| a^{-H-\frac{d}{\alpha}} - b^{-H-\frac{d}{\alpha}} \right| \leq |a - b| \cdot \left| -H - \frac{d}{\alpha} \right| \cdot m^{-H-\frac{d}{\alpha}-1}$$

with a $m > 0$ which fulfills $m^{-H-\frac{d}{\alpha}-1} \geq \max(a^{-H-\frac{d}{\alpha}-1}, b^{-H-\frac{d}{\alpha}-1})$, i.e. $m \leq \min(a, b)$. Be now

$$\mu_{\vec{\mathbf{k}}} := \inf_{\xi \in \Delta_{\vec{\mathbf{k}}}} (||\xi||_\rho). \tag{3.12}$$

Then, for all $\xi \in \Delta_{\vec{\mathbf{k}}}$, we have $\mu_{\vec{\mathbf{k}}}^{-H-\frac{d}{\alpha}-1} \geq ||\xi||_\rho^{-H-\frac{d}{\alpha}-1}$ (note that also $\xi_{\vec{\mathbf{k}}} \in \Delta_{\vec{\mathbf{k}}}$) and it follows that

$$\begin{aligned}
\left| ||\xi||_\rho^{-H-\frac{d}{\alpha}} - ||\xi_{\vec{\mathbf{k}}}||_\rho^{-H-\frac{d}{\alpha}} \right| &\leq \left| ||\xi||_\rho - ||\xi_{\vec{\mathbf{k}}}||_\rho \right| \cdot |-H - d/\alpha| \cdot \mu_{\vec{\mathbf{k}}}^{-H-\frac{d}{\alpha}-1} \\
&\leq d \cdot D \cdot (H + d/\alpha) \cdot \mu_{\vec{\mathbf{k}}}^{-H-\frac{d}{\alpha}-1}
\end{aligned}$$

(using the fact that $||\xi||, ||\xi_{\vec{\mathbf{k}}}|| \in \Delta_{\vec{\mathbf{k}}}, \rho \geq 1 \Rightarrow \left| ||\xi||_\rho - ||\xi_{\vec{\mathbf{k}}}||_\rho \right| \leq ||\xi - \xi_{\vec{\mathbf{k}}}||_\rho \leq d \cdot D$).

With this inequality we obtain

$$\begin{aligned}
I_2 &= 2^{1+\alpha} \cdot \sum_{\vec{\mathbf{k}} \in J} \int_{\Delta_{\vec{\mathbf{k}}}} \left| ||\xi||_\rho^{-H-\frac{d}{\alpha}} - ||\xi_{\vec{\mathbf{k}}}||_\rho^{-H-\frac{d}{\alpha}} \right|^\alpha d\xi \\
&\leq 2^{1+\alpha} \cdot \sum_{\vec{\mathbf{k}} \in J} \int_{\Delta_{\vec{\mathbf{k}}}} \left(d \cdot D \cdot (H + d/\alpha) \cdot \mu_{\vec{\mathbf{k}}}^{-H-\frac{d}{\alpha}-1} \right)^\alpha d\xi \\
&= 2^{1+\alpha} \cdot d^\alpha \cdot D^\alpha \cdot (H + d/\alpha)^\alpha \cdot \sum_{\vec{\mathbf{k}} \in J} \int_{\Delta_{\vec{\mathbf{k}}}} \mu_{\vec{\mathbf{k}}}^{-\alpha H - d - \alpha} d\xi \\
&= 2^{1+\alpha} \cdot d^\alpha \cdot D^\alpha \cdot (H + d/\alpha)^\alpha \cdot D^d \cdot \sum_{\vec{\mathbf{k}} \in J} \mu_{\vec{\mathbf{k}}}^{-\alpha H - d - \alpha}
\end{aligned}$$

$\mu_{\vec{\mathbf{k}}}$ is defined as $\mu_{\vec{\mathbf{k}}} = \inf_{\xi \in \Delta_{\vec{\mathbf{k}}}}(||\xi||_\rho)$, i.e. $\mu_{\vec{\mathbf{k}}} = ||\tilde{\xi}_{\vec{\mathbf{k}}}||_\rho$ where $\tilde{\xi}_{\vec{\mathbf{k}}}$ is the corner of $\Delta_{\vec{\mathbf{k}}}$ which is nearest to the origin. From $\Delta_{\vec{\mathbf{k}}} = [k_1 \cdot D, (k_1 + 1) \cdot D) \times \ldots \times [k_d \cdot D, (k_d + 1) \cdot D)$ it follows that

$$\tilde{\xi}_{\vec{\mathbf{k}}} = \left(\tilde{k}_1 \cdot D, \ldots, \tilde{k}_d \cdot D \right)^T = D \cdot \left(\tilde{k}_1, \ldots, \tilde{k}_d \right)^T = D \cdot t(\vec{\mathbf{k}}) \tag{3.13}$$

with $t(\vec{\mathbf{k}}) = \left(\tilde{k}_1, \ldots, \tilde{k}_d \right)^T$ and (for $j \in \{1, \ldots, d\}$)

$$\tilde{k}_j = \begin{cases} k_j & , \ k_j \geq 0 \\ k_j + 1 & , \ k_j < 0 \end{cases} \tag{3.14}$$

Therefore

$$\sum_{\vec{\mathbf{k}} \in J} \mu_{\vec{\mathbf{k}}}^{-\alpha H - d - \alpha} = \sum_{\vec{\mathbf{k}} \in J} ||\tilde{\xi}_{\vec{\mathbf{k}}}||_\rho^{-\alpha H - d - \alpha} = D^{-\alpha H - d - \alpha} \cdot \sum_{\vec{\mathbf{k}} \in J} ||t(\vec{\mathbf{k}})||_\rho^{-\alpha H - d - \alpha} \tag{3.15}$$

If the set of index vectors J is divided into 2^d parts according to the sign of the components of these vectors $\vec{\mathbf{k}} \in J$, and two of these sets are compared, e.g. $J' := \{\vec{\mathbf{k}} \in J : k_j \geq 0, \ 1 \leq j \leq d\} = \{0, \ldots, M-1\}^d \backslash \{0\}^d$ and $J'' := \{\vec{\mathbf{k}} \in J : k_j < 0, \ 1 \leq j \leq d\} = \{-M, \ldots, -1\}^d \backslash \{-1\}^d$, it can be shown that the sums of $||t(\vec{\mathbf{k}})||_\rho^{-\alpha H - d - \alpha}$ on these sets are equal:

$$\begin{aligned}
\sum_{\vec{\mathbf{k}}\in J''} ||t(\vec{\mathbf{k}})||_\rho^{-\alpha H-d-\alpha} &= \sum_{\vec{\mathbf{k}}\in\{-M,\dots,-1\}^d\setminus\{-1\}^d} \left(|\tilde{k}_1|^\rho+\ldots+|\tilde{k}_d|^\rho\right)^{\frac{-\alpha H-d-\alpha}{\rho}} \\
&\underset{(3.14)}{=} \sum_{\vec{\mathbf{k}}\in\{-M,\dots,-1\}^d\setminus\{-1\}^d} \left(|k_1+1|^\rho+\ldots+|k_d+1|^\rho\right)^{\frac{-\alpha H-d-\alpha}{\rho}} \\
&= \sum_{\vec{\mathbf{k}}\in\{-M+1,\dots,0\}^d\setminus\{0\}^d} \left(|k_1|^\rho+\ldots+|k_d|^\rho\right)^{\frac{-\alpha H-d-\alpha}{\rho}} \\
&= \sum_{\vec{\mathbf{k}}\in\{0,\dots,M-1\}^d\setminus\{0\}^d} \left(|k_1|^\rho+\ldots+|k_d|^\rho\right)^{\frac{-\alpha H-d-\alpha}{\rho}} \\
&= \sum_{\vec{\mathbf{k}}\in J'} \left(|\tilde{k}_1|^\rho+\ldots+|\tilde{k}_d|^\rho\right)^{\frac{-\alpha H-d-\alpha}{\rho}} \\
&= \sum_{\vec{\mathbf{k}}\in J'} ||t(\vec{\mathbf{k}})||_\rho^{-\alpha H-d-\alpha}
\end{aligned}$$

For other parts of J, this equality can be shown analogously. With these equalities, and the fact that $\forall\, k\geq 0 : \tilde{k}=k \Rightarrow \forall\,\vec{\mathbf{k}}\in J' : t(\vec{\mathbf{k}})=\vec{\mathbf{k}}$, it follows that

$$\sum_{\vec{\mathbf{k}}\in J} ||t(\vec{\mathbf{k}})||_\rho^{-\alpha H-d-\alpha} = 2^d\cdot\sum_{\vec{\mathbf{k}}\in J'} ||t(\vec{\mathbf{k}})||_\rho^{-\alpha H-d-\alpha} = 2^d\cdot\sum_{\vec{\mathbf{k}}\in J'} ||\vec{\mathbf{k}}||_\rho^{-\alpha H-d-\alpha} \tag{3.16}$$

Therefore

$$\begin{aligned}
I_2 &\leq 2^{1+\alpha}\cdot d^\alpha\cdot D^\alpha\cdot(H+d/\alpha)^\alpha\cdot D^d\cdot\sum_{\vec{\mathbf{k}}\in J}\mu_{\vec{\mathbf{k}}}^{-\alpha H-d-\alpha} \\
&\leq 2^{1+\alpha}\cdot d^\alpha\cdot D^\alpha\cdot(H+d/\alpha)^\alpha\cdot D^d\cdot D^{-\alpha H-d-\alpha}\cdot\sum_{\vec{\mathbf{k}}\in J}||t(\vec{\mathbf{k}})||_\rho^{-\alpha H-d-\alpha} \\
&\leq 2^{1+\alpha}\cdot d^\alpha\cdot(H+d/\alpha)^\alpha\cdot D^{-\alpha H}\cdot 2^d\cdot\sum_{\vec{\mathbf{k}}\in J'}||\vec{\mathbf{k}}||_\rho^{-\alpha H-d-\alpha}
\end{aligned}$$

For each $m\in\{1,\dots,M-1\}$, be

$$J'_m := \{\vec{\mathbf{k}}\in J' : \max_{1\leq j\leq d}(k_j)=m\} = \bigcup_{j=1}^{d}\{\vec{\mathbf{k}}\in\{0,\dots,m\}^d : k_j=m\}.$$

Then

(a) The sets $J'_m, 1 \leq m \leq M-1$, are disjoint and their union is $\bigcup_{m=1}^{M-1} J'_m = J'$.

(b) If $\vec{\mathbf{k}} \in J'_m$ then $||\vec{\mathbf{k}}||_\infty = m$.

(c) The number of elements in the set J'_m can be estimated by

$$\begin{aligned}\#J'_m &\leq \sum_{j=1}^{d} \#\{\vec{\mathbf{k}} \in \{0,\dots,m\}^d : k_j = m\} = d \cdot \#\{\vec{\mathbf{k}} \in \{0,\dots,m\}^d : k_1 = m\} \\ &= d \cdot (m+1)^{d-1}.\end{aligned}$$

Using these properties of J'_m, the sum can be estimated as follows:

$$\begin{aligned}\sum_{\vec{\mathbf{k}} \in J'} ||\vec{\mathbf{k}}||_\rho^{-\alpha H-d-\alpha} &\leq \sum_{\vec{\mathbf{k}} \in J'} ||\vec{\mathbf{k}}||_\infty^{-\alpha H-d-\alpha} \underset{(a)}{=} \sum_{m=1}^{M-1} \sum_{\vec{\mathbf{k}} \in J'_m} ||\vec{\mathbf{k}}||_\infty^{-\alpha H-d-\alpha} \\ &\underset{(b)}{<} \sum_{m=1}^{M} \sum_{\vec{\mathbf{k}} \in J'_m} m^{-\alpha H-d-\alpha} \\ &\underset{(c)}{\leq} \sum_{m=1}^{M} d \cdot (m+1)^{d-1} m^{-\alpha H-d-\alpha} \\ &\leq d \cdot \left(2^{d-1} + \sum_{m=2}^{M} \left(\frac{3}{2}m\right)^{d-1} m^{-\alpha H-d-\alpha}\right) \\ &\leq d \cdot \left(2^{d-1} + \left(\frac{3}{2}\right)^{d-1} \cdot \sum_{m=2}^{M} m^{-\alpha H-\alpha-1}\right) \\ &\leq d \cdot \left(2^{d-1} + \left(\frac{3}{2}\right)^{d-1} \cdot \sum_{m=2}^{M} \int_{m-1}^{m} x^{-\alpha H-\alpha-1} dx\right) \\ &= d \cdot \left(2^{d-1} + \left(\frac{3}{2}\right)^{d-1} \cdot \int_{1}^{M} x^{-\alpha H-\alpha-1} dx\right) \\ &= d \cdot \left(2^{d-1} + \left(\frac{3}{2}\right)^{d-1} \cdot \left[\frac{1}{-\alpha H-\alpha} \cdot x^{-\alpha H-\alpha}\right]_1^M\right) \\ &= d \cdot \left(2^{d-1} + \left(\frac{3}{2}\right)^{d-1} \cdot \frac{1}{\alpha(1+H)} \left(1 - M^{-\alpha(1+H)}\right)\right)\end{aligned}$$

We obtain as an estimation for I_2:

$$\begin{aligned}
I_2 &\leq 2^{\alpha+d+1} \cdot d^{1+\alpha} \cdot (H + d/\alpha)^{\alpha} \cdot D^{-\alpha H} \cdot \left(2^{d-1} + \frac{(3/2)^{d-1}}{\alpha(1+H)} \left(1 - M^{-\alpha(1+H)}\right)\right) \\
&\leq 2^{\alpha+2} \cdot d^{1+\alpha} \cdot (H + d/\alpha)^{\alpha} \cdot D^{-\alpha H} \cdot \left(4^{d-1} + \frac{3^{d-1}}{\alpha(1+H)}\right) \\
&= \widetilde{C}_2 \cdot D^{-\alpha H}
\end{aligned}$$

with $\widetilde{C}_2 := 2^{\alpha+2} \cdot d^{1+\alpha} \cdot (H + d/\alpha)^{\alpha} \cdot \left(4^{d-1} + \frac{3^{d-1}}{\alpha(1+H)}\right)$.

From these estimations of I_0, I_1 and I_2, and the lemmas 3.14 and 3.15, we conclude that

$$\begin{aligned}
||X_{\psi}^{A}(x) - X_{\psi}^{A,M}(x)||_{\alpha}^{\alpha} &\leq \sum_{\vec{\mathbf{k}} \in \{-M,\ldots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}} \right|^{\alpha} d\xi \\
&\leq I_0 + I_1 + I_2 \\
&\leq \widetilde{C}_{0,x} \cdot D^{\alpha(1-H)} + \widetilde{C}_{1,x} \cdot D^{\alpha(1-H)} + \widetilde{C}_2 \cdot D^{-\alpha H}
\end{aligned}$$

with

$$\begin{aligned}
\widetilde{C}_{0,x} &= ||x||_2^{\alpha} \cdot \frac{d^{1+\alpha/2} \cdot 2^d}{\alpha(1-H)} \\
\widetilde{C}_{1,x} &= ||x||_2^{\alpha} \cdot \frac{d^{1+\alpha/2} \cdot 2^{d+1}}{\alpha H} \\
\widetilde{C}_2 &= 2^{\alpha+2} \cdot d^{1+\alpha} \cdot (H + d/\alpha)^{\alpha} \cdot \left(4^{d-1} + \frac{3^{d-1}}{\alpha(1+H)}\right)
\end{aligned}$$

□

Remark 3.17. If expressed using only the parameters A and M (and not D), then this estimation has the following form:

$$||X_{\psi}^{A}(x) - X_{\psi}^{A,M}(x)||_{\alpha}^{\alpha} \leq \widetilde{C}_{0,A,x} \cdot M^{-\alpha(1-H)} + \widetilde{C}_{1,A,x} \cdot M^{-\alpha(1-H)} + \widetilde{C}_{2,A} \cdot M^{\alpha H}$$

with

$$\begin{aligned}
\widetilde{C}_{0,A,x} &= ||x||_2^{\alpha} \cdot \frac{d^{1+\alpha/2} \cdot 2^d}{\alpha(1-H)} \cdot A^{\alpha(1-H)} \\
\widetilde{C}_{1,A,x} &= ||x||_2^{\alpha} \cdot \frac{d^{1+\alpha/2} \cdot 2^{d+1}}{\alpha H} \cdot A^{\alpha(1-H)} \\
\widetilde{C}_{2,A} &= 2^{\alpha+2} \cdot d^{1+\alpha} \cdot (H + d/\alpha)^{\alpha} \cdot \left(4^{d-1} + \frac{3^{d-1}}{\alpha(1+H)}\right) \cdot A^{-\alpha H}
\end{aligned}$$

Remark 3.18. The estimation of I_2 contains the factor $D^{-\alpha H}$ and thus increases if D is chosen smaller (or, equivalently, if M is chosen larger, see Remark 3.17), which is not desired. Obviously, the reason for this undesired asymptotic behaviour is the fact that for smaller values of D, the "hole" near the origin, which isn't considered when estimating I_2, also decreases. Thereby, the domain of integration is enlarged in the area where the integrated function takes its largest values, and thus also the integral increases. This problem can be avoided by setting ψ to zero not on $[-D, D]^d$, but on an environment of the origin whose size is independent of D, e.g. on $[-B, B]^d$ for a fixed parameter $B > 0$. Therefore a corresponding version of the discretisation is also considered in the following.

Second version of the discretisation

In a variation to the previously presented discretisation of the OSSRF $X_\psi(x)$, this OSSRF is again approximated by a sum in the form of (3.8), but now with a different definition of the function $g_{\vec{\mathbf{k}}}$:

$$g_{\vec{\mathbf{k}}} = \begin{cases} 0, & \vec{\mathbf{k}} \in \{-N, \dots, N-1\}^d \\ \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>} - 1\right) \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}, & \text{else} \end{cases} \tag{3.17}$$

(with $N \in \mathbb{N}$). This can be considered as a generalisation of the previous variant of the discretisation, because the first version can be obtained by setting N to the value 1. As an additional parameter, N, is used for the approximation, the discretised random field is now denoted by $X_\psi^{A,M,N}$ (in contrast to the previously used $X_\psi^{A,M}$). Thus, the "truncated" random field

$$X_\psi^A(x) = \operatorname{Re} \int_{[-A,A]^d} \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}}\, W_\alpha(d\xi)$$

is approximated by the sum

$$X_\psi^{A,M,N}(x) = \operatorname{Re} \sum_{\vec{\mathbf{k}} \in J_N} \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>} - 1\right) \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\, W_\alpha(\Delta_{\vec{\mathbf{k}}}) \tag{3.18}$$

with $J_N := \{-M, \dots, M-1\}^d \backslash \{-N, \dots, N-1\}^d$, $\xi_{\vec{\mathbf{k}}} := D \cdot \vec{\mathbf{k}} = (Dk_1, \dots, Dk_d)^T$ and $\Delta_{\vec{\mathbf{k}}} := [k_1 \cdot D, (k_1+1) \cdot D) \times \dots \times [k_d \cdot D, (k_d+1) \cdot D)$ for each $\vec{\mathbf{k}} = (k_1, \dots, k_d)^T \in J_N$. (Note that J_N is defined different from the set J in the first version of the discretisation).

This can be interpreted as setting $\psi(\xi) = 0$ not only on $[-D, D]^d$ (as in the first version), but on $[-B, B]^d$ with $B = N \cdot D = N \cdot \frac{A}{M} = \frac{N}{M} \cdot A$. Thereby, this environment of the origin which is not considered in the approximation, can be kept at an almost constant size if N is chosen approximately proportional to M.

Lemma 3.14 is also valid for this version of the discretisation, as it doesn't use any information on the definition of the function $g_{\vec{\mathbf{k}}}$:

$$||X_\psi^A(x) - X_\psi^{A,M,N}(x)||_\alpha^\alpha \le \sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}} \right|^\alpha d\xi$$

From Lemma 3.15, a similar lemma can be derived for this version of the discretisation:

Lemma 3.19. *With $g_{\vec{\mathbf{k}}}$ being defined according to* (3.17)*, the term on the right side of* (3.10) *in Lemma 3.14 can be estimated further by*

$$\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}} \right|^\alpha d\xi \le I_0 + I_1 + I_2$$

with

$$\begin{aligned} I_0 &= ||x||_2^\alpha \cdot \int_{[-B,B]^d} ||\xi||_2^\alpha \cdot \psi(\xi)^{-\alpha H - q} d\xi \\ I_1 &= 2 \cdot ||x||_2^\alpha \cdot D^\alpha d^{\alpha/2} \cdot \int_{[-A,A]^d \setminus [-B,B]^d} \psi(\xi)^{-\alpha H - q} d\xi \\ I_2 &= 2^{1+\alpha} \cdot \sum_{\vec{\mathbf{k}}\in J_N} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}} \right|^\alpha d\xi \end{aligned}$$

Proof. The proof of this lemma is analogous to the proof of Lemma 3.15. □

The approximation error is estimated again for a ρ-norm as E-homogeneous function ψ (analogous to Theorem 3.16):

Theorem 3.20. *Be ψ the ρ-norm, i.e. the parameters of ψ in the representation* (2.2) *are chosen as in Theorem 3.16. Then the approximation error due to the discretisation is not more than*

$$||X_\psi^A(x) - X_\psi^{A,M,N}(x)||_\alpha^\alpha \le \widetilde{C}_{0,x} \cdot B^{\alpha(1-H)} + \left(\widetilde{C}_{1,A,B,x} + \widetilde{C}_{2,A,B}\right) \cdot D^\alpha$$

with

$$\begin{aligned} \widetilde{C}_{0,x} &= ||x||_2^\alpha \cdot \frac{d^{1+\frac{\alpha}{2}} \cdot 2^d}{\alpha(1-H)} \\ \widetilde{C}_{1,A,B,x} &= ||x||_2^\alpha \cdot \frac{d^{1+\frac{\alpha}{2}} \cdot 2^{d+1}}{\alpha H} \left(B^{-\alpha H} - A^{-\alpha H}\right) \\ \widetilde{C}_{2,A,B} &= 2^{\alpha+2} d^{1+\alpha} \cdot \left(H + \frac{d}{\alpha}\right)^\alpha \cdot \left(4^{d-1} B^{-\alpha H - \alpha} + \frac{3^{d-1}}{\alpha(1+H)} \left(B^{-\alpha H-\alpha} - A^{-\alpha H - \alpha}\right)\right) \end{aligned}$$

Proof. The choice of the parameters implies again $q = \lambda_1 + \ldots + \lambda_d = d$. Following the lemmas 3.14 and 3.19,

$$\begin{aligned} ||X_\psi^A(x) - X_\psi^{A,M,N}(x)||_\alpha^\alpha \leq & \sum_{\vec{\mathbf{k}} \in \{-M,\ldots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}} \right|^\alpha d\xi \\ \leq & I_0 + I_1 + I_2 \end{aligned}$$

with I_0, I_1 and I_2 being defined as in Lemma 3.19.

The results for I_0 and I_1 can be calculated analogously to their estimation in Theorem 3.16.

The term

$$I_2 = 2^{1+\alpha} \cdot \sum_{\vec{\mathbf{k}} \in J_N} \int_{\Delta_{\vec{\mathbf{k}}}} \left| ||\xi||_\rho^{-H-\frac{d}{\alpha}} - ||\xi_{\vec{\mathbf{k}}}||_\rho^{-H-\frac{d}{\alpha}} \right|^\alpha d\xi$$

is estimated as follows:

As already shown in the proof of Theorem 3.16,

$$\left| ||\xi||_\rho^{-H-\frac{d}{\alpha}} - ||\xi_{\vec{\mathbf{k}}}||_\rho^{-H-\frac{d}{\alpha}} \right| \leq d \cdot D \cdot (H + d/\alpha) \cdot \mu_{\vec{\mathbf{k}}}^{-H-\frac{d}{\alpha}-1}$$

with $\mu_{\vec{\mathbf{k}}} := \inf_{\xi \in \Delta_{\vec{\mathbf{k}}}}(||\xi||_\rho)$, and with this inequality we obtain (analogously to the calculation in that proof):

$$\begin{aligned} I_2 &= 2^{1+\alpha} \cdot \sum_{\vec{\mathbf{k}} \in J_N} \int_{\Delta_{\vec{\mathbf{k}}}} \left| ||\xi||_\rho^{-H-\frac{d}{\alpha}} - ||\xi_{\vec{\mathbf{k}}}||_\rho^{-H-\frac{d}{\alpha}} \right|^\alpha d\xi \\ &\leq 2^{1+\alpha} \cdot d^\alpha \cdot D^\alpha \cdot (H + d/\alpha)^\alpha \cdot D^d \cdot \sum_{\vec{\mathbf{k}} \in J_N} \mu_{\vec{\mathbf{k}}}^{-\alpha H - d - \alpha} \end{aligned}$$

With $J'_N := \{\vec{\mathbf{k}} \in J_N : k_j \geq 0,\ 1 \leq j \leq d\} = \{0, \ldots, M-1\}^d \backslash \{0, \ldots, N-1\}^d$ we can write (analogous to (3.15) and (3.16)):

$$\sum_{\vec{\mathbf{k}} \in J_N} \mu_{\vec{\mathbf{k}}}^{-\alpha H - d - \alpha} = D^{-\alpha H - d - \alpha} \cdot 2^d \cdot \sum_{\vec{\mathbf{k}} \in J'_N} ||\vec{\mathbf{k}}||_\rho^{-\alpha H - d - \alpha}$$

Therefore

$$I_2 \leq 2^{1+\alpha} \cdot d^\alpha \cdot (H + d/\alpha)^\alpha \cdot D^{-\alpha H} \cdot 2^d \cdot \sum_{\vec{\mathbf{k}} \in J'_N} ||\vec{\mathbf{k}}||_\rho^{-\alpha H - d - \alpha}$$

The sum can be estimated, analogous to the similar sum in the proof of Theorem 3.16, as follows (with $L_m := \{\vec{\mathbf{k}} \in J'_N : \max_{1\le j\le d}(k_j) = m\}$):

$$\begin{aligned}
\sum_{\vec{\mathbf{k}}\in J'_N} ||\vec{\mathbf{k}}||_\rho^{-\alpha H-d-\alpha} &\le \sum_{\vec{\mathbf{k}}\in J'_N} ||\vec{\mathbf{k}}||_\infty^{-\alpha H-d-\alpha} = \sum_{m=N}^{M-1}\sum_{\vec{\mathbf{k}}\in L_m} ||\vec{\mathbf{k}}||_\infty^{-\alpha H-d-\alpha} \\
&= \sum_{m=N}^{M-1}\sum_{\vec{\mathbf{k}}\in L_m} m^{-\alpha H-d-\alpha} \\
&< \sum_{m=N}^{M} d\cdot(m+1)^{d-1} m^{-\alpha H-d-\alpha} \\
&\le d\cdot\left((N+1)^{d-1}N^{-\alpha H-d-\alpha} + \sum_{m=N+1}^{M}\left(1+\frac{1}{m}\right)^{d-1} m^{-\alpha H-\alpha-1}\right) \\
&\le d\cdot\left(\left(1+\frac{1}{N}\right)^{d-1} N^{-\alpha H-\alpha-1}\right. \\
&\qquad \left. + \left(1+\frac{1}{N+1}\right)^{d-1}\cdot \sum_{m=N+1}^{M} m^{-\alpha H-\alpha-1}\right) \\
&\le d\cdot\left(\left(1+\frac{1}{N}\right)^{d-1} N^{-\alpha H-\alpha-1}\right. \\
&\qquad \left. + \left(1+\frac{1}{N+1}\right)^{d-1}\cdot \int_N^M x^{-\alpha H-\alpha-1}dx\right) \\
&\le d\cdot\left(\left(1+\frac{1}{N}\right)^{d-1} N^{-\alpha H-\alpha-1}\right. \\
&\qquad \left. + \left(1+\frac{1}{N+1}\right)^{d-1}\cdot \left[\frac{1}{-\alpha H-\alpha}\cdot x^{-\alpha H-\alpha}\right]_N^M\right) \\
&\le d\cdot\left(\left(1+\frac{1}{N}\right)^{d-1} N^{-\alpha H-\alpha-1}\right. \\
&\qquad \left. + \left(1+\frac{1}{N+1}\right)^{d-1}\cdot \frac{1}{\alpha(1+H)}\left(N^{-\alpha H-\alpha} - M^{-\alpha H-\alpha}\right)\right)
\end{aligned}$$

We obtain as an estimation for I_2:

$$\begin{aligned}
I_2 \leq & 2^{\alpha+d+1} \cdot d^{1+\alpha} \cdot \left(H + \frac{d}{\alpha}\right)^{\alpha} \cdot D^{-\alpha H} \cdot \left(\left(1 + \frac{1}{N}\right)^{d-1} N^{-\alpha H - \alpha - 1} \right. \\
& \left. + \left(1 + \frac{1}{N+1}\right)^{d-1} \cdot \frac{1}{\alpha(1+H)} \left(N^{-\alpha H-\alpha} - M^{-\alpha H - \alpha}\right) \right) \\
= & 2^{\alpha+d+1} \cdot d^{1+\alpha} \cdot \left(H + \frac{d}{\alpha}\right)^{\alpha} \cdot D^{\alpha} \cdot \left(\left(1 + \frac{D}{B}\right)^{d-1} \cdot \frac{D}{B} \cdot B^{-\alpha H - \alpha} \right. \\
& \left. + \left(1 + \frac{D}{B+D}\right)^{d-1} \cdot \frac{1}{\alpha(1+H)} \left(B^{-\alpha H-\alpha} - A^{-\alpha H - \alpha}\right) \right) \\
\leq & 2^{\alpha+d+1} \cdot d^{1+\alpha} \cdot \left(H + \frac{d}{\alpha}\right)^{\alpha} \cdot D^{\alpha} \\
& \cdot \left(2^{d-1} \cdot B^{-\alpha H - \alpha} + \frac{(3/2)^{d-1}}{\alpha(1+H)} \left(B^{-\alpha H-\alpha} - A^{-\alpha H - \alpha}\right) \right) \\
= & 2^{\alpha+2} \cdot d^{1+\alpha} \cdot \left(H + \frac{d}{\alpha}\right)^{\alpha} \cdot D^{\alpha} \\
& \cdot \left(4^{d-1} \cdot B^{-\alpha H - \alpha} + \frac{3^{d-1}}{\alpha(1+H)} \left(B^{-\alpha H-\alpha} - A^{-\alpha H - \alpha}\right) \right) \\
= & \widetilde{C}_{2,A,B} \cdot D^{\alpha}
\end{aligned}$$

with $\widetilde{C}_{2,A,B} = 2^{\alpha+2} \cdot d^{1+\alpha} \cdot \left(H + \frac{d}{\alpha}\right)^{\alpha} \cdot \left(4^{d-1} \cdot B^{-\alpha H-\alpha} + \frac{3^{d-1}}{\alpha(1+H)} \left(B^{-\alpha H - \alpha} - A^{-\alpha H-\alpha}\right)\right)$. □

Using the results for $||X_\psi(x) - X_\psi^A(x)||_\alpha^\alpha$ and $||X_\psi^A(x) - X_\psi^{A,M,N}(x)||_\alpha^\alpha$, the total error of the approximation $||X_\psi(x) - X_\psi^{A,M,N}(x)||_\alpha^\alpha$ can also be estimated:

Corollary 3.21. *Be ψ a ρ-norm as in Theorem 3.20. Then*

$$\left|\left| X_\psi(x) - X_\psi^{A,M,N}(x) \right|\right|_\alpha^\alpha \leq \widetilde{C} \cdot A^{-\alpha H} + \widetilde{C}_{0,x} \cdot B^{\alpha(1-H)} + \left(\widetilde{C}_{1,A,B,x} + \widetilde{C}_{2,A,B}\right) D^{\alpha}$$

with

$$\begin{aligned}
\widetilde{C} &= \frac{2^{\alpha+d} \cdot d}{\alpha H} \\
\widetilde{C}_{0,x} &= ||x||_2^\alpha \cdot \frac{d^{1+\frac{\alpha}{2}} \cdot 2^d}{\alpha(1-H)} \\
\widetilde{C}_{1,A,B,x} &= ||x||_2^\alpha \cdot \frac{d^{1+\frac{\alpha}{2}} \cdot 2^{d+1}}{\alpha H} \left(B^{-\alpha H} - A^{-\alpha H}\right) \\
\widetilde{C}_{2,A,B} &= 2^{\alpha+2} d^{1+\alpha} \cdot \left(H + \frac{d}{\alpha}\right)^{\alpha} \cdot \left(4^{d-1} B^{-\alpha H - \alpha} + \frac{3^{d-1}}{\alpha(1+H)} \left(B^{-\alpha H-\alpha} - A^{-\alpha H - \alpha}\right)\right)
\end{aligned}$$

Proof. Similar to the proof of Lemma 3.14, the error of estimation can be estimated by

$$\begin{aligned}
&\left|\left|X_\psi(x) - X_\psi^{A,M,N}(x)\right|\right|_\alpha^\alpha \\
&= \left|\left|\mathrm{Re}\left(\int_{\mathbb{R}^d}\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}}\, W_\alpha(d\xi) - \sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} g_{\vec{\mathbf{k}}}\, W_\alpha(\Delta_{\vec{\mathbf{k}}})\right)\right|\right|_\alpha^\alpha \\
&\le \left|\left|\int_{\mathbb{R}^d}\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}}\, W_\alpha(d\xi) - \sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} g_{\vec{\mathbf{k}}}\int_{\mathbb{R}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(\xi)\, W_\alpha(d\xi)\right|\right|_\alpha^\alpha \\
&= \left|\left|\int_{\mathbb{R}^d}\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - \sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} g_{\vec{\mathbf{k}}}\, 1_{\Delta_{\vec{\mathbf{k}}}}(\xi)\, W_\alpha(d\xi)\right|\right|_\alpha^\alpha \\
&= \int_{\mathbb{R}^d}\left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - \sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(\xi)\cdot g_{\vec{\mathbf{k}}}\right|^\alpha d\xi
\end{aligned}$$

If $\xi \in \mathbb{R}^d\backslash[-A,A]^d$, then $1_{\Delta_{\vec{\mathbf{k}}}}(\xi) = 0$ for all $\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d$, which implies $\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(\xi)\cdot g_{\vec{\mathbf{k}}} = 0$.
However, if $\xi \in [-A,A]^d$, then $1 = \sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(\xi)$, because the $\Delta_{\vec{\mathbf{k}}}$ are disjoint and their sum is $[-A,A]^d$. Therefore

$$\begin{aligned}
&\int_{\mathbb{R}^d}\left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - \sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(\xi)\cdot g_{\vec{\mathbf{k}}}\right|^\alpha d\xi \\
&= \int_{\mathbb{R}^d\backslash[-A,A]^d}\left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}}\right|^\alpha d\xi \\
&\qquad + \int_{[-A,A]^d}\left|\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(\xi)\left(\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}}\right)\right|^\alpha d\xi
\end{aligned}$$

Together with $\psi = ||\cdot||_\rho$, this implies

$$\begin{aligned}
&\left|\left|X_\psi(x) - X_\psi^{A,M,N}(x)\right|\right|_\alpha^\alpha \\
\leq &\int_{\mathbb{R}^d \setminus [-A,A]^d} \left|\left(e^{i<x,\xi>} - 1\right) ||\xi||_\rho^{-H-\frac{d}{\alpha}}\right|^\alpha d\xi \\
&\quad + \int_{[-A,A]^d} \left| \sum_{\vec{\mathbf{k}} \in \{-M,\ldots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(\xi) \left(\left(e^{i<x,\xi>} - 1\right) ||\xi||_\rho^{-H-\frac{d}{\alpha}} - g_{\vec{\mathbf{k}}}\right)\right|^\alpha d\xi \\
= &\int_{\mathbb{R}^d \setminus [-A,A]^d} \left|e^{i<x,\xi>} - 1\right|^\alpha ||\xi||_\rho^{-\alpha H - d} d\xi \\
&\quad + \sum_{\vec{\mathbf{k}} \in \{-M,\ldots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\left(e^{i<x,\xi>} - 1\right) ||\xi||_\rho^{-H-\frac{d}{\alpha}} - g_{\vec{\mathbf{k}}}\right|^\alpha d\xi
\end{aligned}$$

According to the proof of 3.5

$$\int_{\mathbb{R}^d \setminus [-A,A]^d} \left|e^{i<x,\xi>} - 1\right|^\alpha ||\xi||_\rho^{-\alpha H - d} d\xi \leq \frac{2^{\alpha+d} \cdot d}{\alpha H} \cdot A^{-\alpha H}$$

and according to Lemma 3.19

$$\sum_{\vec{\mathbf{k}} \in \{-M,\ldots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}}\right|^\alpha d\xi \leq I_0 + I_1 + I_2$$

with I_0, I_1 and I_2 as in Lemma 3.19. With the proof of Theorem 3.20, it follows that

$$\begin{aligned}
\left|\left|X_\psi(x) - X_\psi^{A,M,N}(x)\right|\right|_\alpha^\alpha \leq &\frac{2^{\alpha+d} \cdot d}{\alpha H} \cdot A^{-\alpha H} + I_0 + I_1 + I_2 \\
\leq &\widetilde{C} \cdot A^{-\alpha H} + \widetilde{C}_{0,x} \cdot B^{\alpha(1-H)} + \left(\widetilde{C}_{1,A,B,x} + \widetilde{C}_{2,A,B}\right) D^\alpha
\end{aligned}$$

with

$$\begin{aligned}
\widetilde{C} &= \frac{2^{\alpha+d} \cdot d}{\alpha H} \\
\widetilde{C}_{0,x} &= ||x||_2^\alpha \cdot \frac{d^{1+\frac{\alpha}{2}} \cdot 2^d}{\alpha(1-H)} \\
\widetilde{C}_{1,A,B,x} &= ||x||_2^\alpha \cdot \frac{d^{1+\frac{\alpha}{2}} \cdot 2^{d+1}}{\alpha H} \left(B^{-\alpha H} - A^{-\alpha H}\right) \\
\widetilde{C}_{2,A,B} &= 2^{\alpha+2} d^{1+\alpha} \cdot \left(H + \frac{d}{\alpha}\right)^\alpha \cdot \left(4^{d-1} B^{-\alpha H - \alpha} + \frac{3^{d-1}}{\alpha(1+H)} \left(B^{-\alpha H-\alpha} - A^{-\alpha H - \alpha}\right)\right)
\end{aligned}$$

□

Remark 3.22. In this version of the discretisation, the error estimate is a monotonic increasing function of D, i.e. it can be reduced by choosing a smaller value of the Parameter D (or, equivalently, by choosing larger values of M and N, so that $D = \frac{A}{M} = \frac{B}{N}$ becomes smaller). The estimation for I_0 is independent of D, while the results for I_1 and I_2 contain the factor D^α and therefore converge to 0 when $D \to 0$.

The error estimates of the approximation can be made arbitrarily small by choosing suitable values for the parameters A, B and D:

(a) First, choose a sufficiently large value for A so that the error of truncation is smaller than a given limit $\varepsilon > 0$. (this is possible, because the error estimate converges to 0 if $A \to \infty$).

(b) Then choose a sufficiently small value of B, so that $\widetilde{C}_{0,x} \cdot B^{\alpha(1-H)} < \varepsilon'$ for a given limit $\varepsilon' > 0$ (Because the exponent is negative, we have $\lim_{B\to 0} \widetilde{C}_{0,x} \cdot B^{\alpha(1-H)} = 0$ for every fixed $x \in \mathbb{R}^d$).

(c) Finally choose a sufficiently small value for D, so that $\left(\widetilde{C}_{1,A,B,x} + \widetilde{C}_{2,A,B}\right) \cdot D^\alpha < \varepsilon''$ for a given $\varepsilon'' > 0$ (i.e. choose D so that $0 < D < ((\widetilde{C}_{1,A,B,x} + \widetilde{C}_{2,A,B})^{-1} \cdot \varepsilon'')^{1/\alpha}$).

Thus, the errors are kept smaller than certain given limits (in the latter two steps, appropriate values of B and D should be chosen so that $M = \frac{A}{D} \in \mathbb{N}$ and $N = \frac{B}{D} \in \mathbb{N}$).

Remark 3.23. The algorithms for the simulation of two- or three-dimensional OSSRF in harmonizable representation, which are presented in this thesis, approximate an OSSRF in the points $x = x_{\vec{\mathbf{m}}} = \frac{\pi}{A} \cdot \vec{\mathbf{m}}$ for $\vec{\mathbf{m}} \in \{-M, \dots, M-1\}^d$ (see sections 5.2 and 5.3). For these points, the euclidean norm which appears in the terms I_0' and I_1', is $||x_{\vec{\mathbf{m}}}||_2 = \frac{\pi}{A} \cdot ||\vec{\mathbf{m}}||_2$. If $\vec{\mathbf{m}} \in \{-K, \dots, K-1\}^d$ for some $K \in \{1, \dots, M\}$, then

$$||x_{\vec{\mathbf{m}}}||_2 = \frac{\pi}{A} \cdot ||\vec{\mathbf{m}}||_2 \le \frac{\pi}{A} \cdot \sqrt{d} \cdot ||\vec{\mathbf{m}}||_\infty = \frac{\pi}{A} \cdot \sqrt{d} \cdot K.$$

Therefore

$$\begin{aligned} I_0' &\le A^{-\alpha} \cdot K^\alpha \cdot \frac{\pi^\alpha \cdot d^{1+\alpha} \cdot 2^d}{\alpha(1-H)} \cdot B^{\alpha(1-H)} \\ I_1' &\le A^{-\alpha} \cdot K^\alpha \cdot D^\alpha \cdot \frac{\pi^\alpha \cdot d^{1+\alpha} \cdot 2^{d+1}}{\alpha H} \left(B^{-\alpha H} - A^{-\alpha H}\right) \end{aligned}$$

for $x = x_{\vec{\mathbf{m}}}$ with $\vec{\mathbf{m}} \in \{-K, \dots, K-1\}^d$ (for a $K \in \mathbb{N}, K \le M$).

3.3.2 Approximation error for general ψ

In this subsection, be ψ a function which fulfills the assumptions of Theorem 3.10, i.e. it is a function in the representation (2.2) with linear independent vectors $\theta_1, \ldots, \theta_d \in \mathbb{R}^d$ for which $||\theta_1||_2 = \ldots = ||\theta_d||_2 = 1$, and $0 < \lambda_1 \leq \ldots \leq \lambda_d$. In order to be able to use Corollary 2.6, we also have to assume that $0 < \rho < 2\lambda_1$. Be also assumed that for the parameters A, B and D the inequalities $0 < D \leq B \leq \frac{1}{\sqrt{d}}$ and $1 < A$ hold, and that $N := \frac{B}{D}$ and $M := \frac{A}{D}$ are integers. As already defined in Lemma 3.9, be also in this subsection $C_{min} := \min\{C_1, \ldots, C_d\}$, $C_{max} := \max\{C_1, \ldots, C_d\}$, $S_2 := \{\xi \in \mathbb{R}^d : ||\xi||_2 = 1\}$ and $\theta_\xi := \frac{\xi}{||\xi||_2} \in S_2$. As in Theorem 3.10, be $\tilde{I}$ a short notation for the integral $\int_{S_2} \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{-\frac{\rho}{\lambda_1}} \right)^{\frac{-H\alpha - q}{\rho}} d\theta_\xi$.

Remark 3.24. Obviously the first version of discretisation is a special case of the second one, which is obtained by setting $N = 1$. In this case, the value of B is $B = N \cdot D = D$. Therefore, results for the first version can be derived from the results for the second version by setting $N = 1$ and $B = D$. Thus, in this subsection the results are calculated for the second version of discretisation, and then the corresponding results for the first version are derived by interpreting this discretisation as a special case of the second one.

Second version of the discretisation

Remark 3.25. Be $X_\psi^{A,M,N}$ an approximation of X_ψ^A which uses the second version of discretisation (see (3.18)). According to the lemmas 3.14 and 3.19, the error of approximation can be estimated by

$$\begin{aligned}
&||X_\psi^A(x) - X_\psi^{A,M,N}(x)||_\alpha^\alpha \\
\leq &\sum_{\vec{\mathbf{k}} \in \{-M, \ldots, M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \left(e^{i<x,\xi>} - 1 \right) \psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}} \right|^\alpha d\xi \\
\leq &I_0 + I_1 + I_2
\end{aligned}$$

with

$$\begin{aligned}
I_0 &= ||x||_2^\alpha \cdot \int_{[-B,B]^d} ||\xi||_2^\alpha \cdot \psi(\xi)^{-\alpha H - q} \, d\xi \\
I_1 &= 2 \cdot ||x||_2^\alpha \cdot D^\alpha d^{\frac{\alpha}{2}} \cdot \int_{[-A,A]^d \setminus [-B,B]^d} \psi(\xi)^{-\alpha H - q} \, d\xi \\
I_2 &= 2^{1+\alpha} \cdot \sum_{\vec{\mathbf{k}} \in J_N} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}} \right|^\alpha d\xi
\end{aligned}$$

Lemma 3.26. *Under the given assumptions, the term I_0 can be estimated by*

$$I_0 \leq \widetilde{C}_{0,x} \cdot B^{\frac{\alpha\lambda_1 - \alpha H - q + d\lambda_1}{\lambda_1}}$$

with

$$\widetilde{C}_{0,x} = ||x||_2^\alpha \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \cdot \tilde{I} \cdot \frac{\lambda_1}{\alpha\lambda_1 - \alpha H - q + d\lambda_1} \cdot d^{\frac{\alpha\lambda_1 - \alpha H - q + d\lambda_1}{2\lambda_1}}$$

and $\tilde{I} = \int_{S_2} \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\frac{\rho}{\lambda_1}} \right)^{\frac{-\alpha H - q}{\rho}} d\theta_\xi$,
if the exponent is positive, i.e. if $\alpha(\lambda_1 - H) > q - d\lambda_1$.

Proof. From the assumption $B \leq \frac{1}{\sqrt{d}}$ follows $||\xi||_2 \leq 1$ for all $\xi \in [-B, B]^d$. According to Lemma 3.9,

$$\psi(\xi) \geq C_{min}^{1/\rho} \cdot ||\xi||_2^{1/\lambda_1} \cdot \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\rho/\lambda_1} \right)^{1/\rho}$$

holds for these ξ. Using this inequality, we estimate

$$\begin{aligned}
I_0 &= ||x||_2^\alpha \cdot \int_{[-B,B]^d} ||\xi||_2^\alpha \cdot \psi(\xi)^{-\alpha H - q} \, d\xi \\
&\leq ||x||_2^\alpha \cdot \int_{[-B,B]^d} ||\xi||_2^\alpha \cdot \left(C_{min}^{\frac{1}{\rho}} \cdot ||\xi||_2^{\frac{1}{\lambda_1}} \cdot \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\frac{\rho}{\lambda_1}} \right)^{\frac{1}{\rho}} \right)^{-\alpha H - q} d\xi \\
&= ||x||_2^\alpha \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \cdot \int_{[-B,B]^d} ||\xi||_2^\alpha \cdot ||\xi||_2^{\frac{-\alpha H - q}{\lambda_1}} \cdot \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\frac{\rho}{\lambda_1}} \right)^{\frac{-\alpha H - q}{\rho}} d\xi \\
&\leq ||x||_2^\alpha \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \cdot \int_0^{\sqrt{d} \cdot B} \int_{S_2} r^\alpha \cdot r^{\frac{-\alpha H - q}{\lambda_1}} \cdot \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\frac{\rho}{\lambda_1}} \right)^{\frac{-\alpha H - q}{\rho}} d\theta_\xi \cdot r^{d-1} \, dr \\
&= ||x||_2^\alpha \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \cdot \int_0^{\sqrt{d} \cdot B} r^{\frac{-\alpha H - q + \alpha\lambda_1}{\lambda_1} + d - 1} \cdot \underbrace{\int_{S_2} \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\frac{\rho}{\lambda_1}} \right)^{\frac{-\alpha H - q}{\rho}} d\theta_\xi}_{:=\tilde{I}} \, dr \\
&= ||x||_2^\alpha \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \cdot \tilde{I} \cdot \int_0^{\sqrt{d} \cdot B} r^{\frac{-\alpha H - q + \alpha\lambda_1}{\lambda_1} + d - 1} \, dr \\
&= ||x||_2^\alpha \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \cdot \tilde{I} \cdot \frac{\lambda_1}{\alpha\lambda_1 - \alpha H - q + d\lambda_1} \left(\sqrt{d} \cdot B \right)^{\frac{\alpha\lambda_1 - \alpha H - q + d\lambda_1}{\lambda_1}} \\
&= \widetilde{C}_{0,x} \cdot B^{\frac{\alpha\lambda_1 - \alpha H - q + d\lambda_1}{\lambda_1}}
\end{aligned}$$

with $\widetilde{C}_{0,x} = ||x||_2^\alpha \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \cdot \tilde{I} \cdot \frac{\lambda_1}{\alpha\lambda_1 - \alpha H - q + d\lambda_1} \cdot d^{\frac{\alpha\lambda_1 - \alpha H - q + d\lambda_1}{2\lambda_1}}$ □

Lemma 3.27. *Under the given assumptions, and the condition that $\alpha H + q - d\lambda_d \neq 0$, the term I_1 can be estimated by*

$$I_1 \leq \widetilde{C}_{1,A,B,x} \cdot D^\alpha$$

with

$$\begin{aligned}\widetilde{C}_{1,A,B,x} =& 2 \cdot ||x||_2^\alpha \cdot d^{\frac{\alpha}{2}} \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \cdot \tilde{I} \cdot \left(\frac{\lambda_1}{\alpha H + q - d\lambda_1} \left(B^{\frac{-\alpha H - q + d\lambda_1}{\lambda_1}} - 1 \right) \right. \\ & \left. + \frac{\lambda_d}{\alpha H + q - d\lambda_d} \left(1 - \left(A\sqrt{d} \right)^{\frac{-\alpha H - q + d\lambda_d}{\lambda_d}} \right) \right)\end{aligned}$$

(and $\tilde{I}$ as before).

Proof. Be $S := \{x \in \mathbb{R}^d : ||x||_2 \leq 1\}$. From the assumption $B \leq \frac{1}{\sqrt{d}}$ follows that $[-B,B]^d \subset S$, and the assumption $1 < A$ implies $S \subset [-A,A]^d$. The domain of integration $[-A,A]^d \backslash [-B,B]^d$ is now divided into the two parts $[-A,A]^d \backslash S$ and $S \backslash [-B,B]^d$. According to Lemma 3.9, $\psi(\xi) \geq C_{min}^{1/\rho} \cdot ||\xi||_2^{1/\lambda_d} \cdot \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\rho/\lambda_1} \right)^{1/\rho}$ for ξ from the first part, and $\psi(\xi) \geq C_{min}^{1/\rho} \cdot ||\xi||_2^{1/\lambda_1} \cdot \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\rho/\lambda_1} \right)^{1/\rho}$ for ξ from the latter part. These inequalities are applied in the following calculation:

$$\begin{aligned} I_1 &= 2 \cdot ||x||_2^\alpha \cdot D^\alpha \cdot d^{\frac{\alpha}{2}} \cdot \int_{[-A,A]^d \backslash [-B,B]^d} \psi(\xi)^{-\alpha H - q} d\xi \\ &= 2 \cdot ||x||_2^\alpha \cdot D^\alpha \cdot d^{\frac{\alpha}{2}} \cdot \left(\int_{[-A,A]^d \backslash S} \psi(\xi)^{-\alpha H - q} d\xi + \int_{S \backslash [-B,B]^d} \psi(\xi)^{-\alpha H - q} d\xi \right) \\ &\leq 2 \cdot ||x||_2^\alpha \cdot D^\alpha \cdot d^{\frac{\alpha}{2}} \cdot \left(\int_{[-A,A]^d \backslash S} \left(C_{min}^{\frac{1}{\rho}} \cdot ||\xi||_2^{\frac{1}{\lambda_d}} \cdot \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\frac{\rho}{\lambda_1}} \right)^{\frac{1}{\rho}} \right)^{-\alpha H - q} d\xi \right. \\ &\qquad \left. + \int_{S \backslash [-B,B]^d} \left(C_{min}^{\frac{1}{\rho}} \cdot ||\xi||_2^{\frac{1}{\lambda_1}} \cdot \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\frac{\rho}{\lambda_1}} \right)^{\frac{1}{\rho}} \right)^{-\alpha H - q} d\xi \right) \\ &\leq 2 \cdot ||x||_2^\alpha \cdot D^\alpha \cdot d^{\frac{\alpha}{2}} \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \\ &\qquad \cdot \left(\int_{[-A,A]^d \backslash S} ||\xi||_2^{\frac{-\alpha H - q}{\lambda_d}} \cdot \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\frac{\rho}{\lambda_1}} \right)^{\frac{-\alpha H - q}{\rho}} d\xi \right. \\ &\qquad \left. + \int_{S \backslash [-B,B]^d} ||\xi||_2^{\frac{-\alpha H - q}{\lambda_1}} \cdot \left(\sum_{k=1}^d |< \theta_\xi, \theta_k >|^{\frac{\rho}{\lambda_1}} \right)^{\frac{-\alpha H - q}{\rho}} d\xi \right) \end{aligned}$$

$$\begin{aligned}
&\leq 2 \cdot ||x||_2^{\alpha} \cdot D^{\alpha} \cdot d^{\frac{\alpha}{2}} \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \\
&\quad \cdot \left(\int_1^{A\sqrt{d}} r^{\frac{-\alpha H - q}{\lambda_d}} \cdot r^{d-1} \cdot \int_{S_2} \left(\sum_{k=1}^{d} |< \theta_\xi, \theta_k >|^{\frac{\rho}{\lambda_1}} \right)^{\frac{-\alpha H - q}{\rho}} d\theta_\xi \, dr \right. \\
&\qquad \left. + \int_B^1 r^{\frac{-\alpha H - q}{\lambda_1}} \cdot r^{d-1} \cdot \int_{S_2} \left(\sum_{k=1}^{d} |< \theta_\xi, \theta_k >|^{\frac{\rho}{\lambda_1}} \right)^{\frac{-\alpha H - q}{\rho}} d\theta_\xi \, dr \right) \\
&\leq 2 \cdot ||x||_2^{\alpha} \cdot D^{\alpha} \cdot d^{\frac{\alpha}{2}} \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \cdot \left(\int_1^{A\sqrt{d}} r^{\frac{-\alpha H - q}{\lambda_d} + d - 1} \cdot \tilde{I} \, dr + \int_B^1 r^{\frac{-\alpha H - q}{\lambda_1} + d - 1} \cdot \tilde{I} \, dr \right) \\
&= 2 \cdot ||x||_2^{\alpha} \cdot D^{\alpha} \cdot d^{\frac{\alpha}{2}} \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \cdot \tilde{I} \cdot \left(\int_1^{A\sqrt{d}} r^{\frac{-\alpha H - q}{\lambda_d} + d - 1} \, dr + \int_B^1 r^{\frac{-\alpha H - q}{\lambda_1} + d - 1} \, dr \right).
\end{aligned}$$

Because $d\lambda_1 < \lambda_1 + \ldots + \lambda_d = q$, for the exponent of the second integral the inequality $\frac{-\alpha H - q + d\lambda_1}{\lambda_1} - 1 < -1$ holds, therefore the integral is

$$\int_B^1 r^{\frac{-\alpha H - q}{\lambda_1} + d - 1} \, dr = \frac{\lambda_1}{-\alpha H - q + d\lambda_d} \cdot \left(1 - B^{\frac{-\alpha H - q + d\lambda_1}{\lambda_1}} \right),$$

and under the assumption $\alpha H + q - d\lambda_d \neq 0$, the exponent of the first integral is $\frac{-\alpha H - q + d\lambda_d}{\lambda_d} - 1 \neq -1$, so that this integral can be calculated analogously:

$$\int_1^{A\sqrt{d}} r^{\frac{-\alpha H - q + d\lambda_d}{\lambda_d} - 1} \, dr = \frac{\lambda_d}{-\alpha H - q + d\lambda_d} \cdot \left(\left(A\sqrt{d} \right)^{\frac{-\alpha H - q + d\lambda_d}{\lambda_d}} - 1 \right).$$

Therefore

$$\begin{aligned}
I_1 \leq{}& 2 \cdot ||x||_2^{\alpha} \cdot D^{\alpha} \cdot d^{\frac{\alpha}{2}} \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \cdot \tilde{I} \cdot \left(\frac{\lambda_1}{\alpha H + q - d\lambda_1} \left(B^{\frac{-\alpha H - q + d\lambda_1}{\lambda_1}} - 1 \right) \right. \\
&\left. + \frac{\lambda_d}{\alpha H + q - d\lambda_d} \left(1 - \left(A\sqrt{d} \right)^{\frac{-\alpha H - q + d\lambda_d}{\lambda_d}} \right) \right) \\
={}& \widetilde{C}_{1,A,B,x} \cdot D^{\alpha}
\end{aligned}$$

with

$$\begin{aligned}
\widetilde{C}_{1,A,B,x} ={}& 2 \cdot ||x||_2^{\alpha} \cdot d^{\frac{\alpha}{2}} \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \cdot \tilde{I} \cdot \left(\frac{\lambda_1}{\alpha H + q - d\lambda_1} \left(B^{\frac{-\alpha H - q + d\lambda_1}{\lambda_1}} - 1 \right) \right. \\
&\left. + \frac{\lambda_d}{\alpha H + q - d\lambda_d} \left(1 - \left(A\sqrt{d} \right)^{\frac{-\alpha H - q + d\lambda_d}{\lambda_d}} \right) \right).
\end{aligned}$$

□

Remark 3.28. If $\lambda_1 = \ldots = \lambda_d$, then $d\lambda_1 = d\lambda_d = q$ and

$$I_1 \leq \widetilde{C}_{1,A,B,x} \cdot D^\alpha$$

with

$$\widetilde{C}_{1,A,B,x} = 2 \cdot ||x||_2^\alpha \cdot d^{\frac{\alpha}{2}} \cdot C_{min}^{\frac{-\alpha H - d\lambda_1}{\rho}} \cdot \tilde{I} \cdot \frac{\lambda_1}{\alpha H} \cdot \left(B^{\frac{-\alpha H}{\lambda_1}} - \left(A \cdot \sqrt{d} \right)^{\frac{-\alpha H}{\lambda_1}} \right).$$

Before estimating I_2, some lemmas are shown first which are used to estimate I_2. In the following, the variables $J_{N,1}$ and $J_{N,2}$ refer to two subsets of the set $J_N = \{-M, \ldots, M-1\}^d \backslash \{-N, \ldots, N-1\}^d$:

$$\begin{aligned} J_{N,1} &:= \{\vec{\mathbf{k}} \in J_N : \Delta_{\vec{\mathbf{k}}} \cap [-1,1)^d \neq \emptyset\}, \\ J_{N,2} &:= \{\vec{\mathbf{k}} \in J_N : \Delta_{\vec{\mathbf{k}}} \cap [-1,1)^d = \emptyset\}. \end{aligned}$$

Then $J_{N,1}, J_{N,2}$ are disjoint, $J_{N,1} \cup J_{N,2} = J_N$ and

$$\begin{aligned} &\vec{\mathbf{k}} \in J_{N,1} \Rightarrow \exists\, \xi \in \Delta_{\vec{\mathbf{k}}} : ||\xi||_\infty \leq 1 \\ &\vec{\mathbf{k}} \in J_{N,1} \Rightarrow \forall\, \xi \in \Delta_{\vec{\mathbf{k}}} : ||\xi||_\infty \leq 1 + D \\ &\vec{\mathbf{k}} \in J_{N,2} \Rightarrow \forall\, \xi \in \Delta_{\vec{\mathbf{k}}} : ||\xi||_\infty \geq 1 \end{aligned}$$

Lemma 3.29. *Be*

$$c_1 := C_{min}^{\frac{1}{\rho}} \cdot \min_{\theta_\xi \in S_2} \left(\sum_{k=1}^{d} |< \theta_\xi, \theta_k >|^{\frac{\rho}{\lambda_1}} \right)^{\frac{1}{\rho}}.$$

Then

(a) $c_1 > 0$.

(b) If $||\xi||_2 \geq 1$ *then* $\psi(\xi) \geq c_1 \cdot ||\xi||_\infty^{\frac{1}{\lambda_d}}$.

(c) If $||\xi||_2 \leq 1$ *then* $\psi(\xi) \geq c_1 \cdot ||\xi||_\infty^{\frac{1}{\lambda_1}}$.

(d) $\psi(\xi) \geq c_1 \cdot B^{\frac{1}{\lambda_1}}$ *for all* $\xi \in [-A, A]^d \backslash [-B, B]^d$.

Proof.

(a) The function $\tilde{\psi}(x) = \left(\sum_{k=1}^{d} |< \theta_\xi, \theta_k >|^{\frac{\rho}{\lambda_1}}\right)^{\frac{1}{\rho}}$ is a continuous E-homogeneous function (according to Corollary 2.6), and this implies $\min_{\theta_\xi \in S_2}\left(\tilde{\psi}(\xi)\right) > 0$ (according to Remark 2.4). Because $C_1, \dots, C_d$ are assumed to be positive, $C_{min} := \min\{C_1, \dots, C_d\}$ is also positive. Thus, $c_1 > 0$.

(b) If $||\xi||_2 \geq 1$, then $\psi(\xi) \geq C_{min}^{1/\rho} \cdot ||\xi||_2^{1/\lambda_d} \cdot \left(\sum_{k=1}^{d} |< \theta_\xi, \theta_k >|^{\rho/\lambda_1}\right)^{1/\rho}$ (see Lemma 3.9). This implies $\psi(\xi) \geq c_1 \cdot ||\xi||_2^{1/\lambda_d} \geq c_1 \cdot ||\xi||_\infty^{1/\lambda_d}$.

(c) If $||\xi||_2 \leq 1$, then $\psi(\xi) \geq C_{min}^{1/\rho} \cdot ||\xi||_2^{1/\lambda_1} \cdot \left(\sum_{k=1}^{d} |< \theta_\xi, \theta_k >|^{\rho/\lambda_1}\right)^{1/\rho}$ (see Lemma 3.9). This implies $\psi(\xi) \geq c_1 \cdot ||\xi||_2^{1/\lambda_1} \geq c_1 \cdot ||\xi||_\infty^{1/\lambda_1}$.

(d) Be $\xi \in [-A, A]^d \backslash [-B, B]^d$. Then $||\xi||_\infty \geq B$.
If $||\xi||_2 \leq 1$, then $\psi(\xi) \geq c_1 \cdot ||\xi||_\infty^{1/\lambda_1} \geq c_1 \cdot B^{1/\lambda_1}$,
and if $||\xi||_2 > 1$, then $\psi(\xi) \geq c_1 \cdot ||\xi||_2^{1/\lambda_d} \geq c_1 \cdot 1 \geq c_1 \cdot B^{1/\lambda_1}$.

□

Lemma 3.30. *There is a $c_4 > 0$ and a $\delta > 0$ (depending on ψ) such that for all $\vec{\mathbf{k}} \in J_N$, for all $\xi \in \Delta_{\vec{\mathbf{k}}}$*

$$|\psi(\xi_{\vec{\mathbf{k}}}) - \psi(\xi)| \leq c_4 D^{(1/\lambda_d - \delta)\beta}$$

Proof. According to Corollary 2.6, ψ is (β, E)-admissible for a matrix E which fulfills the equation $E^T\theta_j = \lambda_j\theta_j$ for all $1 \leq j \leq d$ (in other words: a matrix E for which the transpose E^T has the eigenvectors θ_j and the corresponding eigenvalues λ_j) and for a real number β with $0 < \beta < \min\left(\lambda_1, \rho\frac{\lambda_1}{\lambda_d}\right)$ if $\lambda_1 \leq \rho$ and $\beta = \rho$ if $\lambda_1 > \rho$. With Definition 2.5, this implies that for any $0 < \tilde{A} < \tilde{B}$ there exists a $\tilde{C} > 0$ so that for all $x, y \in \mathbb{R}^d$ with $A \leq ||y|| \leq B$ and $\tau(x) \leq 1$, the inequality $|\psi(x + y) - \psi(y)| \leq C\tau(x)^\beta$ holds. As $\xi_{\vec{\mathbf{k}}}, \xi \in \Delta_{\vec{\mathbf{k}}} \Rightarrow ||\xi_{\vec{\mathbf{k}}} - \xi||_\infty \leq D$ and as we are interested in results for small $D > 0$, it may be assumed that $\xi_{\vec{\mathbf{k}}}$ and ξ are sufficiently close to each other, and especially that the inequality $\tau(\xi_{\vec{\mathbf{k}}} - \xi) < 1$ holds. With this assumption, it follows that

$$|\psi(\xi_{\vec{\mathbf{k}}}) - \psi(\xi)| = |\psi(\xi + (\xi_{\vec{\mathbf{k}}} - \xi)) - \psi(\xi)| \leq \tilde{C}\tau(\xi_{\vec{\mathbf{k}}} - \xi)^\beta.$$

From Lemma 2.1 it follows that there are a norm $|| \cdot ||_0$ and constants $c_2, \delta > 0$ so that $\tau(\xi_{\vec{\mathbf{k}}} - \xi) \leq c_2||\xi_{\vec{\mathbf{k}}} - \xi||_0^{\frac{1}{\lambda_d} - \delta}$. Because all norms are equivalent in $\mathbb{R}^d$, there exists a constant $c_3 > 0$ for which $||x||_0 \le c_3 ||x||_\infty$ for all $x \in \mathbb{R}^d$. Therefore,

$$\begin{aligned}|\psi(\xi_{\vec{\mathbf{k}}}) - \psi(\xi)| &\le \tilde{C}\tau(\xi_{\vec{\mathbf{k}}} - \xi)^\beta \le \tilde{C}\left(c_2 ||\xi_{\vec{\mathbf{k}}} - \xi||_0^{\frac{1}{\lambda_d}-\delta}\right)^\beta \\ &\le \tilde{C}\left(c_2\left(c_3||\xi_{\vec{\mathbf{k}}} - \xi||_\infty\right)^{\frac{1}{\lambda_d}-\delta}\right)^\beta = \tilde{C}c_2^\beta c_3^{(1/\lambda_d-\delta)\beta}||\xi_{\vec{\mathbf{k}}} - \xi||_\infty^{(1/\lambda_d-\delta)\beta} \\ &= c_4||\xi_{\vec{\mathbf{k}}} - \xi||_\infty^{(1/\lambda_d-\delta)\beta} \\ &\le c_4 D^{(1/\lambda_d-\delta)\beta}\end{aligned}$$

with $c_4 := \tilde{C}c_2^\beta c_3^{(1/\lambda_d-\delta)\beta}$. □

In the following two lemmas, this lemma is used to find estimations for the expression $|\psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}|$ in the two cases $\vec{\mathbf{k}} \in J_{N,1}$ and $\vec{\mathbf{k}} \in J_{N,2}$:

Lemma 3.31. *Be $\vec{\mathbf{k}} \in J_{N,1}$. Then constants $c_5, \beta, \delta > 0$ (independent of D, but dependent from ψ, α and H) exist, such that*

$$\left|\psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right| \le c_5 \cdot D^{(1/\lambda_d-\delta)\beta} \cdot B^{\frac{1}{\lambda_1}(-H-\frac{q}{\alpha}-1)}.$$

Proof. Be $\vec{\mathbf{k}} \in J_{N,1}$. According to Lemma 3.29 (d), $\psi(\xi) \ge c_1 \cdot B^{\frac{1}{\lambda_1}}$ for all $\xi \in [-A,A]^d \backslash [-B,B]^d$, and according to Lemma 3.30, there are $c_4', \delta > 0$ with $|\psi(\xi_{\vec{\mathbf{k}}}) - \psi(\xi)| \le c_4' D^{(1/\lambda_d-\delta)\beta}$. From this lemmas and the Mean Value Theorem follows that

$$\begin{aligned}&\left|\psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right| \\ \le&\left|\psi(\xi) - \psi(\xi_{\vec{\mathbf{k}}})\right| \cdot \left|-H - \frac{q}{\alpha}\right| \cdot \left(c_1 \cdot B^{\frac{1}{\lambda_1}}\right)^{-H-\frac{q}{\alpha}-1} \\ \le& c_4' D^{(1/\lambda_d-\delta)\beta} \cdot \left(H + \frac{q}{\alpha}\right) \cdot c_1^{-H-\frac{q}{\alpha}-1} \cdot B^{\frac{1}{\lambda_1}(-H-\frac{q}{\alpha}-1)} \\ =& c_5 \cdot D^{(1/\lambda_d-\delta)\beta} \cdot B^{\frac{1}{\lambda_1}(-H-\frac{q}{\alpha}-1)}\end{aligned}$$

with $c_5 := c_4' \cdot \left(H + \frac{q}{\alpha}\right) \cdot c_1^{-H-\frac{q}{\alpha}-1}$. □

Lemma 3.32. *Be $\vec{\mathbf{k}} \in J_{N,2}$. Then constants $c_6, \beta, \delta > 0$ (independent of D, but dependent from ψ, α and H) exists, such that*

$$\left|\psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right| \le c_6 \cdot D^{\left(\frac{1}{\lambda_d}-\delta\right)\beta + \frac{1}{\lambda_d}\left(-H-\frac{q}{\alpha}-1\right)} \cdot ||t(\vec{\mathbf{k}})||_\infty^{\frac{1}{\lambda_d}\left(-H-\frac{q}{\alpha}-1\right)}$$

(where $t(\vec{\mathbf{k}})$ is defined as in the proof of Theorem 3.16).

Proof. Be $\vec{\mathbf{k}} \in J_{N,2}$. Then $||\xi||_\infty \geq 1 \Rightarrow ||\xi||_2 \geq 1$, and with Lemma 3.29 follows $\psi(\xi) \geq c_1 \cdot ||\xi||_\infty^{\frac{1}{\lambda_d}}$ (for all $\xi \in \Delta_{\vec{\mathbf{k}}}$). The existence of $c_4'', \delta > 0$ with $|\psi(\xi_{\vec{\mathbf{k}}}) - \psi(\xi)| \leq c_4'' \cdot D^{(1/\lambda_d - \delta)\beta}$ (according to Lemma 3.30) is also used:

$$\begin{aligned}
&|\psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}| \\
\leq &|\psi(\xi) - \psi(\xi_{\vec{\mathbf{k}}})| \cdot \left|-H - \frac{q}{\alpha}\right| \cdot \max\left\{\psi(\xi)^{-H-\frac{q}{\alpha}-1}, \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}-1}\right\} \\
\leq &c_4'' \cdot D^{\left(\frac{1}{\lambda_d}-\delta\right)\beta} \cdot \left(H + \frac{q}{\alpha}\right) \cdot \max\left\{\left(c_1 \cdot ||\xi||_\infty^{\frac{1}{\lambda_d}}\right)^{-H-\frac{q}{\alpha}-1}, \left(c_1 \cdot ||\xi_{\vec{\mathbf{k}}}||_\infty^{\frac{1}{\lambda_d}}\right)^{-H-\frac{q}{\alpha}-1}\right\} \\
= &c_4'' \cdot D^{\left(\frac{1}{\lambda_d}-\delta\right)\beta} \cdot \left(H + \frac{q}{\alpha}\right) \cdot c_1^{-H-\frac{q}{\alpha}-1} \cdot \left(\min\left\{||\xi||_\infty, ||\xi_{\vec{\mathbf{k}}}||_\infty\right\}\right)^{\frac{1}{\lambda_d}\left(-H-\frac{q}{\alpha}-1\right)} \\
\leq &c_4'' \cdot \left(H + \frac{q}{\alpha}\right) \cdot c_1^{-H-\frac{q}{\alpha}-1} \cdot D^{\left(\frac{1}{\lambda_d}-\delta\right)\beta} \cdot ||\tilde{\xi}_{\vec{\mathbf{k}}}||_\infty^{\frac{1}{\lambda_d}\left(-H-\frac{q}{\alpha}-1\right)} \\
\leq &c_4'' \cdot \left(H + \frac{q}{\alpha}\right) \cdot c_1^{-H-\frac{q}{\alpha}-1} \cdot D^{\left(\frac{1}{\lambda_d}-\delta\right)\beta} \cdot D^{\frac{1}{\lambda_d}\left(-H-\frac{q}{\alpha}-1\right)} \cdot ||t(\vec{\mathbf{k}})||_\infty^{\frac{1}{\lambda_d}\left(-H-\frac{q}{\alpha}-1\right)} \\
= &c_6 \cdot D^{\left(\frac{1}{\lambda_d}-\delta\right)\beta + \frac{1}{\lambda_d}\left(-H-\frac{q}{\alpha}-1\right)} \cdot ||t(\vec{\mathbf{k}})||_\infty^{\frac{1}{\lambda_d}\left(-H-\frac{q}{\alpha}-1\right)}
\end{aligned}$$

with $c_6 := c_4'' \cdot \left(H + \frac{q}{\alpha}\right) \cdot c_1^{-H-\frac{q}{\alpha}-1}$, and with $\tilde{\xi}_{\vec{\mathbf{k}}}$ and $t(\vec{\mathbf{k}})$ being defined as in (3.13) and (3.14) in the proof of Theorem 3.16. □

The preceding lemmas are now used to estimate I_2:

Lemma 3.33. *Under the assumptions stated at the beginning of this subsection, and the conditions that $-\alpha H - \alpha - q + d\lambda_d \neq 0$ and $\alpha H + \alpha + q - d\lambda_d + \lambda_d \geq 0$, there exist constants (i.e. values which are independent of D) $\widetilde{C}_{2,A,B}, \delta' > 0$ such that*

$$I_2 < \widetilde{C}_{2,A,B} \cdot D^{\frac{\alpha\beta}{\lambda_d}-\delta'} \cdot (1+D)^d.$$

Proof. According to Remark 3.25 and Lemma 3.19,

$$I_2 = 2^{1+\alpha} \cdot \sum_{\vec{\mathbf{k}} \in J_N} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right|^\alpha d\xi.$$

Using the facts that $J_{N,1}, J_{N,2}$ are disjoint and $J_{N,1} \cup J_{N,2} = J_N$, and with the lemmas

3.31 and 3.32, this term can be transformed as follows:

$$\begin{aligned}
I_2 &= 2^{1+\alpha} \cdot \sum_{\vec{\mathbf{k}} \in J_N} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}} \right|^{\alpha} d\xi \\
&= 2^{1+\alpha} \cdot \left(\sum_{\vec{\mathbf{k}} \in J_{N,1}} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}} \right|^{\alpha} d\xi \right. \\
&\qquad \left. + \sum_{\vec{\mathbf{k}} \in J_{N,2}} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}} \right|^{\alpha} d\xi \right) \\
&\le 2^{1+\alpha} \cdot \left(\sum_{\vec{\mathbf{k}} \in J_{N,1}} \int_{\Delta_{\vec{\mathbf{k}}}} \left(c_5 \cdot D^{\left(\frac{1}{\lambda_d}-\delta\right)\beta} \cdot B^{\frac{1}{\lambda_1}(-H-\frac{q}{\alpha}-1)} \right)^{\alpha} d\xi \right. \\
&\qquad \left. + \sum_{\vec{\mathbf{k}} \in J_{N,2}} \int_{\Delta_{\vec{\mathbf{k}}}} \left(c_6 \cdot D^{\left(\frac{1}{\lambda_d}-\delta\right)\beta + \frac{1}{\lambda_d}\left(-H-\frac{q}{\alpha}-1\right)} \cdot ||t(\vec{\mathbf{k}})||_{\infty}^{\frac{1}{\lambda_d}\left(-H-\frac{q}{\alpha}-1\right)} \right)^{\alpha} d\xi \right) \\
&\le 2^{1+\alpha} \cdot \left(c_5^{\alpha} \cdot D^{\left(\frac{1}{\lambda_d}-\delta\right)\alpha\beta} \cdot B^{\frac{1}{\lambda_1}(-H-\frac{q}{\alpha}-1)\alpha} \cdot \sum_{\vec{\mathbf{k}} \in J_{N,1}} \int_{\Delta_{\vec{\mathbf{k}}}} 1 d\xi \right. \\
&\qquad \left. + c_6^{\alpha} \cdot D^{\left(\frac{1}{\lambda_d}-\delta\right)\alpha\beta + \frac{1}{\lambda_d}\left(-H-\frac{q}{\alpha}-1\right)\alpha} \cdot \sum_{\vec{\mathbf{k}} \in J_{N,2}} ||t(\vec{\mathbf{k}})||_{\infty}^{\frac{1}{\lambda_d}\left(-H-\frac{q}{\alpha}-1\right)\alpha} \int_{\Delta_{\vec{\mathbf{k}}}} 1\ d\xi \right) \\
&\le 2^{1+\alpha} \cdot c_5^{\alpha} \cdot D^{\left(\frac{1}{\lambda_d}-\delta\right)\alpha\beta} \cdot B^{\frac{-\alpha H - q - \alpha}{\lambda_1}} \cdot \int_{[-1-D,1+D]^d} 1\ d\xi \\
&\quad + 2^{1+\alpha} \cdot c_6^{\alpha} \cdot D^{\left(\frac{1}{\lambda_d}-\delta\right)\alpha\beta + \frac{-\alpha H - q - \alpha}{\lambda_d}} \cdot D^d \cdot \sum_{\vec{\mathbf{k}} \in J_{N,2}} ||t(\vec{\mathbf{k}})||_{\infty}^{\frac{1}{\lambda_d}(-\alpha H - q - \alpha)}
\end{aligned}$$

Analogous to equation (3.16), the sum can be transformed to

$$\sum_{\vec{\mathbf{k}} \in J_{N,2}} ||t(\vec{\mathbf{k}})||_{\infty}^{\frac{1}{\lambda_d}(-\alpha H - q - \alpha)} = 2^d \cdot \sum_{\vec{\mathbf{k}} \in J'_{N,2}} ||t(\vec{\mathbf{k}})||_{\infty}^{\frac{1}{\lambda_d}(-\alpha H - q - \alpha)} = 2^d \cdot \sum_{\vec{\mathbf{k}} \in J'_{N,2}} ||\vec{\mathbf{k}}||_{\infty}^{\frac{1}{\lambda_d}(-\alpha H - q - \alpha)}$$

with $J'_{N,2} := \{\vec{\mathbf{k}} \in J_{N,2} : k_j \geq 0,\ 1 \leq j \leq d\} = J_{N,2} \cap \{0, \ldots, M-1\}^d$.

Therefore

$$I_2 \le 2^{1+\alpha} \cdot c_5^\alpha \cdot D^{\frac{\alpha\beta}{\lambda_d} - \alpha\beta\delta} \cdot B^{\frac{-\alpha H - q - \alpha}{\lambda_1}} \cdot (2+2D)^d$$
$$+ 2^{1+\alpha+d} \cdot c_6^\alpha \cdot D^{\frac{\alpha\beta}{\lambda_d} - \alpha\beta\delta + \frac{-\alpha H - q - \alpha}{\lambda_d} + d} \cdot \sum_{\vec{\mathbf{k}} \in J'_{N,2}} ||\vec{\mathbf{k}}||_\infty^{\frac{1}{\lambda_d}(-\alpha H - q - \alpha)} \tag{3.19}$$

Using the fact that $t(\vec{\mathbf{k}}) \in \mathbb{Z} \Rightarrow ||t(\vec{\mathbf{k}})||_\infty \in \mathbb{N}_0$, the definition of $J_{N,2}$ can be transformed as follows:

$$J_{N,2} = \{\vec{\mathbf{k}} \in J_N : \Delta_{\vec{\mathbf{k}}} \cap [-1,1)^d = \emptyset\} = \{\vec{\mathbf{k}} \in J_N : \tilde{\xi}_{\vec{\mathbf{k}}} \notin (-1,1)^d\}$$
$$= \{\vec{\mathbf{k}} \in J_N : ||D \cdot t(\vec{\mathbf{k}})||_\infty \ge 1\} = \left\{\vec{\mathbf{k}} \in J_N : ||t(\vec{\mathbf{k}})||_\infty \ge \frac{1}{D}\right\}$$
$$= \left\{\vec{\mathbf{k}} \in J_N : ||t(\vec{\mathbf{k}})||_\infty \ge \left\lceil \frac{1}{D} \right\rceil\right\}$$

so that

$$J'_{N,2} = J_{N,2} \cap \{0,\dots,M-1\}^d = \left\{\vec{\mathbf{k}} \in \{0,\dots,M-1\}^d : ||t(\vec{\mathbf{k}})||_\infty \ge \left\lceil \frac{1}{D} \right\rceil\right\}$$
$$= \left\{\vec{\mathbf{k}} \in \{0,\dots,M-1\}^d : ||\vec{\mathbf{k}}||_\infty \ge \left\lceil \frac{1}{D} \right\rceil\right\}$$
$$= \bigcup_{m=\lceil \frac{1}{D} \rceil}^{M-1} \left\{\vec{\mathbf{k}} \in \{0,\dots,M-1\}^d : ||\vec{\mathbf{k}}||_\infty = m\right\} = \bigcup_{m=\lceil \frac{1}{D} \rceil}^{M-1} L_m$$

with $L_m := \left\{\vec{\mathbf{k}} \in \{0,\dots,M-1\}^d : ||\vec{\mathbf{k}}||_\infty = m\right\}$ (obviously, the L_m are pairwise disjoint).

Thus the remaining sum in (3.19) can be estimated as follows(similar to the estimation of the sum on page 36,in the proof of Theorem 3.20):

$$\sum_{\vec{\mathbf{k}} \in J'_{N,2}} ||\vec{\mathbf{k}}||_\infty^{\frac{1}{\lambda_d}(-\alpha H - q - \alpha)} = \sum_{m=\lceil \frac{1}{D} \rceil}^{M-1} \sum_{\vec{\mathbf{k}} \in L_m} ||\vec{\mathbf{k}}||_\infty^{\frac{1}{\lambda_d}(-\alpha H - q - \alpha)}$$
$$= \sum_{m=\lceil \frac{1}{D} \rceil}^{M-1} \sum_{\vec{\mathbf{k}} \in L_m} m^{\frac{1}{\lambda_d}(-\alpha H - q - \alpha)}$$
$$\le \sum_{m=\lceil \frac{1}{D} \rceil}^{M-1} d(m+1)^{d-1} \cdot m^{\frac{1}{\lambda_d}(-\alpha H - q - \alpha)}$$

$$
\begin{aligned}
&= d \cdot \sum_{m=\lceil \frac{1}{D} \rceil}^{M-1} \left(1 + \frac{1}{m}\right)^{d-1} \cdot m^{\frac{1}{\lambda_d}(-\alpha H - q - \alpha) + d - 1} \\
&\leq d \cdot (1+D)^{d-1} \cdot \left(\left\lceil \frac{1}{D} \right\rceil^{\frac{-\alpha H - \alpha - q + d\lambda_d}{\lambda_d} - 1} + \sum_{m=\lceil \frac{1}{D} \rceil + 1}^{M-1} m^{\frac{-\alpha H - \alpha - q + d\lambda_d}{\lambda_d} - 1} \right) \\
&< d \cdot (1+D)^{d-1} \cdot \left(D^{\frac{\alpha H + \alpha + q - d\lambda_d + \lambda_d}{\lambda_d}} + \int_{\lceil \frac{1}{D} \rceil}^{M} x^{\frac{-\alpha H - \alpha - q + d\lambda_d}{\lambda_d} - 1} dx \right)
\end{aligned}
$$

The last inequalities use the implication $m \geq \frac{1}{D} \Rightarrow \frac{1}{m} \leq D \Rightarrow (1 + \frac{1}{m})^{d-1} \leq (1+D)^{d-1}$ and the fact that $\lceil \frac{1}{D} \rceil \geq \frac{1}{D} \Rightarrow \lceil \frac{1}{D} \rceil^{-1} \leq D$ together with the assumption $\alpha H + \alpha + q - d\lambda_d + \lambda_d \geq 0$.

Because of the assumption $-\alpha H - \alpha - q + d\lambda_d \neq 0$, the integral can be calculated as follows (with $\gamma := -\alpha H - \alpha - q + d\lambda_d$):

$$
\int_{\lceil \frac{1}{D} \rceil}^{M} x^{\frac{\gamma}{\lambda_d} - 1} dx = \frac{\lambda_d}{\gamma} \left[x^{\frac{\gamma}{\lambda_d}} \right]_{\lceil \frac{1}{D} \rceil}^{M} = \frac{\lambda_d}{\gamma} \left(M^{\frac{\gamma}{\lambda_d}} - \left\lceil \frac{1}{D} \right\rceil^{\frac{\gamma}{\lambda_d}} \right).
$$

If $\gamma > 0$, then $\lceil \frac{1}{D} \rceil^{\frac{\gamma}{\lambda_d}} \geq \left(\frac{1}{D}\right)^{\frac{\gamma}{\lambda_d}}$, and if $\gamma < 0$ then $\lceil \frac{1}{D} \rceil^{\frac{\gamma}{\lambda_d}} \leq \left(\frac{1}{D}\right)^{\frac{\gamma}{\lambda_d}}$. In both cases follows $\frac{\lambda_d}{\gamma} \lceil \frac{1}{D} \rceil^{\frac{\gamma}{\lambda_d}} \geq \frac{\lambda_d}{\gamma} \left(\frac{1}{D}\right)^{\frac{\gamma}{\lambda_d}}$, which implies:

$$
\int_{\lceil \frac{1}{D} \rceil}^{M} x^{\frac{\gamma}{\lambda_d} - 1} dx \leq \frac{\lambda_d}{\gamma} \left(M^{\frac{\gamma}{\lambda_d}} - \left(\frac{1}{D}\right)^{\frac{\gamma}{\lambda_d}} \right) = D^{-\frac{\gamma}{\lambda_d}} \cdot \frac{\lambda_d}{\gamma} \left(A^{\frac{\gamma}{\lambda_d}} - 1 \right).
$$

Therefore

$$
\begin{aligned}
I_2 \leq\ & 2^{1+\alpha} \cdot c_5^\alpha \cdot D^{\frac{\alpha\beta}{\lambda_d} - \alpha\beta\delta} \cdot B^{\frac{-\alpha H - q - \alpha}{\lambda_1}} \cdot (2 + 2D)^d \\
& + 2^{1+\alpha+d} \cdot c_6^\alpha \cdot D^{\frac{\alpha\beta}{\lambda_d} - \alpha\beta\delta + \frac{-\alpha H - q - \alpha}{\lambda_d} + d} \cdot \sum_{\vec{\mathbf{k}} \in J'_{N,2}} ||\vec{\mathbf{k}}||_\infty^{\frac{1}{\lambda_d}(-\alpha H - q - \alpha)} \\
<\ & 2^{1+\alpha+d} \cdot c_5^\alpha \cdot D^{\frac{\alpha\beta}{\lambda_d} - \delta'} \cdot B^{\frac{-\alpha H - q - \alpha}{\lambda_1}} \cdot (1 + D)^d \\
& + 2^{1+\alpha+d} \cdot c_6^\alpha \cdot D^{\frac{\alpha\beta}{\lambda_d} - \delta' + \frac{-\alpha H - q - \alpha + d\lambda_d}{\lambda_d}} \cdot d \cdot (1 + D)^{d-1} \\
& \cdot \left(D^{\frac{\alpha H + \alpha + q - d\lambda_d + \lambda_d}{\lambda_d}} + \int_{\lceil \frac{1}{D} \rceil}^{M} x^{\frac{-\alpha H - \alpha - q + d\lambda_d}{\lambda_d} - 1} dx \right)
\end{aligned}
$$

$$
\begin{aligned}
&\leq 2^{1+\alpha+d} \cdot c_5^\alpha \cdot D^{\frac{\alpha\beta}{\lambda_d}-\delta'} \cdot B^{\frac{-\alpha H-q-\alpha}{\lambda_1}} \cdot (1+D)^d \\
&\quad + 2^{1+\alpha+d} \cdot c_6^\alpha \cdot D^{\frac{\alpha\beta}{\lambda_d}-\delta'+\frac{-\alpha H-q-\alpha+d\lambda_d}{\lambda_d}} \cdot d \cdot (1+D)^{d-1} \\
&\quad \cdot \left(D^{\frac{\alpha H+\alpha+q-d\lambda_d+\lambda_d}{\lambda_d}} + D^{\frac{\alpha H+\alpha+q-d\lambda_d}{\lambda_d}} \cdot \frac{\lambda_d}{-\alpha H-\alpha-q+d\lambda_d} \left(A^{\frac{-\alpha H-\alpha-q+d\lambda_d}{\lambda_d}} - 1 \right) \right) \\
&\leq 2^{1+\alpha+d} \cdot c_5^\alpha \cdot D^{\frac{\alpha\beta}{\lambda_d}-\delta'} \cdot B^{\frac{-\alpha H-q-\alpha}{\lambda_1}} \cdot (1+D)^d \\
&\quad + 2^{1+\alpha+d} \cdot c_6^\alpha \cdot d \cdot D^{\frac{\alpha\beta}{\lambda_d}-\delta'} \cdot (1+D)^{d-1} \\
&\quad \cdot \left(D + \frac{\lambda_d}{-\alpha H-\alpha-q+d\lambda_d} \left(A^{\frac{-\alpha H-\alpha-q+d\lambda_d}{\lambda_d}} - 1 \right) \right) \\
&\leq D^{\frac{\alpha\beta}{\lambda_d}-\delta'} \cdot (1+D)^d \cdot \Bigg(2^{1+\alpha+d} \cdot c_5^\alpha \cdot B^{\frac{-\alpha H-q-\alpha}{\lambda_1}} \\
&\quad + 2^{1+\alpha+d} \cdot c_6^\alpha \cdot d \cdot \left(1 + \frac{\lambda_d}{-\alpha H-\alpha-q+d\lambda_d} \left(A^{\frac{-\alpha H-\alpha-q+d\lambda_d}{\lambda_d}} - 1 \right) \right) \Bigg) \\
&= \widetilde{C}_{2,A,B} \cdot D^{\frac{\alpha\beta}{\lambda_d}-\delta'} \cdot (1+D)^d
\end{aligned}
$$

with $\delta' = \alpha\beta\delta$ and

$$
\begin{aligned}
\widetilde{C}_{2,A,B} = {} & 2^{1+\alpha+d} \cdot c_5^\alpha \cdot B^{\frac{-\alpha H-q-\alpha}{\lambda_1}} \\
& + 2^{1+\alpha+d} \cdot c_6^\alpha \cdot d \cdot \left(1 + \frac{\lambda_d}{-\alpha H-\alpha-q+d\lambda_d} \left(A^{\frac{-\alpha H-\alpha-q+d\lambda_d}{\lambda_d}} - 1 \right) \right)
\end{aligned}
$$

□

Theorem 3.34. *Be X_ψ^A the approximation of a harmonizable OSSRF by truncation of the domain of integration, and $X_\psi^{A,M,N}$ a discretisation of this approximation which uses the second type of discretisation (see* (3.18)*). If the function ψ and the other parameters fulfill the assumptions stated at the beginning of this subsection, as well as also the conditions $\alpha\lambda_1 - \alpha H - q + d\lambda_1 > 0$, $-\alpha H - q + d\lambda_d \neq 0$, $-\alpha H - \alpha - q + d\lambda_d \neq 0$ and $\alpha H + \alpha + q - d\lambda_d + \lambda_d \geq 0$, then exist constants $\delta', \widetilde{C}_{0,x}$ (both independent from the parameters A, B and D) and $\widetilde{C}_{1,A,B,x}, \widetilde{C}_{2,A,B}$ (independent from D), so that the approximation error due to the discretisation can be estimated by*

$$
\begin{aligned}
& \left\| X_\psi^A(x) - X_\psi^{A,M,N}(x) \right\|_\alpha^\alpha \\
\leq & \widetilde{C}_0 \cdot B^{\frac{\alpha\lambda_1-\alpha H-q+d\lambda_1}{\lambda_1}} + \widetilde{C}_{1,A,B,x} \cdot D^\alpha + \widetilde{C}_{2,A,B} \cdot D^{\frac{\alpha\beta}{\lambda_d}-\delta'} \cdot (1+D)^d .
\end{aligned}
$$

Proof. According to the lemmas 3.14 and 3.19, the error of approximation can be estimated by

$$\begin{aligned}
&\left|\left|X_{\psi}^{A}(x) - X_{\psi}^{A,M,N}(x)\right|\right|_{\alpha}^{\alpha} \\
\leq & \sum_{\vec{\mathbf{k}}\in\{-M,\ldots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}}\right|^{\alpha} d\xi \\
\leq & I_0 + I_1 + I_2
\end{aligned}$$

with

$$\begin{aligned}
I_0 &= ||x||_2^{\alpha} \cdot \int_{[-B,B]^d} ||\xi||_2^{\alpha} \cdot \psi(\xi)^{-\alpha H - q}\, d\xi \\
I_1 &= 2 \cdot ||x||_2^{\alpha} \cdot D^{\alpha} d^{\frac{\alpha}{2}} \cdot \int_{[-A,A]^d\setminus[-B,B]^d} \psi(\xi)^{-\alpha H - q}\, d\xi \\
I_2 &= 2^{1+\alpha} \cdot \sum_{\vec{\mathbf{k}}\in J_N} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right|^{\alpha} d\xi
\end{aligned}$$

In the lemmas 3.26, 3.27 and 3.33 it has been shown that these terms can be estimated by $I_0 \leq \widetilde{C}_{0,x} \cdot B^{\frac{\alpha\lambda_1 - \alpha H - q + d\lambda_1}{\lambda_1}}$, $I_1 \leq \widetilde{C}_{1,A,B,x} \cdot D^{\alpha}$ and $I_2 < \widetilde{C}_{2,A,B} \cdot D^{\frac{\alpha\beta}{\lambda_d} - \delta'} \cdot (1+D)^d$. □

Similar to Corollary 3.21, the results for $||X_{\psi}(x) - X_{\psi}^{A}(x)||_{\alpha}^{\alpha}$ and $||X_{\psi}^{A}(x) - X_{\psi}^{A,M,N}(x)||_{\alpha}^{\alpha}$ can be combined also in this more general case in order to give an estimation of $||X_{\psi}(x) - X_{\psi}^{A,M,N}(x)||_{\alpha}^{\alpha}$:

Corollary 3.35. *Be X_{ψ} a harmonizable OSSRF, and $X_{\psi}^{A,M,N}$ an approximation of this OSSRF which uses the second type of discretisation (see* (3.18)*). If the function ψ and the other parameters fulfill the assumptions of Theorem 3.10 and Theorem 3.34, then exist constants $\delta', \widetilde{C}, \widetilde{C}_{0,x}$ (independent from the parameters A, B and D) and $\widetilde{C}_{1,A,B,x}, \widetilde{C}_{2,A,B}$ (independent from D), so that the approximation error (due to the truncation and the discretisation) can be estimated by*

$$\begin{aligned}
&\left|\left|X_{\psi}(x) - X_{\psi}^{A,M,N}(x)\right|\right|_{\alpha}^{\alpha} \\
\leq & \widetilde{C} \cdot A^{\frac{-\alpha H - q + d\lambda_d}{\lambda_d}} + \widetilde{C}_{0,x} \cdot B^{\frac{\alpha\lambda_1 - \alpha H - q + d\lambda_1}{\lambda_1}} + \widetilde{C}_{1,A,B,x} \cdot D^{\alpha} + \widetilde{C}_{2,A,B} \cdot D^{\frac{\alpha\beta}{\lambda_d} - \delta'} \cdot (1+D)^d .
\end{aligned}$$

Proof. In the proof of Corollary 3.21 it has been shown that

$$\begin{aligned}
&\left|\left|X_{\psi}(x) - X_{\psi}^{A,M,N}(x)\right|\right|_{\alpha}^{\alpha} \\
\leq & \int_{\mathbb{R}^d\setminus[-A,A]^d} \left|\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}}\right|^{\alpha} d\xi \\
& + \int_{[-A,A]^d} \left| \sum_{\vec{\mathbf{k}}\in\{-M,\ldots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(\xi) \left(\left(e^{i<x,\xi>}-1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}}\right)\right|^{\alpha} d\xi
\end{aligned}$$

which implies

$$\begin{aligned}
&\left|\left|X_\psi(x) - X_\psi^{A,M,N}(x)\right|\right|_\alpha^\alpha \\
\le &\int_{\mathbb{R}^d\setminus[-A,A]^d} \left|e^{i<x,\xi>} - 1\right|^\alpha \psi(\xi)^{-\alpha H - q} d\xi \\
&\quad + \sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\left(e^{i<x,\xi>} - 1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}}\right|^\alpha d\xi.
\end{aligned}$$

According to the proof of Theorem 3.10

$$\begin{aligned}
&\int_{\mathbb{R}^d\setminus[-A,A]^d} \left|e^{i<x,\xi>} - 1\right|^\alpha \psi(\xi)^{-\alpha H - q}\, d\xi \\
\le &\underbrace{2^\alpha \cdot C_{min}^{\frac{-\alpha H - q}{\rho}} \cdot \tilde{I} \cdot \frac{\lambda_d}{\alpha H + q - d\lambda_d}}_{:=\widetilde{C}} \cdot A^{\frac{-\alpha H - q + d\lambda_d}{\lambda_d}},
\end{aligned}$$

and according to the proof of Theorem 3.34

$$\begin{aligned}
&\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\left(e^{i<x,\xi>} - 1\right)\psi(\xi)^{-H-\frac{q}{\alpha}} - g_{\vec{\mathbf{k}}}\right|^\alpha d\xi \\
\le &\widetilde{C}_{0,x} \cdot B^{\frac{\alpha\lambda_1 - \alpha H - q + d\lambda_1}{\lambda_1}} + \widetilde{C}_{1,A,B,x} \cdot D^\alpha + \widetilde{C}_{2,A,B} \cdot D^{\frac{\alpha\beta}{\lambda_d} - \delta'} \cdot (1+D)^d.
\end{aligned}$$

Therefore

$$\begin{aligned}
&\left|\left|X_\psi(x) - X_\psi^{A,M,N}(x)\right|\right|_\alpha^\alpha \\
\le &\widetilde{C} \cdot A^{\frac{-\alpha H - q + d\lambda_d}{\lambda_d}} + \widetilde{C}_{0,x} \cdot B^{\frac{\alpha\lambda_1 - \alpha H - q + d\lambda_1}{\lambda_1}} + \widetilde{C}_{1,A,B,x} \cdot D^\alpha + \widetilde{C}_{2,A,B} \cdot D^{\frac{\alpha\beta}{\lambda_d} - \delta'} \cdot (1+D)^d.
\end{aligned}$$

□

Remark 3.36. If all assumptions on the parameter values are fulfilled, then the exponent of A in this estimation is negative and the exponents of B and D are positive. Therefore, the error estimate decreases with increasing values of A and decreasing values of B and D.

For any fixed $x \in \mathbb{R}^d$, $\left|\left|X_\psi(x) - X_\psi^{A,M,N}(x)\right|\right|_\alpha^\alpha < \varepsilon$ can be achieved for any $\varepsilon > 0$ by choosing suitable values for the approximation parameters A, B and D:

(a) First choose a sufficiently large $A > 0$ and a sufficiently small $B > 0$ so that $\widetilde{C} \cdot A^{\frac{-\alpha H - q + d\lambda_d}{\lambda_d}} < \frac{\varepsilon}{4}$ and $\widetilde{C}_{0,x} \cdot B^{\frac{\alpha\lambda_1 - \alpha H - q + d\lambda_1}{\lambda_1}} < \frac{\varepsilon}{4}$. This is possible because the exponent of A is positive and the one of B is negative.

(b) Then choose a sufficiently small value for D (depending on the values of A and B which have been selected in the previous step), so that $\widetilde{C}_{1,A,B,x} \cdot D^{\alpha} < \frac{\varepsilon}{4}$ and $\widetilde{C}_{2,A,B} \cdot D^{\frac{\alpha\beta}{\lambda_d}-\delta'} \cdot (1+D)^d < \frac{\varepsilon}{4}$. This can be done because $\alpha > 0$, $\frac{\alpha\beta}{\lambda_d} - \delta'$ is supposed to be positive, too, and the factor $(1+D)^d$ is bounded by a constant: $(1+D)^d \le 2^d$.

If the parameters are chosen so that these inequalities are fulfilled, then the previous corollary implies that $\left|\left|X_\psi(x) - X_\psi^{A,M,N}(x)\right|\right|_\alpha^\alpha < \varepsilon$.

First version of the discretisation

Corollary 3.37. *Be X_ψ^A the approximation of a harmonizable OSSRF by truncation of the domain of integration, and $X_\psi^{A,M}$ a discretisation of this approximation which uses the first type of discretisation (see (3.7)). If the function ψ and the other parameters fulfill the assumptions stated at the beginning of this subsection, as well as also the conditions $\alpha\lambda_1 - \alpha H - q + d\lambda_1 > 0$, $-\alpha H - q + d\lambda_d \neq 0$, $-\alpha H - \alpha - q + d\lambda_d \neq 0$ and $\alpha H + \alpha + q - d\lambda_d + \lambda_d \ge 0$, then exist constants $\widetilde{C}_{0,x}, \widetilde{C}_{1,x}, \widetilde{C}_{1,A,x}, \widetilde{C}_2$ and $\widetilde{C}_{2,A}$ (independent from D), so that the approximation error due to the discretisation can be estimated by*

$$\left|\left|X_\psi^A(x) - X_\psi^{A,M}(x)\right|\right|_\alpha^\alpha \le \widetilde{C}_{0,x} \cdot D^{\frac{\alpha\lambda_1 - \alpha H - q + d\lambda_1}{\lambda_1}} + \widetilde{C}_{1,x} \cdot D^{\frac{\alpha\lambda_1 - \alpha H - q + d\lambda_1}{\lambda_1}} + \widetilde{C}_{1,A,x} \cdot D^{\alpha}$$
$$+ \widetilde{C}_2 \cdot D^{\frac{-\alpha H - q - \alpha}{\lambda_1} + \frac{\alpha\beta}{\lambda_d} - \delta'} \cdot (1+D)^d + \widetilde{C}_{2,A} \cdot D^{\frac{\alpha\beta}{\lambda_d} - \delta'} \cdot (1+D)^d$$

Proof. The first version of the discretisation can be interpreted as a special case of the second version with parameter value $N = 1$ (which implies $B = D$). Therefore, we can conclude from the proof of Theorem 3.34 by replacing N by 1 and B by D, that the error of approximation can be estimated by

$$\left|\left|X_\psi^A(x) - X_\psi^{A,M}(x)\right|\right|_\alpha^\alpha \le I_0 + I_1 + I_2$$

with

$$I_0 = ||x||_2^\alpha \cdot \int_{[-D,D]^d} ||\xi||_2^\alpha \cdot \psi(\xi)^{-\alpha H - q}\, d\xi,$$
$$I_1 = 2 \cdot ||x||_2^\alpha \cdot D^\alpha d^{\frac{\alpha}{2}} \cdot \int_{[-A,A]^d \setminus [-D,D]^d} \psi(\xi)^{-\alpha H - q}\, d\xi,$$
$$I_2 = 2^{1+\alpha} \cdot \sum_{\vec{\mathbf{k}} \in J_1} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\psi(\xi)^{-H-\frac{q}{\alpha}} - \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}}\right|^\alpha d\xi.$$

By setting $N := 1$ and $B := D$ in the proofs of the lemmas 3.26, 3.27 and 3.33, the following estimations of I_0, I_1 and I_2 can be obtained:

From the proof of Lemma 3.26 follows

$$\begin{aligned}I_0 &\leq ||x||_2^\alpha \cdot C_{min}^{\frac{-\alpha H-q}{\rho}} \cdot \tilde{I} \cdot \frac{\lambda_1}{\alpha\lambda_1 - \alpha H - q + d\lambda_1} \left(\sqrt{d} \cdot D\right)^{\frac{\alpha\lambda_1-\alpha H-q+d\lambda_1}{\lambda_1}} \\ &= \widetilde{C}_{0,x} \cdot D^{\frac{\alpha\lambda_1-\alpha H-q+d\lambda_1}{\lambda_1}}\end{aligned}$$

with $\widetilde{C}_{0,x} = ||x||_2^\alpha \cdot C_{min}^{\frac{-\alpha H-q}{\rho}} \cdot \tilde{I} \cdot \frac{\lambda_1}{\alpha\lambda_1-\alpha H-q+d\lambda_1} \cdot d^{\frac{\alpha\lambda_1-\alpha H-q+d\lambda_1}{2\lambda_1}}$,

and from the proof of Lemma 3.27 follows:

$$\begin{aligned}I_1 &\leq \underbrace{2 \cdot ||x||_2^\alpha \cdot d^{\frac{\alpha}{2}} \cdot C_{min}^{\frac{-\alpha H-q}{\rho}} \cdot \tilde{I}}_{:=c_{1,x}} \cdot D^\alpha \cdot \left(\frac{\lambda_1}{\alpha H + q - d\lambda_1}\left(D^{\frac{-\alpha H-q+d\lambda_1}{\lambda_1}} - 1\right)\right. \\ &\qquad \left. + \frac{\lambda_d}{\alpha H + q - d\lambda_d}\left(1 - \left(A\sqrt{d}\right)^{\frac{-\alpha H-q+d\lambda_d}{\lambda_d}}\right)\right) \\ &< c_{1,x} \cdot D^\alpha \cdot \left(\frac{\lambda_1}{\alpha H + q - d\lambda_1} D^{\frac{-\alpha H-q+d\lambda_1}{\lambda_1}}\right. \\ &\qquad \left. + \frac{\lambda_d}{\alpha H + q - d\lambda_d}\left(1 - \left(A\sqrt{d}\right)^{\frac{-\alpha H-q+d\lambda_d}{\lambda_d}}\right)\right) \\ &= \frac{c_{1,x} \cdot \lambda_1}{\alpha H + q - d\lambda_1} D^{\frac{-\alpha H-q+d\lambda_1}{\lambda_1} + \alpha} \\ &\qquad + \frac{c_{1,x} \cdot \lambda_d}{\alpha H + q - d\lambda_d}\left(1 - \left(A\sqrt{d}\right)^{\frac{-\alpha H-q+d\lambda_d}{\lambda_d}}\right) \cdot D^\alpha \\ &= \widetilde{C}_{1,x} \cdot D^{\frac{\alpha\lambda_1-\alpha H-q+d\lambda_1}{\lambda_1}} + \widetilde{C}_{1,A,x} \cdot D^\alpha\end{aligned}$$

with

$$\widetilde{C}_{1,x} = \frac{c_{1,x} \cdot \lambda_1}{\alpha H + q - d\lambda_1} \quad \text{and} \quad \widetilde{C}_{1,A,x} = \frac{c_{1,x} \cdot \lambda_d}{\alpha H + q - d\lambda_d}\left(1 - \left(A\sqrt{d}\right)^{\frac{-\alpha H-q+d\lambda_d}{\lambda_d}}\right)$$

where

$$c_{1,x} = 2 \cdot ||x||_2^\alpha \cdot d^{\frac{\alpha}{2}} \cdot C_{min}^{\frac{-\alpha H-q}{\rho}} \cdot \tilde{I}.$$

Finally, the proof of Lemma 3.33 implies:

$$\begin{aligned}I_2 &\leq D^{\frac{\alpha\beta}{\lambda_d}-\delta'} \cdot (1+D)^d \cdot \left(2^{1+\alpha+d} \cdot c_5^\alpha \cdot D^{\frac{-\alpha H-q-\alpha}{\lambda_1}}\right. \\ &\qquad \left. + 2^{1+\alpha+d} \cdot c_6^\alpha \cdot d \cdot \left(1 + \frac{\lambda_d}{-\alpha H - \alpha - q + d\lambda_d}\left(A^{\frac{-\alpha H-\alpha-q+d\lambda_d}{\lambda_d}} - 1\right)\right)\right) \\ &= D^{\frac{\alpha\beta}{\lambda_d}-\delta'} \cdot (1+D)^d \cdot \left(\widetilde{C}_2 \cdot D^{\frac{-\alpha H-q-\alpha}{\lambda_1}} + \widetilde{C}_{2,A}\right) \\ &= \widetilde{C}_2 \cdot D^{\frac{-\alpha H-q-\alpha}{\lambda_1}+\frac{\alpha\beta}{\lambda_d}-\delta'} \cdot (1+D)^d + \widetilde{C}_{2,A} \cdot D^{\frac{\alpha\beta}{\lambda_d}-\delta'} \cdot (1+D)^d\end{aligned}$$

with

$$\widetilde{C}_2 = 2^{1+\alpha+d} \cdot c_5^\alpha$$

and

$$\widetilde{C}_{2,A} = 2^{1+\alpha+d} \cdot c_6^\alpha \cdot d \cdot \left(1 + \frac{\lambda_d}{-\alpha H - \alpha - q + d\lambda_d}\left(A^{\frac{-\alpha H - \alpha - q + d\lambda_d}{\lambda_d}} - 1\right)\right).$$

□

Remark 3.38. Like already in the case of a ρ-norm as E-homogeneous function ψ, no convergence of the error estimate to 0 for $D \to 0$ can be shown when using the first version of discretization, because the exponent $\frac{-\alpha H - q - \alpha}{\lambda_1} + \frac{\alpha\beta}{\lambda_d} - \delta'$ can't be assumed to be positive (compare Remark 3.18).

Chapter 4

Approximation of OSSRFs in moving average representation

4.1 Approximation

Be X_φ an OSSRF in moving average representation on the $\mathbb{R}^d$. In order to simulate this random field numerically, the integral which occurs in its definition (see (2.5)) has to be approximated by a finite sum (analogous to the harmonizable case) . This approximation is performed in two steps:

(a) Truncation of the domain of integration: The scope of the integration is narrowed from the infinite space $\mathbb{R}^d$ to the finite space $[-A, A]^d$ (for a parameter value $A > 1$). Thus, $X_\varphi(x)$ is approximated by

$$X_\varphi^A(x) = \int_{[-A,A]^d} \left(\varphi(x-y)^{H-\frac{q}{\alpha}} - \varphi(-y)^{H-\frac{q}{\alpha}} \right) Z_\alpha(dy) \tag{4.1}$$

(b) Discretization: The integral on $[-A, A]^d$ is approximated by a finite sum, by dividing $[-A, A]^d$ into $(2M)^d$ small hypercubes $\Delta_{\vec{\mathbf{k}}}$ (for $\vec{\mathbf{k}} = (k_1, \ldots, k_d)^T,\ -M \leq k_j \leq M-1,\ 1 \leq j \leq d$) with equal side length $D := \frac{A}{M}$, and approximating the function $\varphi(x-y)^{H-\frac{q}{\alpha}} - \varphi(-y)^{H-\frac{q}{\alpha}}$ on each $\Delta_{\vec{\mathbf{k}}}$ by a constant function. For each $\vec{\mathbf{k}} \in \{-M, \ldots, M-1\}^d$, $y_{\vec{\mathbf{k}}}$ is defined by $y_{\vec{\mathbf{k}}} := D \cdot \vec{\mathbf{k}} = (D \cdot k_1, \ldots, D \cdot k_d)^T$ and $\Delta_{\vec{\mathbf{k}}}$ by $\Delta_{\vec{\mathbf{k}}} = \Delta_{k_1,\ldots,k_d} := [k_1 \cdot D, (k_1+1) \cdot D) \times \ldots \times [k_d \cdot D, (k_d+1) \cdot D)$. For $y \in \Delta_{\vec{\mathbf{k}}}$, the term $\varphi(x-y)^{H-\frac{q}{\alpha}} - \varphi(-y)^{H-\frac{q}{\alpha}}$ is approximated by $\tilde{\varphi}(x - y_{\vec{\mathbf{k}}}) - \tilde{\varphi}(-y_{\vec{\mathbf{k}}})$ with $\tilde{\varphi}$ being defined by

$$\tilde{\varphi}(z) := \begin{cases} \varphi(z)^{H-\frac{q}{\alpha}}, & z \notin [-D, D)^d \\ 0, & z \in [-D, D)^d \end{cases} \tag{4.2}$$

(for $z \in \mathbb{R}^d$), so that X_φ^A is approximated by

$$X_\varphi^{A,M}(x) = \sum_{\vec{\mathbf{k}} \in \{-M,\ldots,M-1\}^d} \left(\tilde{\varphi}(x - y_{\vec{\mathbf{k}}}) - \tilde{\varphi}(-y_{\vec{\mathbf{k}}})\right) Z_\alpha(\Delta_{\vec{\mathbf{k}}}) \tag{4.3}$$

with (preferably large) parameters $A > 0$ and $M \in \mathbb{N}$.

It is assumed that the exponent $H - \frac{q}{\alpha}$ is negative. Therefore, the value of $\varphi(y)^{H-\frac{q}{\alpha}}$ is replaced by 0 in the approximation $\tilde{\varphi}$ in an environment of the origin, in order to avoid the calculation with infinite values during the simulation and in order to be able to estimate the errors of approximation.

4.2 Approximation error due to the truncation

As already in the harmonizable case, it is desired to estimate the errors of the approximations ($X_\varphi^A(x) - X_\varphi(x)$ and $X_\varphi^{A,M}(x) - X_\varphi^A(x)$) for a given $x \in \mathbb{R}^d$, i.e. to calculate an upper bound for the absolute values of these differences. Because $X_\varphi(x)$, $X_\varphi^A(x)$ and $X_\varphi^{A,M}(x)$ are symmetric α-stable ($S\alpha S$) random variables, their differences are $S\alpha S$ random variables, too. Therefore the scale parameters of their distributions ($||X_\varphi^A(x) - X_\varphi(x)||_\alpha$ and $||X_\varphi^{A,M}(x) - X_\varphi^A(x)||_\alpha$) have to be estimated as a measure of the approximation errors. As mentioned in section 3.2, $|| \int_{\mathbb{R}^d} f(y)\, Z_\alpha(dy)||_\alpha$ can be calculated by $|| \int_{\mathbb{R}^d} f(y)\, Z_\alpha(dy)||_\alpha^\alpha = \int_{\mathbb{R}^d} |f(y)|^\alpha \, dy$ (the latter being a non-random integral).

In this chapter, estimations for $||X_\varphi^A(x) - X_\varphi(x)||_\alpha^\alpha$ and $||X_\varphi^{A,M}(x) - X_\varphi^A(x)||_\alpha^\alpha$ are calculated in the case that the function φ is a ρ-norm (i.e. $\varphi = ||\cdot||_\rho$ with $\rho \geq 1$). Such a norm can be obtained as a special case of an E^T-homogeneous and (β, E)-admissible function in the representation of (2.2) by setting the parameter values to $\lambda_1 = \ldots = \lambda_d = 1$ (which implies $q = \lambda_1 + \ldots + \lambda_d = 1 + \ldots + 1 = d$), $C_1 = \ldots = C_d = 1$ and the vectors $\theta_1, \ldots, \theta_d$ to the base of standard unit vectors of the $\mathbb{R}^d$.

Theorem 4.1. *If the function φ is a ρ-norm (i.e. $\varphi = ||\cdot||_\rho$ with $\rho \geq 1$), and the inequalities $0 < H < 1$, $H - \frac{d}{\alpha} < 0$ and $A \geq 2 \cdot ||x||_\rho$ are fulfilled, then the approximation error can be estimated as follows:*

$$||X_\varphi(x) - X_\varphi^A(x)||_\alpha^\alpha \leq \widetilde{C}_{1,x}^\alpha \cdot ||x||_\rho^\alpha \cdot A^{-\alpha(1-H)}$$

with

$$\widetilde{C}_{1,x} = \left(\frac{d}{\alpha} - H\right) \cdot 2^{1-H} \cdot \left(\frac{d \cdot 2^{2d}}{\alpha(1-H)}\right)^{1/\alpha},$$

which implies that

$$||X_\varphi(x) - X_\varphi^A(x)||_\alpha \leq \widetilde{C}_{1,x} \cdot ||x||_\rho \cdot A^{H-1}.$$

Proof. If $\varphi = ||\cdot||_\rho$ (implying $q = d$), then

$$||X_\varphi(x) - X_\varphi^A(x)||_\alpha^\alpha = \left|\left|\int_{\mathbb{R}^d}\left(||x-y||_\rho^{H-\frac{d}{\alpha}} - ||-y||_\rho^{H-\frac{d}{\alpha}}\right)\, Z_\alpha(dy) \right.\right.$$
$$\left.\left. - \int_{[-A,A]^d}\left(||x-y||_\rho^{H-\frac{d}{\alpha}} - \varphi||-y||_\rho^{H-\frac{d}{\alpha}}\right)\, Z_\alpha(dy)\right|\right|_\alpha^\alpha$$

Using the above-mentioned equality $||\int_{\mathbb{R}^d} f(y)\, Z_\alpha(dy)||_\alpha^\alpha = \int_{\mathbb{R}^d}|f(y)|^\alpha\, dy$, this can be transformed to

$$\begin{aligned}
&||X_\varphi(x) - X_\varphi^A(x)||_\alpha^\alpha \\
=&\left|\left|\int_{\mathbb{R}^d\setminus[-A,A]^d}\left(||x-y||_\rho^{H-\frac{d}{\alpha}} - ||-y||_\rho^{H-\frac{d}{\alpha}}\right)\, Z_\alpha(dy)\right|\right|_\alpha^\alpha \\
=&\int_{||y||_\infty > A}\left|||x-y||_\rho^{H-\frac{d}{\alpha}} - ||-y||_\rho^{H-\frac{d}{\alpha}}\right|^\alpha dy \\
=&\int_{||y||_\infty > A}\left|||x+y||_\rho^{H-\frac{d}{\alpha}} - ||y||_\rho^{H-\frac{d}{\alpha}}\right|^\alpha dy \\
=&\int_{||y||_\infty > A}\left|||y||_\rho^{H-\frac{d}{\alpha}}\left(\frac{||x+y||_\rho^{H-d/\alpha}}{||y||_\rho^{H-d/\alpha}} - 1\right)\right|^\alpha dy \\
=&\int_{||y||_\infty > A}||y||_\rho^{\alpha H-d}\left|\left(\frac{||x+y||_\rho}{||y||_\rho}\right)^{H-d/\alpha} - 1\right|^\alpha dy
\end{aligned}$$

For $||y||_\infty > A$ the inequality

$$\begin{aligned}
\left|\frac{||x+y||_\rho}{||y||_\rho} - 1\right| &= \left|\left|\left|\frac{x}{||y||_\rho} + \frac{y}{||y||_\rho}\right|\right|_\rho - \left|\left|\frac{y}{||y||_\rho}\right|\right|_\rho\right| \\
&\le \left|\left|\frac{x}{||y||_\rho}\right|\right|_\rho = \frac{||x||_\rho}{||y||_\rho} \le \frac{A}{2\cdot||y||_\rho} < \frac{||y||_\infty}{2\cdot||y||_\rho} \le \frac{1}{2}
\end{aligned}$$

holds (following from the triangle inequality for the norm $||\cdot||_\rho$, and the assumptions that $||y||_\infty > A$ and $A \ge 2\cdot||x||_\rho$), and therefore

$$\frac{||x+y||_\rho}{||y||_\rho} \ge 1 - \frac{1}{2} = \frac{1}{2},$$

which implies together with the Mean Value Theorem

$$
\begin{aligned}
&\left| \left(\frac{||x+y||_\rho}{||y||_\rho} \right)^{H-d/\alpha} - 1^{H-d/\alpha} \right| \\
\leq &\left| \frac{||x+y||_\rho}{||y||_\rho} - 1 \right| \cdot \left| \left(H - \frac{d}{\alpha} \right) \cdot \max \left\{ \left(\frac{||x+y||_\rho}{||y||_\rho} \right)^{H-d/\alpha-1}, 1^{H-d/\alpha-1} \right\} \right| \\
\leq &\left| \frac{||x+y||_\rho}{||y||_\rho} - 1 \right| \cdot \left| H - \frac{d}{\alpha} \right| \cdot \left| \left(\frac{1}{2} \right)^{H-d/\alpha-1} \right| \\
\leq &\frac{||x||_\rho}{||y||_\rho} \cdot \left(\frac{d}{\alpha} - H \right) \cdot 2^{-H+d/\alpha+1}
\end{aligned}
$$

and thus (using $||y||_\infty \leq ||y||_\rho$)

$$
\begin{aligned}
&||X_\varphi(x) - X_\varphi^A(x)||_\alpha^\alpha \\
= &\int_{||y||_\infty > A} ||y||_\rho^{\alpha H - d} \left| \left(\frac{||x+y||_\rho}{||y||_\rho} \right)^{H-d/\alpha} - 1 \right|^\alpha dy \\
\leq &\int_{||y||_\infty > A} ||y||_\rho^{\alpha H - d} \left(\frac{||x||_\rho}{||y||_\rho} \cdot \left(\frac{d}{\alpha} - H \right) \cdot 2^{-H+\frac{d}{\alpha}+1} \right)^\alpha dy \\
\leq &\left(\frac{d}{\alpha} - H \right)^\alpha \cdot 2^{d+\alpha-\alpha H} \cdot ||x||_\rho^\alpha \cdot \int_{||y||_\infty > A} ||y||_\rho^{\alpha H - d} \cdot ||y||_\rho^{-\alpha} \, dy \\
\leq &\left(\frac{d}{\alpha} - H \right)^\alpha \cdot 2^{d+\alpha-\alpha H} \cdot ||x||_\rho^\alpha \cdot \int_{||y||_\infty > A} ||y||_\infty^{\alpha H - d - \alpha} \, dy \\
\leq &\left(\frac{d}{\alpha} - H \right)^\alpha \cdot 2^{d+\alpha-\alpha H} \cdot ||x||_\rho^\alpha \cdot \int_A^\infty \int_{S_\infty} r^{\alpha H - d - \alpha} \cdot r^{d-1} \; d\zeta \, dr \\
= &\left(\frac{d}{\alpha} - H \right)^\alpha \cdot 2^{d+\alpha-\alpha H} \cdot ||x||_\rho^\alpha \cdot \int_{S_\infty} 1 \, d\zeta \cdot \int_A^\infty r^{\alpha H - 1 - \alpha} \, dr \\
\underset{L.3.3}{=} &\left(\frac{d}{\alpha} - H \right)^\alpha \cdot 2^{d+\alpha-\alpha H} \cdot ||x||_\rho^\alpha \cdot \left(d \cdot 2^d \right) \cdot \left[\frac{1}{\alpha H - \alpha} \cdot r^{\alpha H - \alpha} \right]_A^\infty \\
= &\frac{\left(\frac{d}{\alpha} - H \right)^\alpha \cdot d \cdot 2^{2d+\alpha(1-H)}}{\alpha(1-H)} \cdot ||x||_\rho^\alpha \cdot A^{-\alpha(1-H)} \\
= &\widetilde{C}_{1,x}^\alpha \cdot ||x||_\rho^\alpha \cdot A^{-\alpha(1-H)}
\end{aligned}
$$

with

$$
\widetilde{C}_{1,x} = \left(\frac{d}{\alpha} - H \right) \cdot 2^{1-H} \cdot \left(\frac{d \cdot 2^{2d}}{\alpha(1-H)} \right)^{1/\alpha}
$$

□

4.3 Approximation error due to the discretisation

The algorithm for the simulation of OSSRFs in moving average representation, which is presented in this thesis, approximates the values of an OSSRF in the points $x_{\vec{\mathbf{m}}} = \vec{\mathbf{m}} \cdot D$ for $\vec{\mathbf{m}} \in \{-M, \dots, M-1\}^d$. In this section, an estimation for the approximation error of the discretisation in the points $x_{\vec{\mathbf{m}}}$ with $||\vec{\mathbf{m}}||_\infty < M$ is given (again for the case of the function φ being a ρ-norm).

Lemma 4.2. *Be $x_{\vec{\mathbf{m}}} = \vec{\mathbf{m}} \cdot D$ for a $\vec{\mathbf{m}} \in \{-M, \dots, M-1\}^d$. Then*

$$\begin{aligned}
&\left|\left|X_\varphi^{A,M}(x_{\vec{\mathbf{m}}}) - X_\varphi^A(x_{\vec{\mathbf{m}}})\right|\right|_\alpha^\alpha \\
\leq& 2 \sum_{\vec{\mathbf{k}} \in \left(\{-M,\dots,M-1\}^d - \vec{\mathbf{m}}\right)} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\tilde{\varphi}(-y_{\vec{\mathbf{k}}}) - \varphi(-y)^{H-\frac{q}{\alpha}}\right|^\alpha dy \\
&+ 2 \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\tilde{\varphi}(-y_{\vec{\mathbf{k}}}) - \varphi(-y)^{H-\frac{q}{\alpha}}\right|^\alpha dy
\end{aligned}$$

Proof. For $x \in \mathbb{R}^d$, the term $||X_\varphi^{A,M}(x) - X_\varphi^A(x)||_\alpha^\alpha$ can be transformed to

$$\begin{aligned}
&\left|\left|X_\varphi^{A,M}(x) - X_\varphi^A(x)\right|\right|_\alpha^\alpha \\
=&\Bigg|\Bigg| \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} \left(\tilde{\varphi}(x - y_{\vec{\mathbf{k}}}) - \tilde{\varphi}(-y_{\vec{\mathbf{k}}})\right) Z_\alpha(\Delta_{\vec{\mathbf{k}}}) \\
&\quad - \int_{[-A,A]^d} \left(\varphi(x-y)^{H-\frac{q}{\alpha}} - \varphi(-y)^{H-\frac{q}{\alpha}}\right) Z_\alpha(dy) \Bigg|\Bigg|_\alpha^\alpha \\
=&\Bigg|\Bigg| \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} \left(\tilde{\varphi}(x - y_{\vec{\mathbf{k}}}) - \tilde{\varphi}(-y_{\vec{\mathbf{k}}})\right) \int_{[-A,A]^d} 1_{\Delta_{\vec{\mathbf{k}}}}(y)\, Z_\alpha(dy) \\
&\quad - \int_{[-A,A]^d} \left(\varphi(x-y)^{H-\frac{q}{\alpha}} - \varphi(-y)^{H-\frac{q}{\alpha}}\right) \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(y)\, Z_\alpha(dy) \Bigg|\Bigg|_\alpha^\alpha \\
=&\Bigg|\Bigg| \int_{[-A,A]^d} \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(y) \cdot \left(\tilde{\varphi}(x - y_{\vec{\mathbf{k}}}) - \tilde{\varphi}(-y_{\vec{\mathbf{k}}})\right)\, Z_\alpha(dy) \\
&\quad - \int_{[-A,A]^d} \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(y) \cdot \left(\varphi(x-y)^{H-\frac{q}{\alpha}} - \varphi(-y)^{H-\frac{q}{\alpha}}\right)\, Z_\alpha(dy) \Bigg|\Bigg|_\alpha^\alpha
\end{aligned}$$

$$
\begin{aligned}
=&\int_{[-A,A]^d} \Bigg| \sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(y)\cdot\big(\tilde{\varphi}(x-y_{\vec{\mathbf{k}}})-\tilde{\varphi}(-y_{\vec{\mathbf{k}}})\big)\\
&\qquad - \sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(y)\cdot\Big(\varphi(x-y)^{H-\frac{q}{\alpha}}-\varphi(-y)^{H-\frac{q}{\alpha}}\Big)\Bigg|^{\alpha} dy\\
=&\int_{[-A,A]^d} \sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} 1_{\Delta_{\vec{\mathbf{k}}}}(y)\\
&\qquad\cdot\left|\big(\tilde{\varphi}(x-y_{\vec{\mathbf{k}}})-\tilde{\varphi}(-y_{\vec{\mathbf{k}}})\big)-\Big(\varphi(x-y)^{H-\frac{q}{\alpha}}-\varphi(-y)^{H-\frac{q}{\alpha}}\Big)\right|^{\alpha} dy\\
=&\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d}\int_{\Delta_{\vec{\mathbf{k}}}}\left|\tilde{\varphi}(x-y_{\vec{\mathbf{k}}})-\tilde{\varphi}(-y_{\vec{\mathbf{k}}})-\varphi(x-y)^{H-\frac{q}{\alpha}}+\varphi(-y)^{H-\frac{q}{\alpha}}\right|^{\alpha} dy\\
\le&\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d}\int_{\Delta_{\vec{\mathbf{k}}}}\left(\left|\tilde{\varphi}(x-y_{\vec{\mathbf{k}}})-\varphi(x-y)^{H-\frac{q}{\alpha}}\right|+\left|-\tilde{\varphi}(-y_{\vec{\mathbf{k}}})+\varphi(-y)^{H-\frac{q}{\alpha}}\right|\right)^{\alpha} dy\\
\le&2\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d}\int_{\Delta_{\vec{\mathbf{k}}}}\left|\tilde{\varphi}(x-y_{\vec{\mathbf{k}}})-\varphi(x-y)^{H-\frac{q}{\alpha}}\right|^{\alpha} dy\\
&+2\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d}\int_{\Delta_{\vec{\mathbf{k}}}}\left|\tilde{\varphi}(-y_{\vec{\mathbf{k}}})-\varphi(-y)^{H-\frac{q}{\alpha}}\right|^{\alpha} dy
\end{aligned}
$$

If $x = x_{\vec{\mathbf{m}}} = \vec{\mathbf{m}}\cdot D$, then

$$
\begin{aligned}
&\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d}\int_{\Delta_{\vec{\mathbf{k}}}}\left|\tilde{\varphi}(x-y_{\vec{\mathbf{k}}})-\varphi(x-y)^{H-\frac{q}{\alpha}}\right|^{\alpha} dy\\
=&\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d}\int_{\Delta_{\vec{\mathbf{k}}}}\left|\tilde{\varphi}(\vec{\mathbf{m}}\cdot D-\vec{\mathbf{k}}\cdot D)-\varphi(\vec{\mathbf{m}}\cdot D-y)^{H-\frac{q}{\alpha}}\right|^{\alpha} dy\\
=&\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d}\int_{\Delta_{\vec{\mathbf{k}}}}\left|\tilde{\varphi}((\vec{\mathbf{m}}-\vec{\mathbf{k}})\cdot D)-\varphi(\vec{\mathbf{m}}\cdot D-y)^{H-\frac{q}{\alpha}}\right|^{\alpha} dy\\
=&\sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d}\int_{\Delta_{(\vec{\mathbf{k}}-\vec{\mathbf{m}})}}\left|\tilde{\varphi}((\vec{\mathbf{m}}-\vec{\mathbf{k}})\cdot D)-\varphi(-y)^{H-\frac{q}{\alpha}}\right|^{\alpha} dy\\
=&\sum_{k_1=-M-m_1}^{M-1-m_1}\cdots\sum_{k_d=-M-m_d}^{M-1-m_d}\int_{\Delta_{\vec{\mathbf{k}}}}\left|\tilde{\varphi}(-\vec{\mathbf{k}}\cdot D)-\varphi(-y)^{H-\frac{q}{\alpha}}\right|^{\alpha} dy\\
=&\sum_{\vec{\mathbf{k}}\in\left(\{-M,\dots,M-1\}^d-\vec{\mathbf{m}}\right)}\int_{\Delta_{\vec{\mathbf{k}}}}\left|\tilde{\varphi}(-y_{\vec{\mathbf{k}}})-\varphi(-y)^{H-\frac{q}{\alpha}}\right|^{\alpha} dy
\end{aligned}
$$

□

Theorem 4.3. *If the E-homogeneous, (β, E)-admissible function φ is a ρ-norm (i.e. $\varphi = ||\cdot||_\rho$ with $\rho \geq 1$), the parameter H is in the interval $(0,1)$, the exponent $H - \frac{q}{\alpha}$ is negative (i.e. $\alpha H < q$), and $x = x_{\vec{\mathbf{m}}}$ with $\vec{\mathbf{m}} \in \{-M+1, \dots, M-1\}^d$, then the approximation error in the point $x_{\vec{\mathbf{m}}}$ can be estimated as*

$$\left|\left|X_\varphi^{A,M}(x_{\vec{\mathbf{m}}}) - X_\varphi^A(x_{\vec{\mathbf{m}}})\right|\right|_\alpha^\alpha \leq \widetilde{C}_2 \cdot D^{\alpha H}$$

with

$$\widetilde{C}_2 = \frac{d \cdot 2^{d+2}}{\alpha H} + 2^3 \cdot d^{\alpha+1} \cdot \left(\frac{d}{\alpha} - H\right)^\alpha \cdot \left(4^{d-1} + \frac{3^{d-1}}{\alpha(1-H)}\right),$$

which implies

$$\left|\left|X_\varphi^{A,M}(x_{\vec{\mathbf{m}}}) - X_\varphi^A(x_{\vec{\mathbf{m}}})\right|\right|_\alpha \leq \widetilde{C}_2^{1/\alpha} \cdot D^H.$$

Proof. Because the condition $y_{\vec{\mathbf{k}}} = D \cdot \vec{\mathbf{k}} \in [-D, D)^d$ is equivalent to $\vec{\mathbf{k}} \in \{-1, 0\}^d$, the definition of $\tilde{\varphi}$ in (4.2) implies that

$$\left|\tilde{\varphi}(-y_{\vec{\mathbf{k}}}) - \varphi(-y)^{H-\frac{q}{\alpha}}\right| = \begin{cases} \left|\varphi(-y_{\vec{\mathbf{k}}})^{H-\frac{q}{\alpha}} - \varphi(-y)^{H-\frac{q}{\alpha}}\right|, & \vec{\mathbf{k}} \notin \{-1,0\}^d \\ \varphi(-y)^{H-\frac{q}{\alpha}}, & \vec{\mathbf{k}} \in \{-1,0\}^d \end{cases} \tag{4.4}$$

Therefore the cases $\vec{\mathbf{k}} \in \{-1,0\}^d$ and $\vec{\mathbf{k}} \notin \{-1,0\}^d$ have to be distinguished in the following. Because $\{-1,0\}^d \subset \{-M, \dots, M-1\}^d$ and (following from the assumption $\vec{\mathbf{m}} \in \{-M+1, \dots, M-1\}^d$, i.e. $||\vec{\mathbf{m}}||_\infty < M$) also $\{-1,0\}^d \subset \left(\{-M, \dots, M-1\}^d - \vec{\mathbf{m}}\right)$, Equation (4.4) and Lemma 4.2 imply that

$$\begin{aligned}
&\left|\left|X_\varphi^{A,M}(x_{\vec{\mathbf{m}}}) - X_\varphi^A(x_{\vec{\mathbf{m}}})\right|\right|_\alpha^\alpha \\
\leq& 2 \sum_{\vec{\mathbf{k}} \in \left(\{-M,\dots,M-1\}^d - \vec{\mathbf{m}}\right)} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\tilde{\varphi}(-y_{\vec{\mathbf{k}}}) - \varphi(-y)^{H-\frac{q}{\alpha}}\right|^\alpha dy \\
&+ 2 \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\tilde{\varphi}(-y_{\vec{\mathbf{k}}}) - \varphi(-y)^{H-\frac{q}{\alpha}}\right|^\alpha dy \\
=& 4 \sum_{\vec{\mathbf{k}} \in \{-1,0\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\tilde{\varphi}(-y_{\vec{\mathbf{k}}}) - \varphi(-y)^{H-\frac{q}{\alpha}}\right|^\alpha dy + 2 \sum_{\vec{\mathbf{k}} \in J_1} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\tilde{\varphi}(-y_{\vec{\mathbf{k}}}) - \varphi(-y)^{H-\frac{q}{\alpha}}\right|^\alpha dy \\
&+ 2 \sum_{\vec{\mathbf{k}} \in J_2} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\tilde{\varphi}(-y_{\vec{\mathbf{k}}}) - \varphi(-y)^{H-\frac{q}{\alpha}}\right|^\alpha dy \\
=& 4 \sum_{\vec{\mathbf{k}} \in \{-1,0\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} \varphi(-y)^{\alpha H - q} dy + 2 \sum_{\vec{\mathbf{k}} \in J_1} \int_{\Delta_{\vec{\mathbf{k}}}} \left|\varphi(-y_{\vec{\mathbf{k}}})^{H-\frac{q}{\alpha}} - \varphi(-y)^{H-\frac{q}{\alpha}}\right|^\alpha dy
\end{aligned}$$

$$+ 2 \sum_{\vec{\mathbf{k}} \in J_2} \int_{\Delta_{\vec{\mathbf{k}}}} \left| \varphi(-y_{\vec{\mathbf{k}}})^{H-\frac{q}{\alpha}} - \varphi(-y)^{H-\frac{q}{\alpha}} \right|^{\alpha} dy$$

with $J_1 := \left(\{-M, \ldots, M-1\}^d - \vec{\mathbf{m}} \right) \setminus \{-1, 0\}^d$ and $J_2 := \{-M, \ldots, M-1\}^d \setminus \{-1, 0\}^d$. Because $\varphi = ||\cdot||_\rho$ (and therefore $q = \lambda_1 + \ldots + \lambda_d = d$), this implies

$$\begin{aligned}
\left|\left| X_\varphi^{A,M}(x_{\vec{\mathbf{m}}}) - X_\varphi^A(x_{\vec{\mathbf{m}}}) \right|\right|_\alpha^\alpha \leq & 4 \sum_{\vec{\mathbf{k}} \in \{-1,0\}^d} \int_{\Delta_{\vec{\mathbf{k}}}} ||y||_\rho^{\alpha H - d} \, dy \\
& + 2 \sum_{\vec{\mathbf{k}} \in J_1} \int_{\Delta_{\vec{\mathbf{k}}}} \left| ||y_{\vec{\mathbf{k}}}||_\rho^{H-\frac{d}{\alpha}} - ||y||_\rho^{H-\frac{d}{\alpha}} \right|^\alpha dy \\
& + 2 \sum_{\vec{\mathbf{k}} \in J_2} \int_{\Delta_{\vec{\mathbf{k}}}} \left| ||y_{\vec{\mathbf{k}}}||_\rho^{H-\frac{d}{\alpha}} - ||y||_\rho^{H-\frac{d}{\alpha}} \right|^\alpha dy \\
= & 4 \int_{[-D,D)^d} ||y||_\rho^{\alpha H - d} \, dy + 2 \sum_{\vec{\mathbf{k}} \in J_1} I_{\vec{\mathbf{k}}} + 2 \sum_{\vec{\mathbf{k}} \in J_2} I_{\vec{\mathbf{k}}}
\end{aligned}$$

with $I_{\vec{\mathbf{k}}} := \int_{\Delta_{\vec{\mathbf{k}}}} \left| ||y_{\vec{\mathbf{k}}}||_\rho^{H-\frac{d}{\alpha}} - ||y||_\rho^{H-\frac{d}{\alpha}} \right|^\alpha dy$

The first integral can be estimated as follows (with S_∞ being defined as in Lemma 3.3):

$$\begin{aligned}
\int_{[-D,D)^d} ||y||_\rho^{\alpha H - d} \, dy \leq \int_{[-D,D)^d} ||y||_\infty^{\alpha H - d} \, dy &= \int_0^D \int_{S_\infty} ||r \cdot \zeta||_\infty^{\alpha H - d} \, d\zeta \ \cdot r^{d-1} \, dr \\
&= \int_0^D r^{\alpha H - 1} \, dr \cdot \int_{S_\infty} 1 \, d\zeta = \frac{1}{\alpha H} \cdot \left[r^{\alpha H} \right]_0^D \cdot d \cdot 2^d \\
&= \frac{d \cdot 2^d}{\alpha H} \cdot D^{\alpha H}
\end{aligned}$$

For the estimation of the integrals $I_{\vec{\mathbf{k}}}$, the Mean Value Theorem can be used:

$$\left| ||y_{\vec{\mathbf{k}}}||_\rho^{H-\frac{d}{\alpha}} - ||y||_\rho^{H-\frac{d}{\alpha}} \right| \leq \left| \, ||y_{\vec{\mathbf{k}}}||_\rho - ||y||_\rho \, \right| \cdot \left| \left(H - \frac{d}{\alpha} \right) \cdot \mu_{\vec{\mathbf{k}}}^{H-\frac{d}{\alpha}-1} \right|$$

with $\mu_{\vec{\mathbf{k}}} := \inf_{y \in \Delta_{\vec{\mathbf{k}}}} (||y||_\rho)$ (analogous to (3.12) in the proof of Theorem 3.16). As

$$\left| \, ||y_{\vec{\mathbf{k}}}||_\rho - ||y||_\rho \, \right| \leq ||y_{\vec{\mathbf{k}}} - y||_\rho \leq ||(D, \ldots, D)^T||_\rho \leq d \cdot ||(D, \ldots, D)^T||_\infty = d \cdot D$$

and $H - \frac{d}{\alpha} < 0$, it follows that

$$\left| ||y_{\vec{\mathbf{k}}}||_\rho^{H-\frac{d}{\alpha}} - ||y||_\rho^{H-\frac{d}{\alpha}} \right| \leq d \cdot D \cdot \left(\frac{d}{\alpha} - H \right) \cdot \mu_{\vec{\mathbf{k}}}^{H-\frac{d}{\alpha}-1}$$

which implies that

$$\begin{aligned}
I_{\vec{\mathbf{k}}} =& \int_{\Delta_{\vec{\mathbf{k}}}} \left| ||y_{\vec{\mathbf{k}}}||_\rho^{H-\frac{d}{\alpha}} - ||y||_\rho^{H-\frac{d}{\alpha}} \right|^\alpha \, dy \\
=& \int_{\Delta_{\vec{\mathbf{k}}}} d^\alpha \cdot D^\alpha \cdot \left(\frac{d}{\alpha} - H\right)^\alpha \cdot \mu_{\vec{\mathbf{k}}}^{\alpha H - d - \alpha} \, dy \\
=& d^\alpha \cdot D^\alpha \cdot \left(\frac{d}{\alpha} - H\right)^\alpha \cdot \mu_{\vec{\mathbf{k}}}^{\alpha H - d - \alpha} \cdot \int_{\Delta_{\vec{\mathbf{k}}}} 1 \, dy \\
=& d^\alpha \cdot D^{\alpha+d} \cdot \left(\frac{d}{\alpha} - H\right)^\alpha \cdot \mu_{\vec{\mathbf{k}}}^{\alpha H - d - \alpha}.
\end{aligned}$$

Therefore

$$\begin{aligned}
&\left|\left|X_\varphi^{A,M}(x_{\vec{\mathbf{m}}}) - X_\varphi^A(x_{\vec{\mathbf{m}}})\right|\right|_\alpha^\alpha \\
\leq& 4 \int_{[-D,D)^d} ||y||_\rho^{\alpha H - d} \, dy + 2 \sum_{\vec{\mathbf{k}} \in J_1} I_{\vec{\mathbf{k}}} + 2 \sum_{\vec{\mathbf{k}} \in J_2} I_{\vec{\mathbf{k}}} \\
\leq& 4 \cdot \frac{d \cdot 2^d}{\alpha H} \cdot D^{\alpha H} + 2 \sum_{\vec{\mathbf{k}} \in J_1} d^\alpha \cdot D^{\alpha+d} \cdot \left(\frac{d}{\alpha} - H\right)^\alpha \cdot \mu_{\vec{\mathbf{k}}}^{\alpha H - d - \alpha} \\
&\qquad + 2 \sum_{\vec{\mathbf{k}} \in J_2} d^\alpha \cdot D^{\alpha+d} \cdot \left(\frac{d}{\alpha} - H\right)^\alpha \cdot \mu_{\vec{\mathbf{k}}}^{\alpha H - d - \alpha} \\
=& \frac{d \cdot 2^{d+2}}{\alpha H} \cdot D^{\alpha H} + 2 d^\alpha \cdot D^{\alpha+d} \cdot \left(\frac{d}{\alpha} - H\right)^\alpha \cdot \left(\sum_{\vec{\mathbf{k}} \in J_1} \mu_{\vec{\mathbf{k}}}^{\alpha H - d - \alpha} + \sum_{\vec{\mathbf{k}} \in J_2} \mu_{\vec{\mathbf{k}}}^{\alpha H - d - \alpha} \right)
\end{aligned}$$

It can be shown that $\sum_{\vec{\mathbf{k}} \in J_1} \mu_{\vec{\mathbf{k}}}^{\alpha H - d - \alpha} \leq \sum_{\vec{\mathbf{k}} \in J_2} \mu_{\vec{\mathbf{k}}}^{\alpha H - d - \alpha}$ by dividing the set J_1 into 2^d subsets and replacing these subsets by sets of equal sizes whose sum is J_2, such that the sum of $\mu_{\vec{\mathbf{k}}}^{\alpha H - d - \alpha}$ on each subset of J_1 is less than or equal to the sum on its replacement.

As an example, be $d = 2$ and $\vec{\mathbf{m}} = \binom{m_1}{m_2}$ with $0 \leq m_1, m_2 < M$, and be the values $\mu_{\vec{\mathbf{k}}}$ for $\vec{\mathbf{k}} \in \{-1, 0\}^2$ defined as some positive, finite values, e.g. $\mu_{\vec{\mathbf{k}}} = 1$ for $\vec{\mathbf{k}} \in \{-1, 0\}^2$. Then

$J_1 = \left(\{-M,\ldots,M-1\}^2 - \binom{m_1}{m_2}\right) \backslash \{-1,0\}^2$ and

$$\begin{aligned}
&\sum_{\vec{\mathbf{k}}\in J_1} \mu_{\vec{\mathbf{k}}}^{\alpha H-d-\alpha} + \sum_{\vec{\mathbf{k}}\in\{-1,0\}^2} \mu_{\vec{\mathbf{k}}}^{\alpha H-d-\alpha}\\
=& \sum_{k_1=-M-m_1}^{M-1-m_1} \sum_{k_2=-M-m_2}^{M-1-m_2} \mu_{k_1,k_2}^{\alpha H-d-\alpha}\\
=& \sum_{k_1=-M-m_1}^{-M-1} \sum_{k_2=-M-m_2}^{-M-1} \mu_{k_1,k_2}^{\alpha H-d-\alpha} + \sum_{k_1=-M-m_1}^{-M-1} \sum_{k_2=-M}^{M-1-m_2} \mu_{k_1,k_2}^{\alpha H-d-\alpha}\\
&+ \sum_{k_1=-M}^{M-1-m_1} \sum_{k_2=-M-m_2}^{-M-1} \mu_{k_1,k_2}^{\alpha H-d-\alpha} + \sum_{k_1=-M}^{M-1-m_1} \sum_{k_2=-M}^{M-1-m_2} \mu_{k_1,k_2}^{\alpha H-d-\alpha}\\
\le& \sum_{k_1=M-m_1}^{M-1} \sum_{k_2=M-m_2}^{M-1} \mu_{M,M}^{\alpha H-d-\alpha} + \sum_{k_1=M-m_1}^{M-1} \sum_{k_2=-M}^{M-1-m_2} \mu_{M,k_2}^{\alpha H-d-\alpha}\\
&+ \sum_{k_1=-M}^{M-1-m_1} \sum_{k_2=M-m_2}^{M-1} \mu_{k_1,M}^{\alpha H-d-\alpha} + \sum_{k_1=-M}^{M-1-m_1} \sum_{k_2=-M}^{M-1-m_2} \mu_{k_1,k_2}^{\alpha H-d-\alpha}\\
\le& \sum_{k_1=M-m_1}^{M-1} \sum_{k_2=M-m_2}^{M-1} \mu_{k_1,k_2}^{\alpha H-d-\alpha} + \sum_{k_1=M-m_1}^{M-1} \sum_{k_2=-M}^{M-1-m_2} \mu_{k_1,k_2}^{\alpha H-d-\alpha}\\
&+ \sum_{k_1=-M}^{M-1-m_1} \sum_{k_2=M-m_2}^{M-1} \mu_{k_1,k_2}^{\alpha H-d-\alpha} + \sum_{k_1=-M}^{M-1-m_1} \sum_{k_2=-M}^{M-1-m_2} \mu_{k_1,k_2}^{\alpha H-d-\alpha}\\
=& \sum_{k_1=-M}^{M-1} \sum_{k_2=-M}^{M-1} \mu_{k_1,k_2}^{\alpha H-d-\alpha}\\
=& \sum_{\vec{\mathbf{k}}\in J_2} \mu_{\vec{\mathbf{k}}}^{\alpha H-d-\alpha} + \sum_{\vec{\mathbf{k}}\in\{-1,0\}^2} \mu_{\vec{\mathbf{k}}}^{\alpha H-d-\alpha}
\end{aligned}$$

If $m_1 < 0$ or $m_2 < 0$, or for higher dimensions d, the inequality $\sum_{\vec{\mathbf{k}}\in J_1} \mu_{\vec{\mathbf{k}}}^{\alpha H-d-\alpha} \le \sum_{\vec{\mathbf{k}}\in J_2} \mu_{\vec{\mathbf{k}}}^{\alpha H-d-\alpha}$ can be shown analogously. Therefore

$$\begin{aligned}
&\left|\left|X_\varphi^{A,M}(x_{\vec{\mathbf{m}}}) - X_\varphi^{A}(x_{\vec{\mathbf{m}}})\right|\right|_\alpha^\alpha\\
\le& \frac{d\cdot 2^{d+2}}{\alpha H} \cdot D^{\alpha H} + 2d^\alpha \cdot D^{\alpha+d} \cdot \left(\frac{d}{\alpha} - H\right)^\alpha \cdot 2 \cdot \sum_{\vec{\mathbf{k}}\in J_2} \mu_{\vec{\mathbf{k}}}^{\alpha H-d-\alpha}
\end{aligned}$$

The sum $\sum_{\vec{\mathbf{k}}\in J_2} \mu_{\vec{\mathbf{k}}}^{\alpha H-d-\alpha}$ can be estimated analogously to the estimation of

$\sum_{\vec{\mathbf{k}}\in J} \mu_{\vec{\mathbf{k}}}^{-\alpha H-d-\alpha}$ in the proof of Theorem 3.16: The equation

$$\sum_{\vec{\mathbf{k}}\in J_2} \mu_{\vec{\mathbf{k}}}^{\alpha H-d-\alpha} = D^{\alpha H-d-\alpha}\cdot 2^d \cdot \sum_{\vec{\mathbf{k}}\in J'} ||\vec{\mathbf{k}}||_\rho^{\alpha H-d-\alpha} \tag{4.5}$$

with $J' := \{0,\dots,M-1\}^d\backslash\{0\}^d$ can be shown analogous to (3.15) and (3.16), and similar to the following calculation, we obtain

$$\sum_{\vec{\mathbf{k}}\in J'} ||\vec{\mathbf{k}}||_\rho^{\alpha H-d-\alpha} \le d\cdot\left(2^{d-1}+\frac{(3/2)^{d-1}}{\alpha(1-H)}\left(1-M^{-\alpha(1-H)}\right)\right) \tag{4.6}$$

Therefore

$$\begin{aligned}
&\left|\left|X_\varphi^{A,M}(x_{\vec{\mathbf{m}}})-X_\varphi^A(x_{\vec{\mathbf{m}}})\right|\right|_\alpha^\alpha \\
\le&\frac{d\cdot 2^{d+2}}{\alpha H}\cdot D^{\alpha H}+2d^\alpha\cdot D^{\alpha+d}\cdot\left(\frac{d}{\alpha}-H\right)^\alpha\cdot 2\cdot D^{\alpha H-d-\alpha}\cdot 2^d \\
&\qquad\qquad\cdot d\cdot\left(2^{d-1}+\frac{(3/2)^{d-1}}{\alpha(1-H)}\left(1-M^{-\alpha(1-H)}\right)\right) \\
\le&\frac{d\cdot 2^{d+2}}{\alpha H}\cdot D^{\alpha H}+2^3\cdot d^{\alpha+1}\cdot\left(\frac{d}{\alpha}-H\right)^\alpha\cdot D^{\alpha H}\cdot\left(4^{d-1}+\frac{3^{d-1}}{\alpha(1-H)}\right) \\
=&\widetilde{C}_2\cdot D^{\alpha H}
\end{aligned}$$

with

$$\widetilde{C}_2=\frac{d\cdot 2^{d+2}}{\alpha H}+2^3\cdot d^{\alpha+1}\cdot\left(\frac{d}{\alpha}-H\right)^\alpha\cdot\left(4^{d-1}+\frac{3^{d-1}}{\alpha(1-H)}\right).$$

□

Chapter 5

Approximation algorithms

In this chapter, algorithms for the numerical simulation of OSSRFs in harmonizable and in moving average representation will be developed. First, the generation of α-stable random variables, which is necessary for the simulation of α-stable random fields, is considered, then the simulation of random fields in harmonizable representation, and after this the simulation in the moving average case. For easier notation and better readability, we start by developing an algorithm for the simulation of two-dimensional OSSRF in both cases, and then generalize it to higher dimensions. Finally we consider algorithms for a fast Fourier transform and for a fast convolution, which are used for the simultaneous calculation of a large number of certain sums by the simulation algorithms, in the last two sections of this chapter.

5.1 Simulation of stable random variables

5.1.1 Simulation of an isotropic complex-valued stable random variable

For the approximation of a realization of an OSSRF in *harmonizable* representation, the simulation of isotropic complex-valued α-stable random variables is required.

If $\alpha = 2$, then the random variable X which has to be simulated, is an isotropic gaussian random variable, and can be calculated by $X := G_1 + i \cdot G_2$, for two standard gaussian random variables G_1 and G_2. In MATLAB and Java, the two programming platforms used for this project, functions for the simulation of standard gaussian random variables are already provided by the API (application programming interface) - in MATLAB with the function `randn`, and in Java with the function `Random.nextGaussian()`. The algorithm which is used in Java (in `Random.nextGaussian()`) to generate gaussian distributed random numbers is explicitely stated in the Java API documentation of this function.

However, if $\alpha < 2$, then an isotropic α-stable random variable can be simulated using the following equations from [10]:

(a) Be $A \sim S_{\alpha/2}\left((\cos\frac{\pi\alpha}{4})^{2/\alpha}, 1, 0\right)$ and $G = (G_1, \dots, G_d)$ be a zero mean Gaussian vector, independent of A. Then the random vector $X = (A^{1/2}G_1, \dots, A^{1/2}G_d)$ has a symmetric α-stable distribution (see [10], section 2.5). Note that (under the assumption $\alpha < 2$) the shape parameter of A is $\frac{\alpha}{2} < 1$.

(b) If $X \sim S_\alpha(1, \beta, 0)$, then $\sigma X + \mu \sim S_\alpha(\sigma, \beta, \mu)$ if $\alpha \neq 1$, and $\sigma X + \frac{2}{\pi}\beta\sigma\ln(\sigma) + \mu \sim S_\alpha(\sigma, \beta, \mu)$ if $\alpha = 1$ (see [10], section 1.7, page 43).

(c) A $S_\alpha(1, \beta, 0)$-distributed r.v. can be simulated using the algorithm by Chambers, Mallows and Stuck (see [4]), which (when called with the parameters $\alpha \in (0, 2)$ and $\beta \in [-1, 1]$) simulates a random variable

$$Y \sim \begin{cases} S_\alpha(1, \beta, -\beta\tan(\frac{\pi\alpha}{2})) & \text{if } \alpha \neq 1 \\ S_\alpha(1, \beta, 0) & \text{if } \alpha = 1 \end{cases}$$

Therefore, in order to simulate an α-stable isotropic complex random variable X, the following algorithm may be used:

- If $\alpha = 2$: Simulate $G_1, G_2 \sim \mathcal{N}_{0,1}$ (e.g. in MATLAB with `randn` or in Java with `Random.nextGaussian()`) and set $X := G_1 + i \cdot G_2$
- if $0 < \alpha < 2$:
  (a) Simulate a random variable $Y \sim S_{\alpha'}\left(1, 1, -\tan(\frac{\pi\alpha'}{2})\right)$ with $\alpha' = \frac{\alpha}{2}$, i.e.
  $$Y \sim S_{\alpha/2}\left(1, 1, -\tan\left(\frac{\pi\alpha}{4}\right)\right)$$
  using the method by Chamber, Mallows and Stuck: $Y :=$ `rstab` $\left(\frac{\alpha}{2}, 1\right)$. Note that we assume $\alpha < 2$, therefore $\frac{\alpha}{2} < 1$.
  (b) Calculate
  $$A := \left(Y + \tan\left(\frac{\pi\alpha}{4}\right)\right) \cdot \left(\cos\left(\frac{\pi\alpha}{4}\right)\right)^{2/\alpha}$$
  (Then $Y + \tan(\pi\alpha/4) \sim S_{\alpha/2}(1, 1, 0) \Rightarrow A \sim S_{\alpha/2}\left(\cos(\pi\alpha/4)^{2/\alpha}, 1, 0\right)$)
  (c) Simulate $G_1, G_2 \sim \mathcal{N}_{0,1}$ (as in the case $\alpha = 2$)
  (d) Calculate $X := \sqrt{A} \cdot G_1 + i \cdot \sqrt{A} \cdot G_2$

Then (in both cases) X is a realization of an isotropic complex α-stable random variable (with scale parameter 1). If a random variable with scale parameter $\sigma \neq 1$ should be simulated, then X has to be multiplied with σ, i.e. the requested random number is $\sigma \cdot X$.

5.1.2 Simulation of a real-valued symmetric stable random variable

For the approximation of a realization of an OSSRF in *moving average* representation, the simulation of symmetric α-stable random variables is required. These random variables can be simulated directly with the algorithm by Chambers, Mallows and Stuck (see [4]) by calling it with the shape parameter α and the skewness parameter 0. The obtained random number then has to be multiplied with the required scale parameter σ, if a random variable with scale parameter $\sigma \neq 1$ needs to be simulated.

5.2 Approximation of two-dimensional OSSRF in harmonizable distribution

A 2-dimensional OSSRF in harmonizable representation

$$X_\psi(x) = \operatorname{Re} \int_{\mathbb{R}^2} \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-\frac{q}{\alpha}} W_\alpha(d\xi), \qquad x \in \mathbb{R}^2$$

is approximated by the finite sum

$$X_\psi^{A,B,D}(x) = \operatorname{Re} \sum_{(k,l)\in J} \left(e^{i<x,\xi_{k,l}>} - 1\right) \psi(\xi_{k,l})^{-H-\frac{q}{\alpha}} W_\alpha(\Delta_{k,l})$$

with $M := \frac{A}{D} \in \mathbb{N}$ and $N := \frac{B}{D} \in \mathbb{N}$, $J := \{-M, \dots, M-1\}^2 \backslash \{-N, \dots, N-1\}^2$, $\xi_{k,l} := (k \cdot D, l \cdot D)^T$, and $\Delta_{k,l} := [k \cdot D, (k+1) \cdot D) \times [l \cdot D, (l+1) \cdot D)$ for each $(k,l) \in J$.

This can be transformed as follows:

$$\begin{aligned} X_\psi^{A,B,D}(x) &= \operatorname{Re} \sum_{(k,l)\in J} \left(e^{i<x,\xi_{k,l}>} - 1\right) \psi(\xi_{k,l})^{-H-\frac{q}{\alpha}}\ W_\alpha(\Delta_{k,l}) \\ &= \operatorname{Re} \sum_{(k,l)\in J} \left(e^{i<x,\xi_{k,l}>} \psi(\xi_{k,l})^{-H-\frac{q}{\alpha}} - e^{i\cdot 0} \psi(\xi_{k,l})^{-H-\frac{q}{\alpha}}\right)\ W_\alpha(\Delta_{k,l}) \\ &= \operatorname{Re}\left(V_\psi^{A,B,D}(x)\right) - \operatorname{Re}\left(V_\psi^{A,B,D}(0)\right) \end{aligned}$$

with

$$V_\psi^{A,B,D}(x) = \sum_{(k,l)\in J} e^{i<x,\xi_{k,l}>} \psi(\xi_{k,l})^{-H-\frac{q}{\alpha}}\ W_\alpha(\Delta_{k,l})$$

which can be represented by

$$V_\psi^{A,B,D}(x) = \sum_{-M \le k,l < M} e^{i<x,\xi_{k,l}>} \cdot f_{k,l} \cdot w_{k,l}$$

where

$$f_{k,l} = \begin{cases} 0 & \text{if } (k,l) \in \{-N, \dots, N-1\}^2 \\ \psi(\xi_{k,l})^{-H-\frac{q}{\alpha}} & \text{else} \end{cases}$$

is a function of the indices k and l, and where (for each pair of indices (k,l)), $w_{k,l} = W_\alpha(\Delta_{k,l})$ is an α-stable, isotropic complex-valued random variable with scale parameter $(A/M)^{2/\alpha}$ (because $W_\alpha(d\xi)$ has Lebesgue control measure, and $\Delta_{k,l}$ has Lebesgue measure $(A/M)^2$).

Usually, we are not interested in simulating the value of $X_\psi^{A,B,D}(x)$ for only one $x \in \mathbb{R}^2$, or only a few such arguments, but for a large set of $x \in \mathbb{R}^2$. In order to perform the necessary computations efficiently, it is useful to simulate the values of the OSSRF on a regular grid $(x_{m,n})$ with $2M \times 2M$ points (e.g. the number of approximated values of the OSSRF is the same as the number of simulated α-stable random variables $W_\alpha(\Delta_{k,l})$ which are used for their calculation). Thus, the approximated random field $X_\psi^{A,B,D}$ (and for that purpose, the values of $V_\psi^{A,B,D}$) are calculated on $x_{m,n} = (C \cdot m, C \cdot n)^T$, with $-M \le m, n < M$ and a certain constant $C > 0$, so that (for each pair $(m,n) \in \{-M, \dots, M\}$) we have to calculate

$$\begin{aligned} V_\psi^{A,B,D}(x_{m,n}) &= \sum_{-M \le k,l < M} e^{i<x_{m,n},\xi_{k,l}>} \cdot f_{k,l} \cdot w_{k,l} \\ &= \sum_{-M \le k,l < M} e^{i<\binom{C\cdot m}{C\cdot n},\binom{D\cdot k}{D\cdot l}>} \cdot (f_{k,l} \cdot w_{k,l}) \\ &= \sum_{-M \le k,l < M} e^{i\cdot CD\cdot(km+ln)} \cdot (f_{k,l} \cdot w_{k,l}) \end{aligned}$$

If the constant C is chosen as $C = \frac{\pi}{A}$, then $C \cdot D = \frac{\pi}{A} \cdot \frac{A}{M} = \frac{\pi}{M}$, so that $i \cdot CD \cdot (km+ln) = 2\pi i \cdot \frac{km+ln}{2M}$ which means that $(V_\psi^{A,B,D}(x_{m,n}))$ is the two-dimensional discrete Fourier transform (DFT) of $(f_{k,l} \cdot w_{k,l})$. In this case, a fast Fourier transform (FFT) algorithm can be used for this calculation. Therefore this choice for the constant C is used in the algorithm, so that

$$V_\psi^{A,B,D}(x_{m,n}) = \sum_{-M \le k,l < M} e^{2\pi i \cdot \frac{km+ln}{2M}} \cdot (f_{k,l} \cdot w_{k,l}).$$

The considerations of this section can be summarized in the following algorithm for the simulation of two-dimensional OSSRF in harmonizable representation:

(a) Simulation of $(2M) \times (2M)$ i.i.d. complex-valued isotropic α-stable random variables (r.v.) $w_{k,l} = W_\alpha(\Delta_{k,l})$ (with scale parameter $\sigma = D^{2/\alpha}$),
for $k, l \in \{-M, \dots, M-1\}^2$ with the algorithm from subsection 5.1.1.

(b) Calculation of

$$f_{k,l} = \begin{cases} 0 & \text{if } (k,l) \in \{-N,\dots,N-1\}^2 \\ \psi(\xi_{k,l})^{-H-\frac{q}{\alpha}} & \text{else} \end{cases}$$

and of the products $f_{k,l} \cdot w_{k,l}$ for $(k,l) \in \{-M,\dots,M-1\}^2$.

(c) Calculation of $V_\psi^{A,B,D}(x_{m,n}) = \sum_{-M \le k,l < M} e^{2\pi i \cdot \frac{km+ln}{2M}} \cdot (f_{k,l} \cdot w_{k,l}) = DFT((f_{k,l} \cdot w_{k,l})_{k,l})$, i.e. the two-dimensional discrete Fourier transform of the products $(f_{k,l} \cdot w_{k,l})$, using a fast Fourier transform algorithm.

(d) Calculation of $X_\psi^{A,B,D}(x_{m,n}) = \operatorname{Re}\left(V_\psi^{A,B,D}(x_{m,n})\right) - \operatorname{Re}\left(V_\psi^{A,B,D}(0)\right)$

5.3 Approximation of d-dimensional OSSRF in harmonizable distribution

The algorithm for the simulation of two-dimensional OSSRF in harmonizable representation can be easily generalized to an algorithm for the simulation of d-dimensional OSSRF (with $d \geq 2$, e.g. for three-dimensional OSSRF). Like in the two-dimensional case, the OSSRF is simulated on the points $x_{\vec{\mathbf{m}}} := \frac{\pi}{A} \cdot \vec{\mathbf{m}}$ for $\vec{\mathbf{m}} \in \{-M,\dots,M-1\}^d$:

(a) Simulation of $(2M)^d$ i.i.d. complex-valued isotropic α-stable random variables (with scale parameter $\sigma = D^{d/\alpha}$) $w_{\vec{\mathbf{k}}} = W_\alpha(\Delta_{\vec{\mathbf{k}}})$, for $\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d$ with the algorithm from subsection 5.1.1.

(b) Calculation of $f_{\vec{\mathbf{k}}} = \begin{cases} \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}} \text{ for } \xi_{\vec{\mathbf{k}}} = \vec{\mathbf{k}} \cdot D & , \quad \vec{\mathbf{k}} \in J \\ 0 & , \quad \text{else.} \end{cases}$

and of the products $g_{\vec{\mathbf{k}}} := f_{\vec{\mathbf{k}}} \cdot w_{\vec{\mathbf{k}}}$ for $\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d$.

(c) Calculation of $V_\psi^{A,B,D}(x_{\vec{\mathbf{m}}}) = \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} e^{\frac{2\pi i}{2M} < \vec{\mathbf{m}}, \vec{\mathbf{k}} >} g_{\vec{\mathbf{k}}} = DFT(g_{\vec{\mathbf{k}}})$ using a fast Fourier transform algorithm.

(d) Calculation of
$X_\psi^{A,B,D}(x_{\vec{\mathbf{m}}}) = \operatorname{Re}(V_\psi^{A,B,D}(x_{\vec{\mathbf{m}}})) - \operatorname{Re}(V_\psi^{A,B,D}(0))$.

5.4 Approximation of two-dimensional OSSRF in moving average distribution

A 2-dimensional OSSRF in moving average representation

$$X_\varphi(x) = \int_{\mathbb{R}^2} (\varphi(x-y)^{H-q/\alpha} - \varphi(-y)^{H-q/\alpha}) Z_\alpha(dy), \qquad x \in \mathbb{R}^2$$

is approximated by the finite sum

$$X_\varphi^{A,M}(x) = \sum_{k,l=-M}^{M-1} \left(\tilde{\varphi}(x - y_{k,l}) - \tilde{\varphi}(-y_{k,l})\right) Z_\alpha(\Delta_{k,l})$$

with parameters $A > 0$ and $M \in \mathbb{N}$, and with $D := \frac{A}{M}$, $y_{k,l} := (k \cdot D, l \cdot D)^T = \binom{k}{l} \cdot D$, $\Delta_{k,l} := [k \cdot D, (k+1) \cdot D) \times [l \cdot D, (l+1) \cdot D)$ and

$$\tilde{\varphi}(z) := \begin{cases} \varphi(z)^{H-q/\alpha}, & z \notin [-D, D)^2 \\ 0, & z \in [-D, D)^2 \end{cases}$$

(see (4.2) and (4.3)).

This can be transformed as follows:

$$\begin{aligned} X_\varphi^{A,M}(x) &= \sum_{k,l=-M}^{M-1} \left(\tilde{\varphi}(x - y_{k,l}) - \tilde{\varphi}(-y_{k,l})\right)\ Z_\alpha(\Delta_{k,l}) \\ &= \sum_{k,l=-M}^{M-1} \tilde{\varphi}(x - y_{k,l})\ Z_\alpha(\Delta_{k,l}) - \sum_{k,l=-M}^{M-1} \tilde{\varphi}(-y_{k,l})\ Z_\alpha(\Delta_{k,l}) \\ &= V_{\tilde{\varphi}}^{A,M}(x) - V_{\varphi}^{A,M}(0) \end{aligned}$$

with

$$V_{\tilde{\varphi}}^{A,M}(x) = \sum_{k,l=-M}^{M-1} \tilde{\varphi}(x - y_{k,l}) Z_\alpha(\Delta_{k,l}).$$

In order to obtain a useful and efficient algorithm for the simulation of a whole random field $X_\varphi^{A,M}(x)$, the values of $V_\varphi^{A,M}$ are evaluated not just on some arbitrary point $x \in \mathbb{R}^2$, but on the same grid of points, on which the function φ is evalated during the simulation. This means, the values of $V_{\tilde{\varphi}}^{A,M}(x_{m,n})$ (and, using these results, the values of $X_\varphi^{A,M}(x_{m,n})$) are simulated for $x_{m,n} := y_{m,n} = D \cdot \binom{m}{n}$ for all $m, n \in \{-M, \dots, M-1\}$. As we will see, this choice of evaluation points enables an efficient implementation of the simulation of $2M \cdot 2M$ values of the OSSRF at once.

With this choice of evaluation points, $V_{\tilde{\varphi}}^{A,M}$ has the form

$$\begin{aligned} V_{\tilde{\varphi}}^{A,M}(x_{m,n}) &= \sum_{k,l=-M}^{M-1} \tilde{\varphi}(x_{m,n} - y_{k,l})\ Z_\alpha(\Delta_{k,l}) \\ &= \sum_{k,l=-M}^{M-1} \tilde{\varphi}\left(D \cdot \begin{pmatrix} m \\ n \end{pmatrix} - D \cdot \begin{pmatrix} k \\ l \end{pmatrix}\right)\ Z_\alpha(\Delta_{k,l}) \\ &= \sum_{k,l=-M}^{M-1} \tilde{\varphi}\left(D \cdot \begin{pmatrix} m-k \\ n-l \end{pmatrix}\right)\ Z_\alpha(\Delta_{k,l}) \\ &= \sum_{k,l=-M}^{M-1} f_{m-k,n-l} \cdot z_{k,l} \end{aligned}$$

with $f_{r,s} = \tilde{\varphi}(D \cdot \binom{r}{s})$ for each $k, l \in \{-2M+1, \ldots, 2M-1\}$, and where $z_{k,l} := Z_\alpha(\Delta_{k,l})$ is a realization of a symmetric α-stable random variable with scale parameter $D^{2/\alpha}$, because the random measure Z has Lebesgue control measure, and $\Delta_{k,l}$ has the Lebesgue measure D^2. In other words: $V_{\varphi}^{A,M}$ is a convolution of the arrays $(f_{r,s})$ (the function values of $\tilde{\varphi}$) and $(z_{k,l})$ (the simulated values of a symmetric α-stable random variable). This convolution can be calculated very efficiently using the fast Fourier transform and the inverse fast Fourier transform (see section 5.7).

Therefore, the random field $X_{\varphi}^{A,M}(x)$ can be simulated by the following algorithm:

(a) Simulation of $(2M)^2$ i.i.d. symmetric α-stable random variables $z_{k,l} = Z_\alpha(\Delta_{k,l})$, $-M \leq k, l \leq M-1$, with parameters $\mu = 0, \beta = 0, \sigma = D^{2/\alpha}$ (see subsection 5.1.2).

(b) Calculation of

$$f_{k,l} := \tilde{\varphi}(y_{k,l}) = \begin{cases} \varphi(y_{k,l})^{H-q/\alpha}, & (k,l) \notin \{-1,0\}^2 \\ 0, & (k,l) \in \{-1,0\}^2 \end{cases}$$

for $y_{k,l} = \binom{k \cdot D}{l \cdot D}, \quad -2M \leq k, l \leq 2M-1.$

(c) Calculation of

$$V_{\tilde{\varphi}}^{A,M}(x_{m,n}) := \sum_{k,l=-M}^{M-1} \tilde{\varphi}(y_{m-k,n-l}) \cdot Z_\alpha(\Delta_{k,l}) = \sum_{k,l=-M}^{M-1} f_{m-k,n-l} \cdot z_{k,l}$$

for $m, n \in \{-M, \ldots, M-1\}$ (with the algorithm presented in section 5.7).

(d) Calculation of
$$X_{\varphi}^{A,M}(x_{m,n}) = V_{\tilde{\varphi}}^{A,M}(x_{m,n}) - V_{\tilde{\varphi}}^{A,M}(0)$$
for $m, n \in \{-M, \ldots, M-1\}$.

5.5 Approximation of d-dimensional OSSRF in moving average representation

In the moving-average case, the algorithm for the simulation of two-dimensional OSSRF can also be generalized straight-forward to the simulation of OSSRF in higher dimension (analogous to the generalization to higher dimensions in the harmonizable representation): Be $d \geq 2$, then an OSSRF in moving average representation can be approximated (in the points $x_{\vec{\mathbf{m}}} = \vec{\mathbf{m}} \cdot D$ for $\vec{\mathbf{m}} \in \{-M, \ldots, M-1\}^d$) with the following procedure:

(a) Simulation of $(2M)^d$ i.i.d. symmetric α-stable random variables $z_{\vec{\mathbf{k}}} = Z_\alpha(\Delta_{\vec{\mathbf{k}}})$, $\vec{\mathbf{k}} \in \{-M, \ldots, M-1\}^d$, with parameters $\mu = 0, \beta = 0, \sigma = D^{d/\alpha}$ (see subsection 5.1.2).

(b) Calculation of
$$f_{\vec{\mathbf{k}}} = \tilde{\varphi}(y_{\vec{\mathbf{k}}}) = \begin{cases} \varphi(y_{\vec{\mathbf{k}}})^{H-q/\alpha}, & \vec{\mathbf{k}} \notin \{-1,0\}^d \\ 0, & \vec{\mathbf{k}} \in \{-1,0\}^d \end{cases}$$
for $y_{\vec{\mathbf{k}}} = \vec{\mathbf{k}} \cdot D, \quad \vec{\mathbf{k}} \in \{-2M, \ldots, 2M-1\}^d$.

(c) Calculation of
$$V_{\tilde{\varphi}}^{A,M}(x_{\vec{\mathbf{m}}}) = \sum_{\vec{\mathbf{k}} \in \{-M,\ldots,M-1\}^d} \tilde{\varphi}(y_{\vec{\mathbf{m}}-\vec{\mathbf{k}}}) \cdot Z_\alpha(\Delta_{\vec{\mathbf{k}}}) = \sum_{\vec{\mathbf{k}} \in \{-M,\ldots,M-1\}^d} f_{\vec{\mathbf{m}}-\vec{\mathbf{k}}} \cdot z_{\vec{\mathbf{k}}}$$
for $\vec{\mathbf{m}} \in \{-M, \ldots, M-1\}^d$ (see section 5.7).

(d) Calculation of
$$X_{\varphi}^{A,M}(x_{\vec{\mathbf{m}}}) = V_{\tilde{\varphi}}^{A,M}(x_{\vec{\mathbf{m}}}) - V_{\tilde{\varphi}}^{A,M}(0)$$
for $\vec{\mathbf{m}} \in \{-M, \ldots, M-1\}^d$.

5.6 The fast Fourier transform

The methods for the simulation of OSSRF, which have been presented in this chapter, use d-dimensional discrete Fourier transforms, i.e. the calculation of sums of the form

$$y_{\vec{\mathbf{m}}} = \sum_{\vec{\mathbf{k}} \in \{0,\dots,N-1\}^d} x_{\vec{\mathbf{k}}} \cdot e^{2\pi i \frac{<\vec{\mathbf{m}},\vec{\mathbf{k}}>}{N}}, \quad \vec{\mathbf{m}} \in \{0,\dots,N-1\}$$

for a number $N \in \mathbb{N}$. If the transform was implemented directly from this sum formula (i.e. calculating N^d sums of N^d summands each), the required time for the calculation would be asymptotically proportional to N^{2d} (respective P^2, where $P = N^d$ is the number of processed data points), i.e. the calculation would have a complexity of $O(N^{2d})$ (resp. $O(P^2)$). However, with a more sophisticated algorithm (a "fast Fourier transform"), the same sums can be calculated with a complexity of $O(N^d \cdot \log(N))$ (resp. $O(P \cdot \log(P))$), thus performing the calculations much faster. In the following paragraphs, some basic principles of a "fast" Fourier transform, which reduce the time complexity (and thus the needed time for the calculation), are presented.

Multi-dimensional discrete Fourier transforms

A discrete Fourier transform of a d-dimensional data array (with $d \geq 2$) can be partitioned into the calculation of one-dimensional Fourier transforms. More precisely, the Fourier transform of an d-dimensional array of size N^d (i.e. length N in each dimension) can be calculated by performing N^{d-1} one-dimensional DFTs (each of size N) in the first dimension, followed by N^{d-1} one-dimensional DFTs in the second dimension, etc. Combined, $d \times N^{d-1}$ one-dimensional DFTs (with a complexity of $O(N^2)$ each) have to be calculated, so that the complexity has been reduced from $O(N^{2d})$ to $O(N^{d-1} \cdot N^2) = O(N^{d+1})$.

This sectioning of the DFT is shown more detailed in the case of a two-dimensional transform: Be the array $y_{m,n}$ the result of a DFT of $x_{k,l}$ (with $k, l, m, n \in \{0, \dots, N-1\}$), i.e.

$$y_{m,n} = \sum_{0 \leq k,l < N} x_{k,l} \cdot e^{2\pi i \cdot \frac{km+ln}{N}}.$$

This can be transformed to:

$$\begin{aligned} y_{m,n} &= \sum_{0 \leq k,l < N} x_{k,l} \cdot e^{2\pi i \cdot \frac{km+ln}{N}} = \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} x_{k,l} \cdot e^{2\pi i \cdot \frac{km}{N}} \cdot e^{2\pi i \cdot \frac{ln}{N}} \\ &= \sum_{l=0}^{N-1} e^{2\pi i \cdot \frac{ln}{N}} \cdot \underbrace{\sum_{k=0}^{N-1} x_{k,l} \cdot e^{2\pi i \cdot \frac{km}{N}}}_{=:z_{m,l}} \end{aligned}$$

i.e. $y_{m,n} = \sum_{l=0}^{N-1} z_{m,l} \cdot e^{2\pi i \cdot \frac{ln}{N}}$ with $z_{m,l} = \sum_{k=0}^{N-1} x_{k,l} \cdot e^{2\pi i \cdot \frac{km}{N}}$. Therefore, the two-dimensional DFT can be calculated by calculating first the $z_{m,l}$ from the $x_{k,l}$ for each l (DFTs of the columns) and then calculating the $y_{m,n}$ from the $z_{m,l}$ for each m (DFTs of the rows). Thus, the algorithm of the two-dimensional DFT has been changed from the calculation of N^2 sums with N^2 summands each to the calculation of $2N$ one-dimensional DFTs, i.e. $2N \cdot N$ sums with N sums each. Therefore, the number of evaluations of the exp-function and the number of multiplications have been reduced from N^4 to $2N^3$. The same method can be used to calculate a d-dimensional DFT using one-dimensional DFTs also for higher dimensions ($d > 2$). This technique is used in the implementation of FFTs in JAVA which is part of the JAVA implementation of the simulation algorithms for OSSRF described in this chapter, and it is also used for the calculation of multi-dimensional FFTs in MATLAB, as it is described in [9] (a German MATLAB manual) in section 8.6, and in [7] in the descriptions of the functions `fft2` and `fftn`.

One-dimensional Fourier transform

Be $y_m = \sum_{k=0}^{N-1} x_k \cdot e^{2\pi i \cdot \frac{km}{N}}$. If this sum is calculated for each $m \in \{0, \ldots, N-1\}$ separately, then N^2 evaluations of the complex exp-function and N^2 complex multiplications need to be performed. With a more efficient algorithm, the fast Fourier transform, these sums can be calculated much faster. It is most efficient if N is a power of 2, then its complexity is in $O(N \times \log(N))$ (compared to $O(N^2)$ for the simple DFT algorithm). An algorithm for the fast calculation of a Fourier transform was published by J. Cooley and J. Tukey in 1965 (see [3]).

If N is an even number, the calculation of this sum can be divided into the calculation of two smaller sums (one sum for all even indices of the original sum, and one sum of the summands with odd indices), and their combination, as follows (be $N' = \frac{N}{2}$):

$$\begin{aligned}
y_m &= \sum_{k=0}^{N-1} x_k \cdot e^{2\pi i \cdot \frac{km}{N}} \\
&= \sum_{k'=0}^{N'-1} x_{2k'} \cdot e^{2\pi i \cdot \frac{2k'm}{2N'}} + \sum_{k''=0}^{N'-1} x_{2k''+1} \cdot e^{2\pi i \cdot \frac{(2k''+1)m}{2N'}} \\
&= \underbrace{\sum_{k'=0}^{N'-1} x_{2k'} \cdot e^{2\pi i \cdot \frac{k'm}{N'}}}_{=:y'_m} + e^{2\pi i \cdot \frac{m}{2N'}} \cdot \underbrace{\sum_{k''=0}^{N'-1} x_{2k''+1} \cdot e^{2\pi i \cdot \frac{k''m}{N'}}}_{=:y''_m} .
\end{aligned}$$

The sums y'_m and y''_m, which again have the form of a discrete Fourier transform, have $N' = N/2$ summands each (opposed to the N summands of the original sum), and they only have to be calculated for $m < N'$, because their values are the same for m and for $m + N'$:

$$\begin{aligned} y'_{m+N'} &= \sum_{k'=0}^{N'-1} x_{2k'} \cdot e^{2\pi i \cdot \frac{k'(m+N')}{N'}} = \sum_{k'=0}^{N'-1} x_{2k'} \cdot e^{2\pi i \cdot \frac{k'm}{N'}} \cdot e^{2\pi i \cdot \frac{k'N'}{N'}} \\ &= \sum_{k'=0}^{N'-1} x_{2k'} \cdot e^{2\pi i \cdot \frac{k'm}{N'}} \cdot 1 = y'_m \end{aligned}$$

(and analogously for $y''_{m+N'}$). After calculating these sums for $0 \le m < N'$ (which can be done recursively with this method, if N' is also even, or by a direct summation with only $N'^2 = N^2/4$ summands for each of the two sums), the values of y_m and $y_{m+N'}$ are calculated (in $O(N)$ time) by $y_m = y'_m + e^{2\pi i \cdot \frac{m}{N}} \cdot y''_m$ and $y_{m+N'} = y'_m + e^{2\pi i \cdot \frac{m+N'}{N}} \cdot y''_m = y'_m - e^{2\pi i \cdot \frac{m}{N}} \cdot y''_m$ (for $0 \le m < N'$).

If N is a power of 2, then the DFT can be calculated by dividing the summation into two parts of half length recursively, until only DFTs of length 1 have to be processed.

This partitioning method can be applied not only if N is divisible by 2, but analogously also if N is divisible by 3, 5, or other numbers. Therefore, the DFT can be calculated very efficiently if N can be factorized into small prime factors. However, the best performance is achieved for powers of 2.

The effect of an efficient implementation of the discrete Fourier transform is demonstrated in table 5.1, which compares the runtimes of a direct implementation of the DFT without optimizations (t_{DFT2}), of an implementation which uses calls to a simple implementation of the one-dimensional DFT (t_{DFT}), of an implementation which uses calls to a one-dimensional fast Fourier transform (t_{FFT}) - all of them implemented in JAVA - and of the `fft2` function in MATLAB (t_{MATLAB}) for a two-dimensional DFT. The values are times in seconds, unless stated otherwise, and were measured for calculations on a single core of a 2.4 Ghz Intel Core 2 Duo processor. Times in brackets have not been measured, but extrapolated from the measured smaller values. The table shows that using a fast algorithm reduces the processing time by a large factor, thereby enabling the calculation of DFTs with several million data points within a few seconds instead of several hours or even days. For example, the calculation of a DFT with one million elements can be calculated by the used implementation of the FFT within one second, while it would have taken about 7.5 hours with a simple, direct implementation. The corresponding function in MATLAB took only 0.14 seconds to finish the calculation.

N	N^2	t_{DFT2}	t_{DFT}	t_{FFT}	t_{MATLAB}
40	1600	0.069	0.0032	0.00055	0.00015
60	3600	0.350	0.011	0.0018	0.00039
80	6400	1.08	0.026	0.0024	0.00066
100	10000	2.71	0.051	0.0060	0.0015
200	40000	[43]	0.42	0.027	0.0048
300	90000	[$\approx$ 3.7 min.]	1.5	0.072	0.011
500	250000	[$\approx$ 28 min.]	7.4	0.23	0.034
1000	1000000	[$\approx$ 7.5 h.]	[$\approx$ 1.0 min.]	0.95	0.14
2000	4000000	[$\approx$ 5.0 days]	[$\approx$ 8.0 min.]	4.0	0.60
3000	9000000	[$\approx$ 25 days]	[$\approx$ 27.0 min.]	10.5	1.38
64	4096	0.44	0.013	0.0009	0.0007
128	16384	7.3	0.11	0.0042	0.0020
256	65536	[116]	0.92	0.019	0.0071
512	262144	[$\approx$ 31 min.]	7.95	0.089	0.041
1024	1048576	[$\approx$ 8.2 h.]	[64]	0.40	0.17
2048	4194304	[$\approx$ 5.5 days]	[$\approx$ 8.6 min.]	1.8	0.71
4096	16777216	[$\approx$ 88 days]	[$\approx$ 69 min.]	8.5	2.94

Table 5.1: Times for the calculation of a two-dimensional discrete Fourier transform on $N \cdot N$ data points (times without a stated unit are in seconds). t_{DFT2} is the time for a direct implementation of the two-dimensional transform, t_{DFT} the time for an implementation which splits the transform into one-dimensional DFTs, t_{FFT} the time for an implementation using one-dimensional fast Fourier transforms (all three being implemented in JAVA), and t_{MATLAB} the computing time for the corresponding FFT function in MATLAB. Times in brackets have not been measured, but extrapolated.

5.7 The fast convolution

During the simulation of a moving-average OSSRF, sums of the form

$$v_{\vec{\mathbf{m}}} := V_{\tilde{\varphi}}^{A,M}(x_{\vec{\mathbf{m}}}) = \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} f_{\vec{\mathbf{m}}-\vec{\mathbf{k}}} \cdot z_{\vec{\mathbf{k}}}$$

for $\vec{\mathbf{m}} \in \{-M,\dots,M-1\}^d$ have to be calculated (in step (c) of the algorithms in sections 5.4 and 5.5). If these sums are simply calculated one after another by explicitly calculating each single summand and adding these summands, $(2M)^d$ sums of $(2M)^d$ summands each have to be calculated, i.e. the complexity of this calculation is in $O(M^{2d})$. As we will show in this section, this convolution can also be calculated by performing two fast Fourier transforms and one inverse FFT of arrays of size $(4M)^d$. Because these

FFTs and the inverse FFT have a complexity of $O(M^d \cdot \log(M))$ (as it has been shown in the previous section), the computing time for the convolution is reduced to a small fraction for sufficiently large values of M.

In this section, $(\mathrm{FFT}(x_{\vec{\mathbf{k}}}))$ be the notation of the array which is the result of a fast Fourier transform applied to the array $x_{\vec{\mathbf{k}}}$, and $(\mathrm{IFFT}(v_{\vec{\mathbf{n}}}))$ be the notation for the result of an inverse fast Fourier transform on $v_{\vec{\mathbf{n}}}$. Additionally be

$$\mathcal{M}_4 := \{0, \ldots, 4M-1\}^d$$

and

$$\tilde{z}_{\vec{\mathbf{k}}} := \begin{cases} z_{\vec{\mathbf{k}}}, & \vec{\mathbf{k}} \in \{-M, \ldots, M-1\}^d \\ 0, & \text{else} \end{cases} .$$

Be $\hat{f}$ and $\hat{z}$ the fast Fourier transforms of the shifted arrays $(f_{\vec{\mathbf{l}}})_{\vec{\mathbf{l}} \in \{-2M, \ldots, 2M-1\}}$ and $(\tilde{z}_{\vec{\mathbf{k}}})_{\vec{\mathbf{k}} \in \{-2M, \ldots, 2M-1\}}$, i.e.

$$\hat{f}_{\vec{\mathbf{n}}} := (FFT(f_{\vec{\mathbf{l}}-\{2M\}^d}))_{\vec{\mathbf{n}}} = \sum_{\vec{\mathbf{l}} \in \mathcal{M}_4} e^{2\pi i \frac{<\vec{\mathbf{n}}, \vec{\mathbf{l}}>}{4M}} f_{\vec{\mathbf{l}}-\{2M\}^d}$$

and

$$\begin{aligned} \hat{z}_{\vec{\mathbf{n}}} := (FFT(\tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d}))_{\vec{\mathbf{n}}} &= \sum_{\vec{\mathbf{k}} \in \mathcal{M}_4} e^{2\pi i \frac{<\vec{\mathbf{n}}, \vec{\mathbf{k}}>}{4M}} \tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d} \\ &= \sum_{\vec{\mathbf{k}} \in \{M, \ldots, 3M-1\}^d} e^{2\pi i \frac{<\vec{\mathbf{n}}, \vec{\mathbf{k}}>}{4M}} z_{\vec{\mathbf{k}}-\{2M\}^d} \end{aligned}$$

for $\vec{\mathbf{n}} \in \mathcal{M}_4$. Then the elementwise product of these arrays is (again, for each $\vec{\mathbf{n}} \in \mathcal{M}_4$)

$$\begin{aligned} \hat{v}_{\vec{\mathbf{n}}} &:= \hat{f}_{\vec{\mathbf{n}}} \cdot \hat{z}_{\vec{\mathbf{n}}} = (FFT(f_{\vec{\mathbf{l}}-\{2M\}^d}))_{\vec{\mathbf{n}}} \cdot (FFT(\tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d}))_{\vec{\mathbf{n}}} \\ &= \left(\sum_{\vec{\mathbf{l}} \in \mathcal{M}_4} e^{2\pi i \frac{<\vec{\mathbf{n}}, \vec{\mathbf{l}}>}{4M}} f_{\vec{\mathbf{l}}-\{2M\}^d} \right) \cdot \left(\sum_{\vec{\mathbf{k}} \in \mathcal{M}_4} e^{2\pi i \frac{<\vec{\mathbf{n}}, \vec{\mathbf{k}}>}{4M}} \tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d} \right) \\ &= \sum_{\vec{\mathbf{l}} \in \mathcal{M}_4} \sum_{\vec{\mathbf{k}} \in \mathcal{M}_4} e^{2\pi i \frac{<\vec{\mathbf{n}}, \vec{\mathbf{l}}> + <\vec{\mathbf{n}}, \vec{\mathbf{k}}>}{4M}} \cdot f_{\vec{\mathbf{l}}-\{2M\}^d} \cdot \tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d} \\ &= \sum_{\vec{\mathbf{l}} \in \mathcal{M}_4} \sum_{\vec{\mathbf{k}} \in \mathcal{M}_4} e^{2\pi i \frac{n_1(l_1+k_1)+\ldots+n_d(l_d+k_d)}{4M}} \cdot f_{\vec{\mathbf{l}}-\{2M\}^d} \cdot \tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d} \\ &= \sum_{\vec{\mathbf{p}} \in \mathcal{M}_4} \sum_{\substack{\vec{\mathbf{l}}, \vec{\mathbf{k}} \in \mathcal{M}_4 \\ l_j+k_j \equiv p_j \pmod{4M}, \\ 1 \leq j \leq d}} e^{2\pi i \frac{n_1 p_1 + \ldots + n_d p_d}{4M}} \cdot f_{\vec{\mathbf{l}}-\{2M\}^d} \cdot \tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d} \end{aligned}$$

$$= \sum_{\vec{\mathbf{p}}\in\mathcal{M}_4} \sum_{\vec{\mathbf{k}}\in\mathcal{M}_4} \sum_{\substack{\vec{\mathbf{l}}\in\mathcal{M}_4 \\ l_j \equiv p_j - k_j \ (\mathrm{mod}\ 4M), \\ 1\le j\le d}} e^{2\pi i \frac{<\vec{\mathbf{n}},\vec{\mathbf{p}}>}{4M}} \cdot f_{\vec{\mathbf{l}}-\{2M\}^d} \cdot \tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d}$$

$$= \sum_{\vec{\mathbf{p}}\in\mathcal{M}_4} e^{2\pi i \frac{<\vec{\mathbf{n}},\vec{\mathbf{p}}>}{4M}} \sum_{\vec{\mathbf{k}}\in\mathcal{M}_4} \tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d} \cdot f_{t(\vec{\mathbf{p}}-\vec{\mathbf{k}})-\{2M\}^d}$$

where $t(\vec{\mathbf{l}}) = t((l_1,\dots,l_d)^T) = (t(l_1),\dots,t(l_d))^T$, with

$$t(l_j) := l_j \mod 4M = \begin{cases} l_j, & 0 \le l_j \le 4M-1 \\ l_j + 4M, & -4M \le l_j \le -1 \end{cases}$$

The inverse Fourier transform of $\hat{v}_{\vec{\mathbf{n}}}$,
i.e. $IFFT(\hat{v}_{\vec{\mathbf{n}}}) = IFFT((FFT(f_{\vec{\mathbf{l}}-\{2M\}^d}))_{\vec{\mathbf{n}}} \cdot (FFT(\tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d}))_{\vec{\mathbf{n}}})$, is given by

$$\begin{aligned}
\tilde{v}_{\vec{\mathbf{m}}-\{2M\}^d} &:= (IFFT(\hat{v}_{\vec{\mathbf{n}}}))_{\vec{\mathbf{m}}} \\
&= \frac{1}{(4M)^d} \sum_{\vec{\mathbf{n}}\in\mathcal{M}_4} e^{-2\pi i \frac{<\vec{\mathbf{m}},\vec{\mathbf{n}}>}{4M}} \hat{v}_{\vec{\mathbf{n}}} \\
&= \frac{1}{(4M)^d} \sum_{\vec{\mathbf{n}}\in\mathcal{M}_4} e^{-2\pi i \frac{<\vec{\mathbf{m}},\vec{\mathbf{n}}>}{4M}} \sum_{\vec{\mathbf{p}}\in\mathcal{M}_4} e^{2\pi i \frac{<\vec{\mathbf{n}},\vec{\mathbf{p}}>}{4M}} \sum_{\vec{\mathbf{k}}\in\mathcal{M}_4} \tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d} \cdot f_{t(\vec{\mathbf{p}}-\vec{\mathbf{k}})-\{2M\}^d} \\
&= \frac{1}{(4M)^d} \sum_{\vec{\mathbf{n}}\in\mathcal{M}_4} \sum_{\vec{\mathbf{p}}\in\mathcal{M}_4} e^{2\pi i \frac{<\vec{\mathbf{n}},\vec{\mathbf{p}}-\vec{\mathbf{m}}>}{4M}} \sum_{\vec{\mathbf{k}}\in\mathcal{M}_4} \tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d} \cdot f_{t(\vec{\mathbf{p}}-\vec{\mathbf{k}})-\{2M\}^d} \\
&= \frac{1}{(4M)^d} \sum_{\vec{\mathbf{p}}\in\mathcal{M}_4} \left(\sum_{\vec{\mathbf{k}}\in\mathcal{M}_4} \tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d} \cdot f_{t(\vec{\mathbf{p}}-\vec{\mathbf{k}})-\{2M\}^d} \right) \cdot \left(\sum_{\vec{\mathbf{n}}\in\mathcal{M}_4} e^{2\pi i \frac{<\vec{\mathbf{n}},\vec{\mathbf{p}}-\vec{\mathbf{m}}>}{4M}} \right) \\
&= \frac{1}{(4M)^d} \sum_{\vec{\mathbf{p}}\in\mathcal{M}_4} \left(\sum_{\vec{\mathbf{k}}\in\mathcal{M}_4} \tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d} \cdot f_{t(\vec{\mathbf{p}}-\vec{\mathbf{k}})-\{2M\}^d} \right) \cdot (4M)^d \cdot 1_{\{\vec{\mathbf{m}}\}}(\vec{\mathbf{p}}) \\
&= \sum_{\vec{\mathbf{k}}\in\mathcal{M}_4} \tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d} \cdot f_{t(\vec{\mathbf{m}}-\vec{\mathbf{k}})-\{2M\}^d} \\
&= \sum_{\vec{\mathbf{k}}\in\{M,\dots,3M-1\}^d} z_{\vec{\mathbf{k}}-\{2M\}^d} \cdot f_{t(\vec{\mathbf{m}}-\vec{\mathbf{k}})-\{2M\}^d}
\end{aligned}$$

for $\vec{\mathbf{m}} \in \mathcal{M}_4$.

Therefore,

$$\begin{aligned}
\tilde{v}_{\vec{\mathbf{m}}} &= \sum_{\vec{\mathbf{k}}\in\{M,\dots,3M-1\}^d} z_{\vec{\mathbf{k}}-\{2M\}^d} \cdot f_{t(\vec{\mathbf{m}}+\{2M\}^d-\vec{\mathbf{k}})-\{2M\}^d} \\
&= \sum_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d} z_{\vec{\mathbf{k}}} \cdot f_{t(\vec{\mathbf{m}}-\vec{\mathbf{k}})-\{2M\}^d}
\end{aligned}$$

for $\vec{\mathbf{m}} \in \{-2M, \dots, 2M-1\}^d$. In the following, be $s(\vec{\mathbf{m}} - \vec{\mathbf{k}}) := t(\vec{\mathbf{m}} - \vec{\mathbf{k}}) - \{2M\}^d$. If $\vec{\mathbf{m}} \in \{-M, \dots, M-1\}^d$ and $\vec{\mathbf{k}} \in \{-M, \dots, M-1\}^d$, then follows for all $j \in \{1, \dots, d\}$ that

$$\begin{aligned} &m_j - k_j \in \{-2M+1, \dots, 2M-1\} \\ \Rightarrow &t(m_j - k_j) = \begin{cases} m_j - k_j, & m_j - k_j \in \{0, \dots, 2M-1\} \\ m_j - k_j + 4M, & m_j - k_j \in \{-2M, \dots, -1\} \end{cases} \\ \Rightarrow &t(m_j - k_j) - 2M = \begin{cases} m_j - k_j - 2M, & m_j - k_j \in \{0, \dots, 2M-1\} \\ m_j - k_j + 2M, & m_j - k_j \in \{-2M, \dots, -1\} \end{cases}, \end{aligned}$$

i.e. the array $f_{s(\vec{\mathbf{l}})} := f_{t(\vec{\mathbf{l}})-\{2M\}^d}$ is obtained from the array $f_{\vec{\mathbf{l}}}$ by a transformation s which cuts the array in each dimension into two halves and exchanges them (this is exactly the operation which - for one dimension - is provided in Matlab by the function `fftshift`). Obviously, this transformation is inverse to itself.

Thus, $\tilde{v}_{\vec{\mathbf{m}}} = \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} z_{\vec{\mathbf{k}}} \cdot f_{t(\vec{\mathbf{m}}-\vec{\mathbf{k}})-\{2M\}^d} = \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} z_{\vec{\mathbf{k}}} \cdot f_{s(\vec{\mathbf{m}}-\vec{\mathbf{k}})}$ can be calculated by

$$\tilde{v}_{\vec{\mathbf{m}}-\{2M\}^d} := (IFFT(\hat{v}_{\vec{\mathbf{n}}}))_{\vec{\mathbf{m}}} = IFFT((FFT(f_{\vec{\mathbf{l}}-\{2M\}^d}))_{\vec{\mathbf{n}}} \cdot (FFT(\tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d}))_{\vec{\mathbf{n}}}),$$

or, equivalently, $v_{\vec{\mathbf{m}}} = \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} z_{\vec{\mathbf{k}}} \cdot f_{\vec{\mathbf{m}}-\vec{\mathbf{k}}} = \sum_{\vec{\mathbf{k}} \in \{-M,\dots,M-1\}^d} z_{\vec{\mathbf{k}}} \cdot f_{s(s(\vec{\mathbf{m}}-\vec{\mathbf{k}}))}$ can be calculated by

$$v_{\vec{\mathbf{m}}-\{2M\}^d} = IFFT((FFT(f_{s(\vec{\mathbf{l}}-\{2M\}^d)}))_{\vec{\mathbf{n}}} \cdot (FFT(\tilde{z}_{\vec{\mathbf{k}}-\{2M\}^d}))_{\vec{\mathbf{n}}}),$$

For example, in the case of $d = 2$ this algorithm for the fast convolution of the arrays `f` and `z` can be implemented in Matlab as follows (assuming that an array `z` of size $2M \times 2M$ and an array `f` of size $4M \times 4M$ have already been initialized):

```
function v = fastconv2(f, z)
    f = fftshift(fftshift(f, 1), 2);      % 'transformation s'
    f = fft2(f);                          % Fourier transform of f

    zz = zeros (4*M, 4*M);                % 'tilde{z}'
    zz(M+1:3*M, M+1:3*M) = z;             % copy z to center of zz
    zz = fft2(zz);                        % Fourier transform of tilde{z}

    vv = zz.*f;                           % componentwise multiplication
    vv = ifft2(vv);                       % inverse Fourier transforms
    v = vv(M+1:3*M, M+1:3*M);             % use only the central part ...
end                                       % ... of size 2M*2M
```

Chapter 6

Implementations of the approximation algorithms

The algorithms for the simulation of OSSRF in harmonizable representation or in moving average representation which are presented in the previous chapter, have been implemented for the two-dimensional and the three-dimensional cases in Matlab and also in the programming language Java. In this chapter, some visualisations of OSSRFs which have been generated by these programs are presented, as well as also several statistics of the required system resources (memory and computing time) in relation to the size of the simulated random field. The source codes for the simulation functions in Matlab are also included in this chapter (however, sources of the Java implementation are not listed in this thesis because of their length, but they can be found as files on the attached CD).

6.1 Implementations in Matlab

6.1.1 Simulation of two-dimensional OSSRFs in Matlab

The algorithms for the simulation of two-dimensional OSSRF, which are described in sections 5.2 and 5.4, have been implemented in the Matlab language. These implementations are given in the following program listings and are based on Matlab programs for the simulation of OSSRF by Prof. H. P. Scheffler.

Simulation of harmonizable OSSRF

The first function implements the simulation of a harmonizable OSSRF. It has to be provided with the parameters of the OSSRF which should be simulated, and of the simulation procedure (A and M), and returns a matrix with the simulated values of the OSSRF and a matrix of the function values $f_{k,l} := \varphi\left(\xi_{k,l}\right)^{-H-\frac{q}{\alpha}}$:

Listing 6.1: The file simuH2ds.m

```
function [X,phi]=simuH2ds(alpha, H, A, M, N, a1, a2, v1, v2, C1, C2, rho)

    % two values which are repeatedly used in the calculations
    M2 = M*2;
    D = A/M;

    % generate complex random numbers
    Z = complex(randn(M2, M2), randn(M2,M2));
    Z = Z* D^(2/alpha);

    if alpha < 2
        A1shift = tan(pi*alpha/4);
        A1 = reshape( rstab(alpha/2, 1, M2*M2) , M2, M2) + A1shift;
        A1 = (cos(pi*alpha/4))^(2/alpha) *A1;
        Z = A1.^(1/2).*Z;
    end;
    clear('A1', 'A1shift');

    % calculate values of E-homogeneous function phi
    [k,l]=meshgrid(-A:D:A-D, -A:D:A-D);
    phi = (C1*(abs(k*cos(v1) + l*sin(v1))).^(rho/a1) ...
          + C2*(abs(k*cos(v2) + l*sin(v2))).^(rho/a2)) ...
          .^((-H-(a1+a2)/alpha)/rho);
    phi( -N+M+1 : N+M, -N+M+1 : N+M) = 0;
    clear('k', 'l');

    %multiply complex random numbers with values of phi
    newphi = phi.*Z;
    clear('Z');

    % FFT
    newphi = fftshift(fftshift(newphi, 1), 2);
    X = real(fft2(newphi)-sum(sum(newphi)));
    X = fftshift(fftshift(X, 1), 2);
end
```

In order to test this function and to simulate an OSSRF, this function can be called by the following MATLAB script, which calls the function, providing it with the necessary parameter values, and displays the values of the simulated random field in a picture (mapping the values to different colors):

Listing 6.2: The file ossrfHs2d.m

```
% initialization of parameters for simulation
alpha = 1.5;          H = 0.5  ;
M    = 2000;          N = 4;
a1   = 1;             a2 = 1;
v1   = 0;             v2 = 0.8;
```

```
C1  = 1;                C2 = 1;
rho = 2;                 A = 5.0;

% simulate ossrf
tic;
Y=simuH2ds(alpha,H,A,M,N,a1,a2,v1,v2,C1,C2,rho);
disp(sprintf('OSSRF of size %dx%d simulated in %g seconds.', ...
             2*M, 2*M, toc));

% display ossrf
figure, surf(Y), view(0,90), shading flat, axis('off');
```

The required amount of memory space for this program is particularly large during the simulation of non-gaussian random variables by the function `rstab`. This function does not simulate single random numbers one after another in a loop, but avoids using loops by manipulating the whole vector of random numbers at once (which is a usual strategy when programming in Matlab). This implies that the intermediate results are also stored as vectors of the length which is specified as the third parameter of the function `rstab`. In the presented program it is $(2M)^2$. During the calculation, the function initializes several variables with intermediate results as vectors of this length, so that twelve such vectors are in the memory at the end of a simulation by `rstab`. This means that if the parameter M has, for example, the value $M = 2000$, then the function `rstab` needs memory space for more than $12 \cdot 4000^2 = 192000000$ double-precision floating point numbers (with 8 bytes each), which is about 1.43 GB - or 96 bytes per simulated random number. Therefore, the amount of required memory space can be reduced by calling the `rstab` function several times with a smaller third parameter in order to simulate a part of the random numbers, instead of simulating all random numbers at once. If the respective lines of code in the function `simuH2ds`

```
        A1 = reshape( rstab(alpha/2, 1, M2*M2) , M2, M2) + A1shift;
        A1 = (cos(pi*alpha/4))^(2/alpha) *A1;
```

are replaced, for example, by the following lines

```
        A1 = zeros(M2);
        A1(   1:M,     1:M)  = reshape(rstab(alpha/2,1,M*M) ,M,M);
        A1(   1:M,  1+M:M2)  = reshape(rstab(alpha/2,1,M*M) ,M,M);
        A1(1+M:M2,     1:M)  = reshape(rstab(alpha/2,1,M*M) ,M,M);
        A1(1+M:M2,  1+M:M2)  = reshape(rstab(alpha/2,1,M*M) ,M,M);
        A1 = (cos(pi*alpha/4))^(2/alpha)* (A1 + A1shift);
```

then the function `rstab` uses only vectors of length M^2 instead of $4M^2$. Considering the fact that now the matrix `A1` with $4M^2$ has been initialized (with zeros) before applying `rstab` (and therefore also occupies memory space), the amount of memory needed for the simulation of the random numbers in `A1` has been reduced from $8 \cdot 12 \cdot 4M^2 = 384M^2$

bytes to $8 \cdot (4M^2 + 12M^2) = 128M^2$ bytes, which is a reduction by the factor 3. This procedure is implemented in the function `simuH2dl`, which is called, e.g. by the script in the file `ossrfH2d.m`. Usually, it is a bit slower (by less than 1%) than the first (simpler) code variant.

Another, even more memory-saving alternative for the simulation of the non-gaussian random variables is to call the function `rstab` in a loop (with more than 4 repetitions of the loop), as in the following code example:

```
numBlocks = 20;
bStart = floor((0:numBlocks-1)*M2/numBlocks)+1;
bEnd   = floor((1:numBlocks)  *M2/numBlocks);
A1 = zeros(M2);
for ib = 1:numBlocks
    bSize = bEnd(ib)-bStart(ib)+1;
    A1(bStart(ib):bEnd(ib), :) = ...
                 reshape(rstab(alpha/2, 1, bSize*M2), bSize, M2);
end
A1 = (cos(pi*alpha/4))^(2/alpha) * (A1 + A1shift);
```

With this variant for the simulation of the non-gaussian random variables, the simulation of an OSSRF takes about 2% more time than with the first (simpler) variant.

Another task which requires much memory space is the display of the simulated OSSRF as an image. Displaying the image only for a part of the random field, rather than for all simulated values, reduced the memory consumption of the display functions and allows to simulate larger fields without causing an error due to a lack of available memory (Anyways, random fields with a width and height of e.g. 2000 - or even more - can't be shown in all details on usual computer displays, because the displays don't have such a high resolution). This can be achieved by exchanging the line

```
figure, surf(Y),  view(0,90),  shading flat,  axis('off');
```

by

```
m2 = M+floor(displaySize/2);
m1 = m2 - displaySize+1;
figure, surf(Y(m1:m2, m1:m2)  ),  view(0,90);
shading flat,  axis('off');
```

where the variable `displaySize` has been defined before as the size of the displayed central part of the OSSRF (see the file `ossrfH2d.m`).

Some examples of harmonizable OSSRFs which have been simulated using this Matlab program, are shown in figure 6.1.

Simulation of moving-average OSSRF

The algorithm for the simulation of two-dimensional OSSRF in moving-average representation, which is described in section 5.4, has been implemented in the following two files: The function `simuM2d` simulates an OSSRF, and the script `ossrfM2d` calls this function and displays the resulting random field (similar to the files `simuH2ds.m` and `ossrfHs2d.m`, or `simuH2dl.m` and `ossrfH2d.m` in the harmonizable case).

Listing 6.3: The file simuM2d.m

```
function [X,phi]=simuM2d(alpha, H, A, M, a1, a2, v1, v2, C1, C2, rho)

    % two values which are repeatedly used in the calculations
    M2 = M*2;
    D = A/M;

    %generate alpha-stable random numbers
    Z= reshape(rstab(alpha, 0, M2*M2), M2, M2);
    Z=D^(2/alpha)*Z;

    %calculate values of E-homogeneous function phi
    [k,l]=meshgrid(-2*A:D:2*A-D, -2*A:D:2*A-D);
    phi = (C1*(abs(k*cos(v1) + l*sin(v1))).^(rho/a1) ...
          + C2*(abs(k*cos(v2) + l*sin(v2))).^(rho/a2))...
          .^((H-(a1+a2)/alpha)/rho);
    phi(M2:M2+1,M2:M2+1)=0;
    clear('k', 'l');

    %convolution of random numbers and values of phi, using the fft
    phi = fftshift(phi, 1);
    phi = fftshift(phi, 2);
    phi = fft2(phi);
    Y = zeros (M*4, M*4);
    Y(M+1:M*3, M+1:M*3) = Z;
    clear('Z');
    Y = fft2(Y);
    Y = Y .* phi;
    clear('phi');
    Y = ifft2(Y);
    X = Y (M+1:M*3, M+1:M*3);
    X=X-X(M+1,M+1);
end
```

Listing 6.4: The file ossrfM2d.m

```
% initialization of parameters for simulation
alpha = 2.0;        H = 0.6 ;
a1 = 1.0;          a2 = 2.0;
v1 = 1.0;          v2 = 2.57;
C1 = 1;            C2 = 1;
```

```
rho = 2.0;
A = 20.0;          M = 800;

%simulate OSSRF
tic;
[Y,phi]=simuM2d(alpha, H,A,M,a1,a2,v1,v2,C1,C2,rho);
disp(sprintf('OSSRF of size %dx%d simulated in %g seconds.', ...
             2*M, 2*M, toc));

%display OSSRF
figure, surf(Y), view(0,90), shading flat, axis('off');
set(gca, 'ActivePositionProperty', 'position', 'Position', [0 0 1 1]);
```

In figure 6.2, some examples of two-dimensional OSSRF in moving-average representation are shown, which have been generated with the given Matlab functions.

As an alternative to setting the values of the parameters explicitly in the source code (of the file `ossrfM2d.m`), and therefore having to edit this file always when a parameter must be changed, the script `ossrfMi2d` can be used instead, which prompts the user to input these values. Analogously, the script `ossrfHi2d` asks the user to state parameter values for the simulation of a harmonizable OSSRF (unlike `ossrfH2dl`, where the parameter values are set in the source code), and with the command `ossrfgui2d` a dialog window can be opened (see figure 6.3 (a)) which enables the user to input and change the parameter values and start the simulation of an OSSRF in harmonizable or moving average representation.

Required memory and CPU time

Table 6.1 shows the times which were required for the simulation of two-dimensional OSSRF with this Matlab programs. These times are also considered in relation to the size of the simulated field (i.e. as time for the calculation per million values of the simulated field) in this table. The times were taken with the Matlab commands `tic` and `toc`. The used test system (which was also used for all other simulations which are described in this chapter) is an Apple MacBook with a 2.4 GHz Intel Core 2 Duo processor and 4 GB RAM. The tests of the Matlab programs were performed using one CPU core (i.e. no multithreading) and with 2GB free RAM. Using this amount of memory, the maximum size of an OSSRF which could be simulated was about 31.4 mio. values (i.e. parameter $M = 2800$, generating a matrix of size 5600×5600) for a harmonizable field, and about 10.2 mio. values ($M = 1600$, or a 3200×3200-matrix) in the moving-average case. This demonstrates the fact that the simulation of a harmonizable OSSRF can be achieved much more memory-efficient than the simulation of an equally-sized moving-average OSSRF. The main reason is the need to perform FFTs with $4M \times 4M$- matrices of complex numbers during the fast calculation of the convolution in the simulation of moving average OSSRFs, which are four times larger than the $2M \times 2M$-matrices which are used in the functions for the simulation of harmonizable random fields. The numbers in table 6.1 demonstrate that:

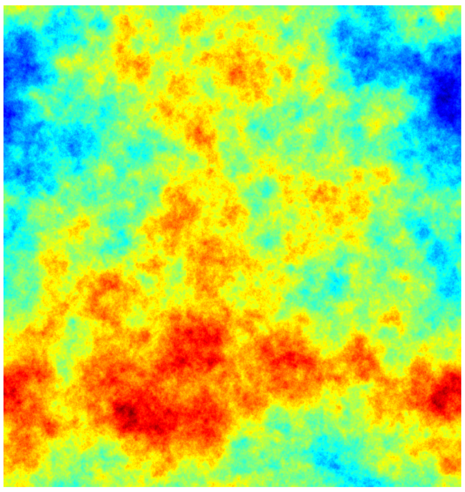
(a) The time for the calculation is approximately proportional to the number of points for which the OSSRF is simulated (if the other parameters remain unchanged).

(b) There is a huge difference between the simulation time of a harmonizable gaussian random field and the simulation time of a harmonizable non-gaussian field of the same size, the latter being approximately three times as large as the first: The approximation of a gaussian field takes about 0.8 seconds per mio. elements of the simulated matrix, while for a non-gaussian field about 2.4 seconds per mio. elements are needed. This is due to the fact that the part of the calculation which is only needed when simulating non-gaussian OSSRFs (mainly the simulation of non-gaussian random variables with `rstab`) takes about twice as much time as all the calculation steps for the simulation of gaussian random fields (i.e. the simulation of gaussian r.v.s, the calculation of the function phi, and a Fourier transform).

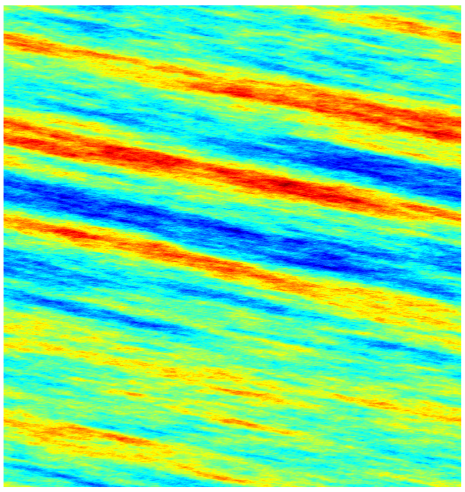
(c) The simulation of moving-average random fields does not only require much more memory space, but also much more time for the calculation than the simulation of a harmonizable field of the same size.

size of the OSSRF		simulation time			simulation time per mio. values		
`M`	mio. values	ha. $\alpha = 2$	ha. $\alpha < 2$	mov. av.	ha. $\alpha = 2$	ha. $\alpha < 2$	mov. av.
100	0.04	0.035	0.097	0.238	0.88	2.43	5.95
200	0.16	0.118	0.364	0.95	0.74	2.28	5.94
400	0.64	0.462	1.49	3.80	0.72	2.33	5.94
800	2.56	1.84	6.01	16.7	0.72	2.35	6.52
1600	10.24	8.07	24.4	68.0	0.79	2.38	6.64
2800	31.36	25.7	75.2	$--$	0.82	2.40	$--$

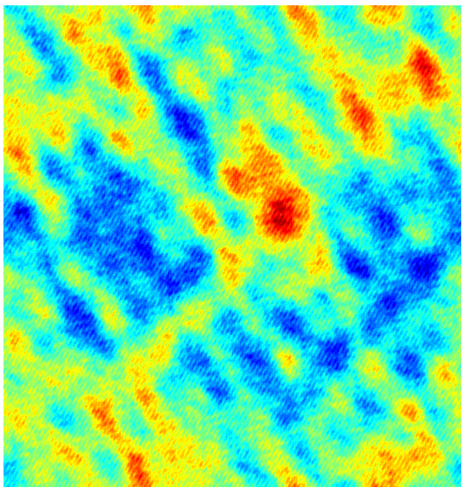
Table 6.1: Time (in s.) for the calculation of two-dimensional OSSRF in Matlab (ha. "$\alpha = 2$" = gaussian, harmonizable OSSRFs, "ha. $\alpha < 2$" = non-gaussian, harmonizable OSSRFs, "mov. av." = moving average OSSRFs).



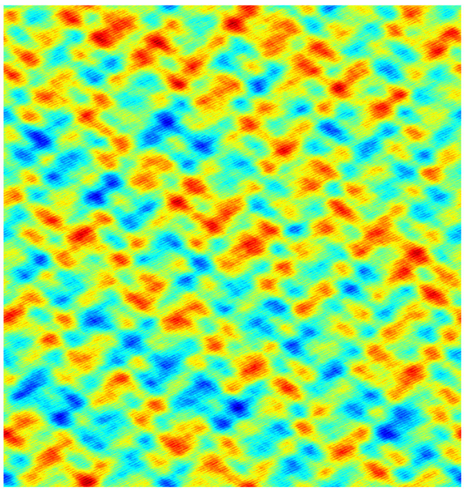
(a) $\alpha = 2.0$, isotropic



(b) $\alpha = 2.0$, anisotropic

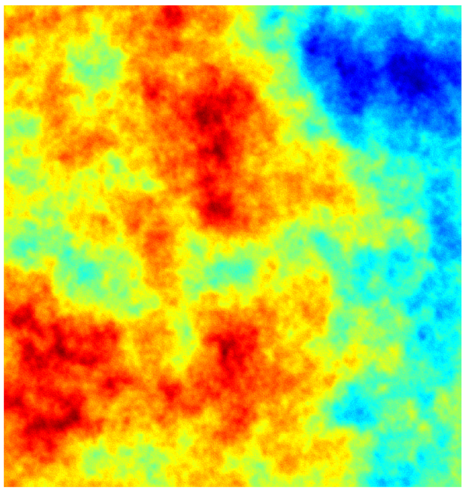


(c) $\alpha = 1.5$, isotropic

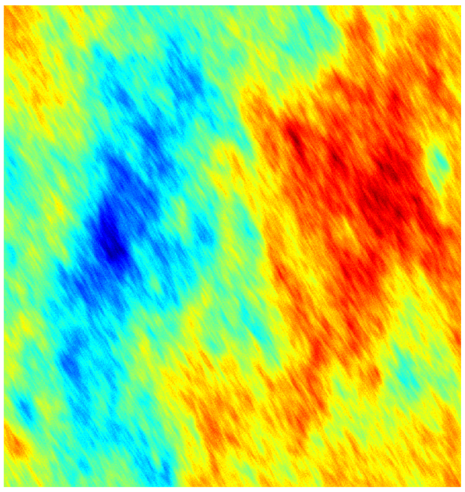


(d) $\alpha = 1.5$, anisotropic

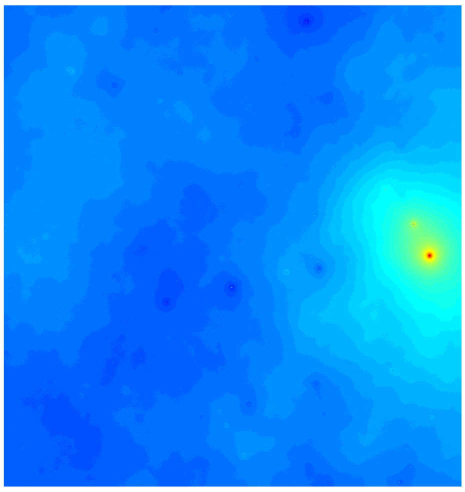
Figure 6.1: Some examples of two-dimensional harmonizable OSSRFs, simulated with Matlab.



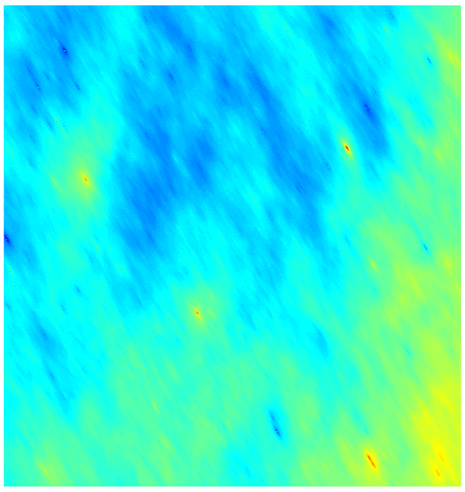
(a) $\alpha = 2.0$, isotropic



(b) $\alpha = 2.0$, anisotropic



(c) $\alpha = 1.8$, isotropic



(d) $\alpha = 1.8$, anisotropic

Figure 6.2: Some examples of two-dimensional moving-average OSSRFs, simulated with Matlab.

6.1.2 Simulation of three-dimensional OSSRF in Matlab

The algorithms for the simulation of OSSRF, which are described in sections 5.3 and 5.5, have been implemented in the Matlab language also for the three-dimensional case. These implementations are based on the programs for the simulation of two-dimensional OSSRF (which have been presented on the previous pages).

Harmonizable three-dimensional OSSRF can be simulated in Matlab with the function `simuH3ds`, whose source code is given in the following listing. When 2 GB (=2048 MB) free memory are available, then this function can simulate gaussian fields up to a size of about 26.5 million data points (parameter $M = 149$) and non-gaussian stable fields up to a size of approximately 12.8 mio. data points (parameter $M = 117$).

Listing 6.5: The file simuH3ds.m

```
function Y=simuH3ds (alpha,H,A, M, N, a1,a2,a3, v1,v2,v3, C1, C2, C3, rho)

    % two values which are repeatedly used in the calculations
    M2 = M*2;
    D = A/M;

    % Part 1: Simulation of isotropic stable r.v.s
    %   Part 1.1: Simulation of complex isotropic gaussian r.v.s
    disp('Simulate stable random variables...');

    %   Part 1.1: generate complex random numbers
    Z = complex(randn(2*M, 2*M, 2*M), randn(2*M, 2*M, 2*M));
    Z = Z* D^(3/alpha);

    %   Part 1.2: If alpha < 2, then multiply with sqrt of stable r.v.
    %   with parameter beta' = 1 and alpha' = alpha/2
    if alpha < 2
        A1shift = tan(pi*alpha/4);
        A1 =reshape(rstab(alpha/2,1, M2*M2*M2), M2,M2,M2);
        A1 = (cos(pi*alpha/4))^(2/alpha) * (A1 + A1shift);
        Z = A1.^(1/2).*Z;
        clear('A1', 'A1shift');
    end;

    % Part 2: Calculate values of the function phi
    disp('Calculate function PHI ...');

    [k,l,m] = meshgrid(-M*D:D:(M-1)*D);

    phi = (  C1*(abs( k*v1(1) + l*v1(2) + m*v1(3))).^(rho/a1) ...
           + C2*(abs( k*v2(1) + l*v2(2) + m*v2(3))).^(rho/a2) ...
           + C3*(abs( k*v3(1) + l*v3(2) + m*v3(3))).^(rho/a3) ...
          ).^((-H-(a1+a2+a3)/alpha)/rho);
    clear('k', 'l', 'm');

    % cut out area in the center
    phi(M-N+1:M+N, M-N+1:M+N, M-N+1:M+N) = 0;

    % Part 3: FFT
    disp('FFT ...');
    Y = phi.*Z;
    clear('phi', 'Z');
    Y = fftshift(fftshift(fftshift(Y, 1), 2), 3);
    sumY = sum(sum(sum(Y)));
    Y = fftn(Y);
    Y = real(Y-sumY);
    Y = fftshift(fftshift(fftshift(Y, 1), 2), 3);
end
```

The required amount of memory can be reduced by calculating the simulated random numbers and the values of the function phi in several parts, e.g. separately for each two-dimensional plane with the same z-coordinate, rather than calculating this values at once for the whole three-dimensional array. The following variation of the simulation function (`simuH3dl`) uses this improvement and allows the simulation of three-dimensional OSSRFs in harmonizable representation with a maximum parameter value of $M = 177$ (for gaussian and for non-gaussian OSSRFs), corresponding to a maximum size of almost 44.4 millionen data points, when 2 GB of free memory space are available. This maximum sample size is about 68% larger than before in the gaussian case, and almost 3.5 times the previous maximum number of simulated values in the non-gaussian case.

Listing 6.6: The file simuH3dl.m

```
function Y=simuH3dl (alpha,H,A, M, N, a1,a2,a3, v1,v2,v3, C1, C2, C3, rho)

    % two values which are repeatedly used in the calculations
    M2 = M*2;
    D = A/M;

    [k, l] = meshgrid(-M*D:D:(M-1)*D);
    arrayD = D*ones(M2, M2);

    Y = zeros(2*M, 2*M, 2*M);    % reserve space for large array
    for j = 1:M2
        % Part 1: Simulation of isotropic stable r.v.s
        %   Part 1.1: Simulation of complex isotropic gaussian r.v.s
        Z = complex(randn(2*M, 2*M), randn(2*M, 2*M));
        Z = Z* D^(3/alpha);

        %   Part 1.2: If alpha < 2, then multiply with sqrt of stable r.v.
        %   with parameter beta ' = 1 and alpha ' = alpha/2
        if alpha < 2
            A1shift = tan(pi*alpha/4);
```

```
                A1 =reshape(rstab(alpha/2,1, M2*M2), M2,M2);
                A1 = (cos(pi*alpha/4))^(2/alpha) * (A1 + A1shift);
                Z = A1.^(1/2).*Z;
                clear('A1', 'A1shift');
            end;

            % Part 2: Calculate values of the function phi
            m =  (j-M-1)*arrayD;
            phi = (  C1*(abs( k*v1(1) + l*v1(2) + m*v1(3))).^(rho/a1) ...
                + C2*(abs( k*v2(1) + l*v2(2) + m*v2(3))).^(rho/a2) ...
                + C3*(abs( k*v3(1) + l*v3(2) + m*v3(3))).^(rho/a3) ...
              ).^((-H-(a1+a2+a3)/alpha)/rho);
            clear( 'm');

            % cut out area in the center
            if ((j>= M-N+1)&&(j<= M+N))
                phi(M-N+1:M+N, M-N+1:M+N) = 0;
            end;

            Z = phi.*Z;

            % fftshift
            Z = fftshift(fftshift(Z, 1), 2);
            if (j <= M)
                Y(:, :, j+M) = Z;
            else
                Y(:, :, j-M) = Z;
            end;
        end;  % end of loop 'for j= 1:M2'

        % Part 3: FFT
        clear('phi', 'Z', 'k', 'l', 'arrayD');

        sumY = sum(sum(sum(Y)));
        Y = fftn(Y);
        Y = real(Y-sumY);
        Y = fftshift(fftshift(fftshift(Y, 1), 2), 3);
end
```

OSSRFs in moving average representation can be simulted with the function `simuM3d`. If 2 GB free memory space are available, random fields with a maximal parameter value of $M = 80$ (and a size of about 4.1 mio. data points) can be simulated by this function:

Listing 6.7: The file simuM3d.m

```
function X=simuM3d (alpha, H, A, M, a1,a2,a3, v1,v2,v3, C1,C2,C3, rho)

    % two values which are repeatedly used in the calculations
    M2 = M*2;
    D = A/M;

    % Part 1: Simulation of symmetric stable random variables
    Z=rstab(alpha, 0, M2*M2*M2);
    Z = D^(3/alpha)*reshape(Z, M2, M2, M2);

    % Part 2: Calculation of values for the function phi
    [k,l,m] = meshgrid(-2*A:D:2*A-D);
    phi = (  C1*(abs( k*v1(1) + l*v1(2) + m*v1(3))).^(rho/a1) ...
           + C2*(abs( k*v2(1) + l*v2(2) + m*v2(3))).^(rho/a2) ...
           + C3*(abs( k*v3(1) + l*v3(2) + m*v3(3))).^(rho/a3) ...
          ).^((H-(a1+a2+a3)/alpha)/rho);
    phi(M2:M2+1, M2:M2+1, M2:M2+1) = 0;
    clear('k', 'l', 'm');

    % Part 3:  Convolution
    phi = fftshift(phi, 1);
    phi = fftshift(phi, 2);
    phi = fftshift(phi, 3);
    phi = fftn(phi);
    Y = zeros (M*4, M*4, M*4);
    Y(M+1:M*3, M+1:M*3, M+1:M*3) = Z;
    clear('Z');
    Y = fftn(Y);
    Y = Y .* phi;
    clear('phi');
    Y = ifftn(Y);
    X = Y (M+1:M*3, M+1:M*3, M+1:M*3);
    X=X-X(M+1,M+1, M+1);
end
```

These functions for the simulation of three-dimensional OSSRF (`simuH3ds`, `simuH3dl` and `simuM3d`) are called by the script `ossrf3d` which is given in the following listing. As an alternative, they can also (like in the two-dimensional case) be started from a dialog window which allows the user to input the parameters for the simulation. This dialog window can be opened with the command `ossrfgui3d` (see figure 6.3).

Listing 6.8: The file ossrf3d.m

```
alpha = 2.0;            A = 5;                  M = 50;
rho = 2;                H = 0.9 ;               N = 1;
a1 = 1;                 a2 = 1;                 a3 = 1;
v1 = [0.8, 0, 0.6];     v2 = [0.0, 1.0, 0.0];     v3 = [-0.6, 0, 0.8];
C1 = 1;                 C2 = 1;                 C3 = 1;

harmonizable = true;    %set this 'true' to simulate a harmonizable OSSRF
                        %      or 'false' for a moving average OSSRF

maxNumBlue = 40000;
maxNumRed  = 40000;
```

```

% Simulation of OSSRF
tic;
if (harmonizable)
    disp(sprintf('Simulate a harmonizable OSSRF of size %d^3...', 2*M));
    % use simpler version for small OSSRF,
    % and more sophisticated version with for-loop for larger OSSRF
    if (M<=50)
      X = simuH3ds(alpha, H, A, M, N, a1,a2,a3, v1,v2,v3, C1,C2,C3, rho);
    else
      X = simuH3dl(alpha, H, A, M, N, a1,a2,a3, v1,v2,v3, C1,C2,C3, rho);
    end;
else
    disp(sprintf('Simulate a moving average OSSRF of size %d^3...', 2*M));
    X = simuM3d(alpha, H, A, M, a1, a2, a3, v1, v2, v3, C1, C2, C3, rho);
end;
disp(sprintf('OSSRF of size %dx%dx%d simulated in %g seconds.', ...
                2*M, 2*M, 2*M, toc));

% Display OSSRF
tic;
[figures, movies] = displayossrf3d(X, A, M, 4, 'ossrf3d-1.avi', ...
          'ossrf3d-2.avi', 'ossrf3d-3.avi', 'ossrf3d-4.avi',   ...
          maxNumBlue, maxNumRed);
disp(sprintf('OSSRF of size %dx%dx%d displayed in %g seconds.', ...
                  2*M, 2*M, 2*M, toc));
```

This script (as does also the graphical interface `ossrfgui3d`) uses the function `displayossrf3d` in order to display up to four series of images of the generated random fields:

(a) A series of two-dimensional cuts through the three-dimensional field: For each value of the z-coordinate, the set of simulated points with this coordinate value is displayed as a two-dimensional image (with x-coordinate axis from left to the right and y-coordinate axis pointing up). Thereby, the z-axis is interpreted as the dimension time.

(b) A three-dimensional plot of the points with the largest and smallest values of the random field, which is viewed from different angles.

(c) A three-dimensional isosurface plot of the same points, which is also viewed from different sides.

(d) A red/cyan anaglyph version of this three-dimensional isosurface plot. This plot is intended to be viewed with red/cyan 3d-glasses, in order to see it three-dimensional.

Examples for this plots are shown in figures 6.4 to 6.6. The visualizations in figure 6.6 should be viewed using red/cyan 3d-glasses. Because of the length of the file `displayossrf3d.m`, it is not listed here completely, but only a few short sections shall be quoted here (however, the entire file can be found, together with all other source code files, on the attached CD). In this code fractions, the variable `X` is the given array of the values of the OSSRF, `M` is the parameter M and `M2` the length of one side of the random field (i.e. `M2` $= 2 *$ `M`).

Listing 6.9: display of a series of two-dimensional layers

```
% find minimal and maximal values of simulated OSSRF
% and norm it to values in the interval[0, 1]
xmax = max(max(max(X)));
xmin = min(min(min(X)));
Y = (X-xmin)/(xmax-xmin);

% ==========================================================
% Figure 1: Draw a series of two-dimensional images
figures(1) = figure('Position',[10 10 500 500], 'Color', 'white');
set(gca, 'ActivePositionProperty', 'position', 'Position', [0 0 1 1]);
nFrames = M2;                    % number of frames in the movie
Frames = moviein(nFrames);  % initialize the matrix 'Frames'

for k=1:nFrames
    surf( Y(: , :, k)),  view(0,90);
    shading interp,  axis('off');
    caxis([0,1]);
    drawnow;
    Frames(:,k)=getframe;
end
movies(1, 1:nFrames) = Frames;
% Now save as movie:
if (length(aviFilename1) > 0)
    movie2avi(Frames, aviFilename1,'compression', 'None','fps',15);
end;
```

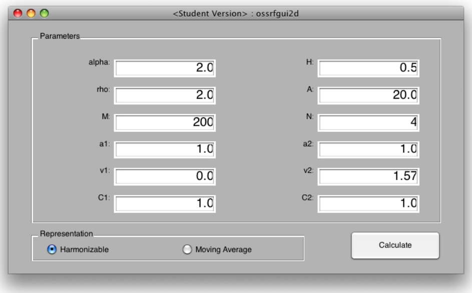
Listing 6.10: display of a three-dimensional scatter plot

```
    % three-dimensional scatter plot
    figures(2) = figure('Position',[520 10 500 500],'Color','white');
    pba = pi/A;
    [mgx, mgy, mgz] = meshgrid(-pba*M:pba:pba*(M-1));
    view(30, 27);
    hold on
    plot3(mgx(Y>redLim), mgy(Y>redLim), mgz(Y>redLim), 'r.');
    plot3(mgx(Y<bluLim), mgy(Y<bluLim), mgz(Y<bluLim), 'b.');
    hold off
    axis([-pba*M,pba*(M-1), -pba*M,pba*(M-1), -pba*M,pba*(M-1)]);
    axis vis3d;
    grid on;
```

When 2 GB of free memory are available, harmonizable OSSRFs with a maximal parameter value of $M = 160$ (and a size of about 32.8 mio. simulated values) and moving-average fields with a parameter M up to $M = 80$ (about 4.1 mio. values) can be simulated. Like for the two-dimensional OSSRFs, the necessary computing time was also tested for some three-dimensional fields of different sizes, with the results which are given in table 6.2. From this table, the same conclusions can be drawn as in the two-dimensional case: For the tested fields, the time of simulation is approximately proportional to the number of simulated values (i.e. proportional to M^3 in the case of three-dimensional fields), the simulation of non-gaussian harmonizable fields takes about three times as long as the simulation of gaussian harmonizable fields, and the simulation of moving-average fields takes even much more time.

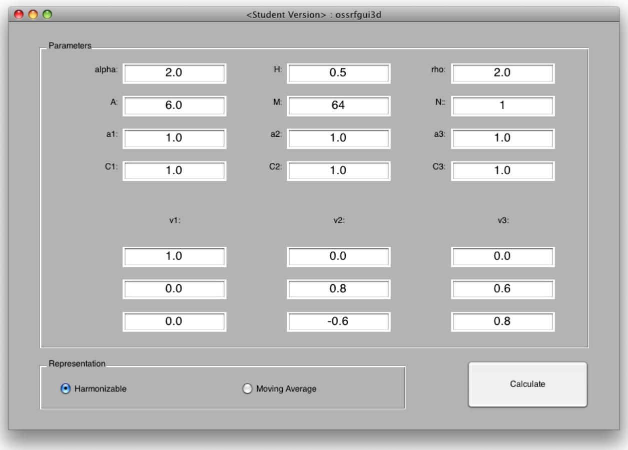
size of the OSSRF		simulation time			simulation time per mio. values		
`M`	mio. values	ha. $\alpha = 2$	ha. $\alpha < 2$	mov. av.	ha. $\alpha = 2$	ha. $\alpha < 2$	mov. av.
50	1.0	0.87	2.54	9.95	0.87	2.54	9.95
64	2.1	1.84	5.21	20.99	0.88	2.48	9.99
80	4.1	3.42	9.83	43.89	0.84	2.40	10.72
100	8.0	7.03	19.46	−−	0.88	2.43	−−
128	16.8	15.21	41.15	−−	0.91	2.45	−−
160	32.8	30.98	81.93	−−	0.95	2.50	−−

Table 6.2: Time (in s.) for the calculation of three-dimensional OSSRF in Matlab (ha. = harmonizable, mov. av. = moving average).



alpha: 2.0
rho: 2.0
M: 200
a1: 1.0
v1: 0.0
C1: 1.0
H: 0.5
A: 20.0
N: 4
a2: 1.0
v2: 1.57
C2: 1.0
[✓ Harmonizable] Moving Average

(a) ossrfgui2d



alpha: 2.0
H: 0.5
rho: 2.0
A: 6.0
M: 64
N:: 1
a1: 1.0
a2: 1.0
a3: 1.0
C1: 1.0
C2: 1.0
C3: 1.0
v1: 1.0, 0.0, 0.0
v2: 0.0, 0.8, -0.6
v3: 0.0, 0.6, 0.8
[✓ Harmonizable] Moving Average

(b) ossrfgui3d

Figure 6.3: Dialog windows for the input of parameters for the simulation of OSSRFs in Matlab.

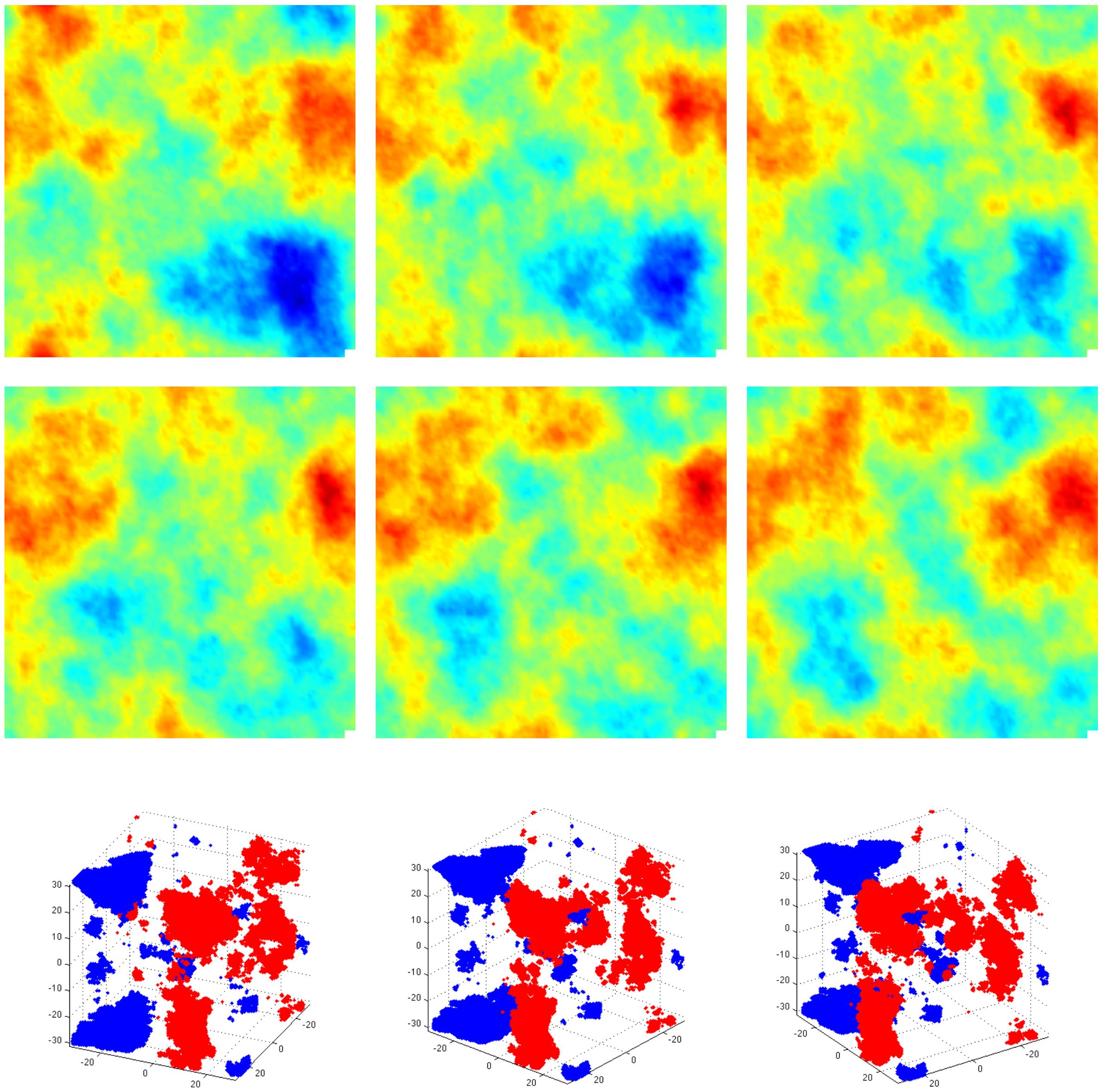


Figure 6.4: Visualization (types (a) and (b)) of a three-dimensional harmonizable OSSRF with $\alpha = 1.8$.

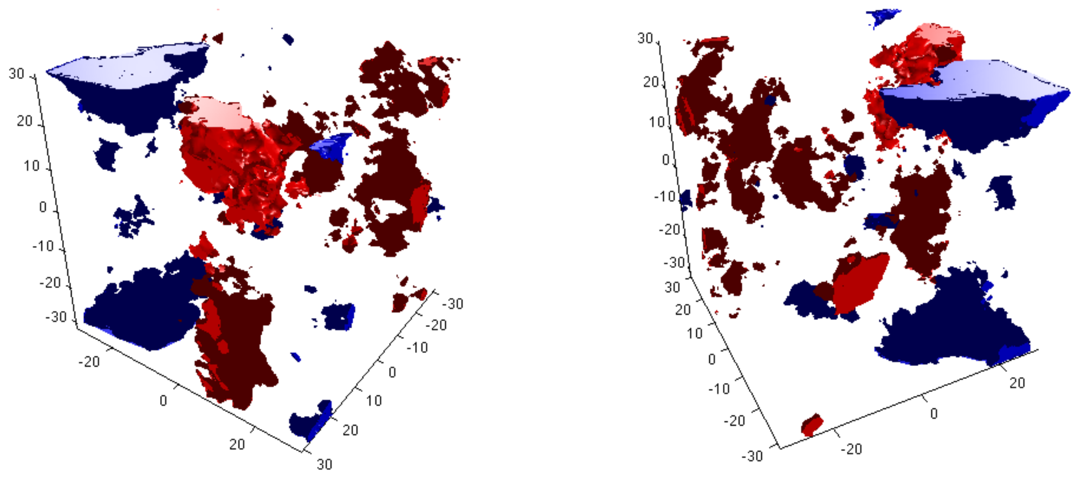


Figure 6.5: Visualization (type (c)) of a three-dimensional harmonizable OSSRF with $\alpha = 1.8$.

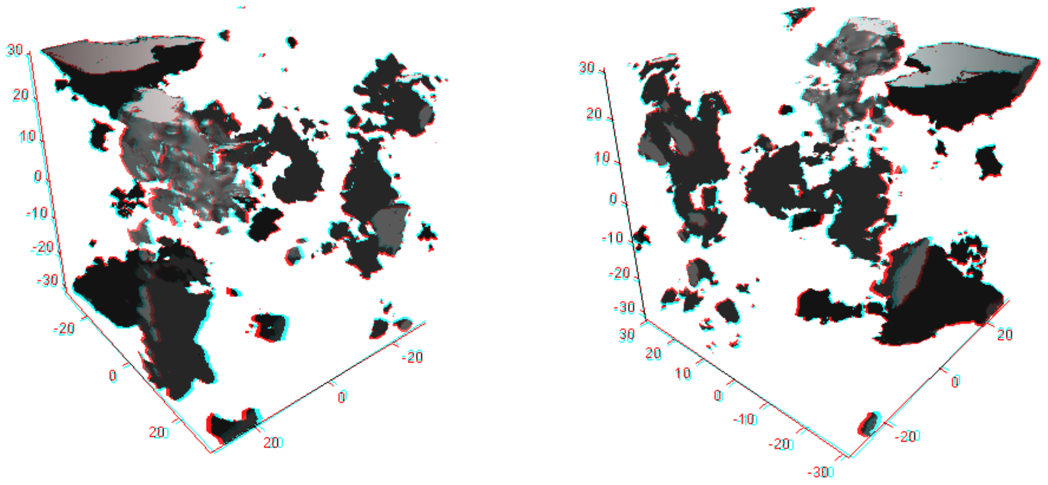


Figure 6.6: Visualization of a three-dimensional harmonizable OSSRF with $\alpha = 1.8$ (type (d): anaglyph plot. These pictures should be viewed with red/cyan 3d-glasses).

6.2 Implementations in Java

6.2.1 Simulation of two-dimensional OSSRF in Java

The simulation of two-dimensional OSSRF in harmonizable or moving-average representation has been also implemented in the programming language Java (the source code of this program is too large to be quoted in this thesis, but is included on the attached CD-ROM). The simulation algorithms have been implemented in a version which uses variables of the type “double” (64 bit long) for all floating-point data (like the implementations in MATLAB), and a version which saves intermediate results in “float” variables (32 bit long). The second version needs only approximately half as much memory space for the simulation of large random fields as the first implementation, which allows the simulation of larger fields with a certain given amount of available memory. Saving the results in 32-bit fields instead of 64-bit fields reduces the precision of the calculation, but for the simulation of random fields a precision of about 5 digits is usually acceptable, and a comparisons of both versions using the same simulated random fields showed differences in the magnitude of only about 10^{-5}. Therefore also the second version is a useful method for the simulation of random fields.

In order to increase the speed of the simulation, most of the calculations can be split into several parts which can be processed in parallel threads (on a CPU with multiple cores). The simulation of harmonizable OSSRF has been also implemented in variants which use parallel computing, and tests of the computing times showed that this improvement indeed considerably increases the speed of the calculations on computers with multiple CPU cores.

Thus, the simulation of two-dimensional harmonizable random fields has been implemented in four versions (which can be chosen by editing the file “options.txt” in the program folder: Using 32-bit or 64-bit floating point variables for the intermediate results during the simulation, and each of these versions with or without the use of multiple threads), and two versions exist for the simulation of moving-average random fields (32-bit or 64-bit variables for intermediate data).

When simulating with a variant which uses only 64-bit (“double precision”) floating point variables and 2 GB of memory, harmonizable fields up to a parameter value of about $M = 5750$ (with approximately 132 mio. simulated values) and moving-average fields up to about $M = 2000$ (with 16 mio. values) can be simulated (i.e. with the Java program, the simulation of a random field needs less memory than with the Matlab implementation, and therefore larger fields can be simulated with the same available amount of memory). When a variant is used which stores intermediate results in 32-bit variables, then the needed amount of memory per simulated value is approximately reduced to the half, so that random fields can be simulated which are about twice as large as with the 64-bit variant. The upper limit for the size of a harmonizable field is now about 262 mio. simulated values (with the parameter $M = 8100$), and moving average fields can be simulated up to a parameter value of about $M = 2880$ (i.e. about 33 mio. simulated values). As examples, the figures 6.7 and 6.8 show two-dimensional harmonizable random fields with a parameter value of $M = 8000$ (e.g. with 256 mio. simulated values) and a moving-average random field with $M = 2800$ (e.g. about 31 mio. simulated values).

size of the OSSRF		harmonizable, 1 thread		harmonizable, 2 threads		moving average
`M`	mio. values	gaussian	non-gaussian	gaussian	non-gaussian	
100	0.04	0.068	0.095	0.050	0.065	0.474
200	0.16	0.268	0.378	0.169	0.230	1.84
400	0.64	1.06	1.52	0.633	0.887	7.61
800	2.56	4.29	6.15	2.51	3.51	32.5
1600	10.24	17.9	25.2	10.1	13.9	164.1
2000	16.00	31.4	43.2	18.1	24.1	343.2
2800	31.36	67.7	90.2	39.5	51.4	−−
5600	125.44	342.3	424.3	186.5	228.3	−−
5750	132.25	460.5	556.4	261.3	314.2	−−

Table 6.3: Times (in s.) for the calculation of two-dimensional OSSRFs with the Java implementation (using 64bit floats).

size of the OSSRF		harmonizable, 1 thread		harmonizable, 2 threads		moving average
`M`	mio. values	gaussian	non-gaussian	gaussian	non-gaussian	
100	0.04	1.70	2.38	1.25	1.63	11.9
200	0.16	1.68	2.36	1.06	1.44	11.5
400	0.64	1.66	2.38	0.99	1.39	11.9
800	2.56	1.68	2.40	0.98	1.37	12.7
1600	10.24	1.75	2.46	0.99	1.36	16.0
2000	16.00	1.96	2.70	1.13	1.51	21.5
2800	31.36	2.16	2.88	1.26	1.64	−−
5600	125.44	2.73	3.38	1.49	1.82	−−
5750	132.25	3.48	4.21	1.98	2.38	−−

Table 6.4: Times (in s.) per mio. elements for the calculation of two-dimensional OSSRFs with the Java implementation (using 64bit floats).

In the tables 6.3 and 6.4, computing times for the simulation of two-dimensional OSSRFs (using the implementations with 64-bit floating point numbers) in different sizes are summarized. These tables compare the computing times for the simulation of gaussian OSSRFs vs. non-gaussian OSSRFs, and the computing times for simulations using only one thread vs. parallel calculations in two threads. The following observations are made with the numbers in these tables:

(a) As with the implementations in Matlab, the computing times for an OSSRF of a certain type is approximately proportional to the number of simulated values. However, for large fields, the time per mio. values rises slightly when increasing the size of the field.

(b) The simulation of a gaussian harmonizable or of a moving average random field with the Java program (without using parallel threads) takes about two to three times as much time than with the Matlab program. The simulation of a non-gaussian harmonizable random field, however, has a similar speed as with the Matlab implementation.

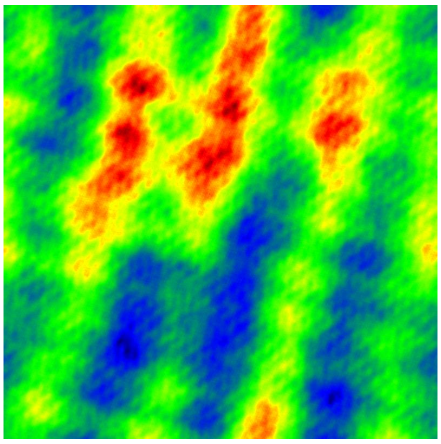
(c) The simulation of non-gaussian harmonizable OSSRFs takes between 20 and 45 percent more time than the simulation of gaussian harmonizable OSSRFs of the same size. This difference is much smaller than in the Matlab implementation, where the simulation of a non-gaussian field takes about 200 percent more time. This can be explained by the use of a very fast and optimized implementation of the FFT algorithm by Matlab, making the part of the calculations which is performed in both cases much faster in Matlab than in Java, while the simulation of non-gaussian random variables in Matlab is not faster than in the Java implementation.

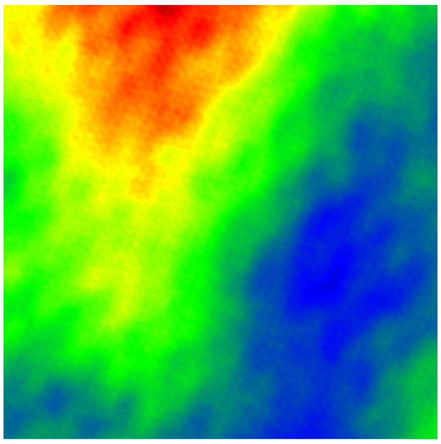
(d) When the simulation procedure uses two parallel threads for the simulation of a field with 0.64 mio. data points or more (i.e. $M \geq 400$), the needed amount of time for the simulation is only 53 to 60 percent of the time needed without using parallel threads. This shows that the simulation of OSSRF is a task which can very efficiently use calculation in parallel threads.

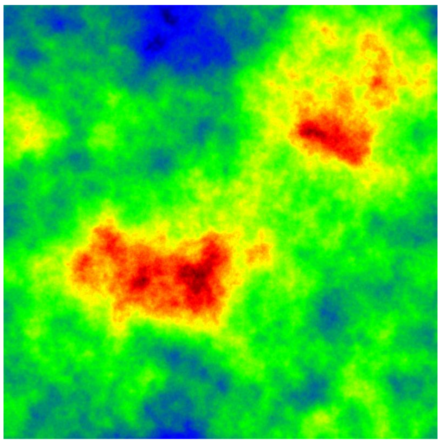
size of the OSSRF		simulation time			simulation time per mio. values		
`M`	mio. values	ha. 1 thr.	ha. 2 thr.	mov. av.	ha. 1 thr.	ha. 2 thr.	mov. av.
100	0.04	0.101	0.085	0.475	2.53	2.13	11.9
200	0.16	0.294	0.202	1.85	1.84	1.26	11.6
800	2.56	4.39	2.56	32.6	1.71	1.00	12.7
2000	16.00	31.4	18.0	278.3	1.96	1.13	17.4
2800	31.36	65.2	38.3	623.4	2.08	1.22	19.9
5600	125.44	297.0	169.6	−−	2.37	1.35	−−
8000	256.00	654.6	364.0	−−	2.56	1.42	−−

Table 6.5: Time (in s.) for the calculation of 2-dim. gaussian OSSRFs (using 32bit floats).

The times for the calculation of some gaussian random fields using 32-bit floating point variables to store the data during simulation are listed in table 6.5. A comparison with the respective times for the 64-bit version (in tables 6.3 and 6.4) shows that the computation for small fields (with about $M < 2000$) is slower when using the 32-bit variant, however for larger OSSRFs (with $M > 2000$) it is faster than the 64-bit version.


(a) Non-gaussian field ($\alpha = 1.5$).


(b) enlarged detail from the center of (a).


(c) Gaussian field ($\alpha = 2.0$).

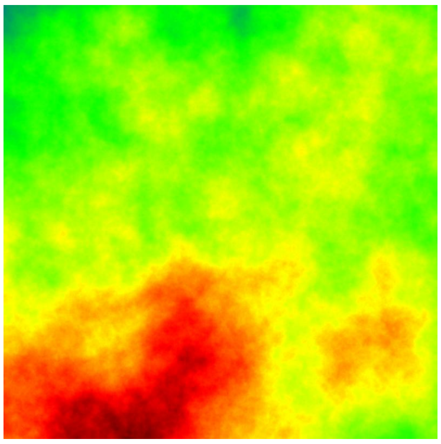
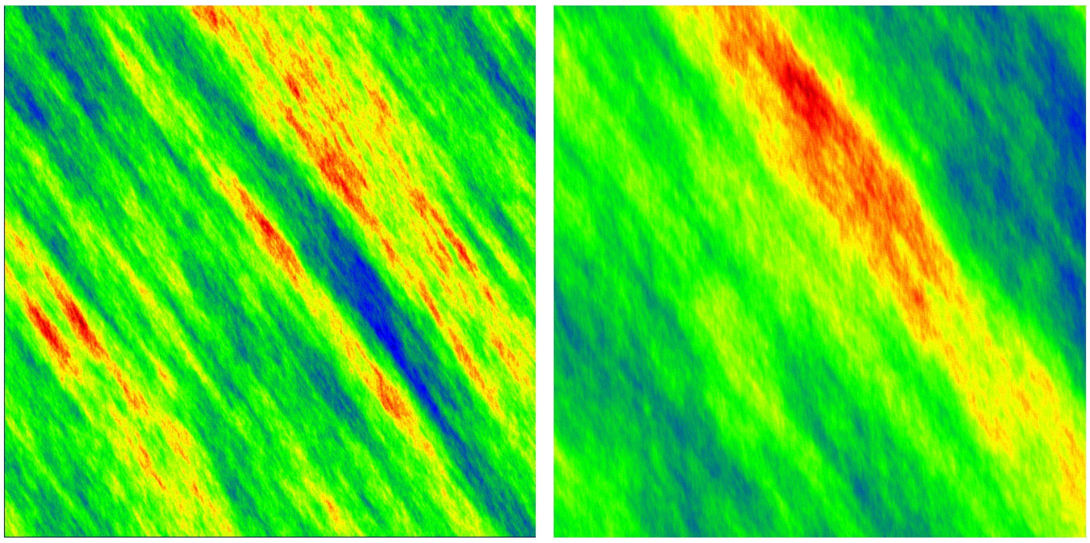
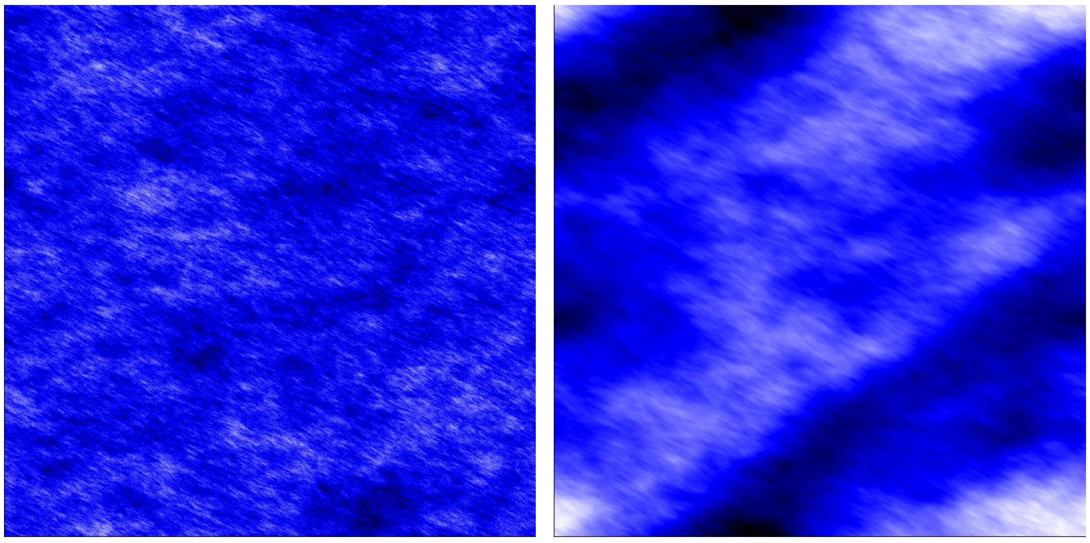

(d) enlarged detail from the center of (c).

Figure 6.7: Some examples of two-dimensional, isotropic, harmonizable OSSRFs with parameter $M = 8000$ (i.e. with 256 mio. simulated values), calculated with the Java program “OSSRFSIM”.



(a) $\alpha = 2.0$, $M = 2800$. (b) enlarged detail from the center of (a).

Figure 6.8: An example of a two-dimensional, anisotropic, moving-average OSSRF, simulated with the Java program “OSSRFSIM”.



(a) $H = 0.10$. (b) $H = 0.99$.

Figure 6.9: Examples of two-dimensional, anisotropic, gaussian ($\alpha = 2.0$), harmonizable OSSRFs with $M = 3200$ (i.e. 41 mio. simulated values), simulated with the Java program “OSSRFSIM”.

6.2.2 Simulation of three-dimensional OSSRF in Java

The algorithms for the simulation of three-dimensional OSSRF in harmonizable or moving-average representation have been implemented in Java in several versions: In order to enable the simulation of large OSSRFs, whose data doesn't fit into the available RAM, versions which cache parts of the data on the hard disk drive (HDD) have also been implemented, in addition to the simpler versions which only use the RAM to store the data. The versions which store the OSSRF data on the HDD are slower, but allow simulation of OSSRFs whose size is not limited by the RAM but rather by the size of the HDD. For each simulated value of the OSSRF, the program needs 24 bytes of free space on the disc if a harmonizable random field is simulated, and 512 bytes if it is a moving average OSSRF. For example, a harmonizable field with parameter $M = 512$ (i.e. with values simulated in 1.074 billion points) can be simulated using 24 GB disc space, and 192 GB free space on the HDD are required for the simulation of a field with parameter $M = 1024$ (and 8.59 billion simulated values). The program decides automatically whether to cache data on the disk or not, depending on the size of the OSSRF (i.e., on the parameter M), and on the amount of RAM which can be used by the program. Figure 6.10 shows some visualizations of a harmonizable OSSRF with parameter $M = 512$ (i.e. with about 1.07 billion simulated values), which was calculated using the HDD.

For both cases - simulation with or without using the HDD to cache data - there are also implementations for the simulation of harmonizable OSSRFs which use parallel threads to accelerate the computations (analogous to the simulation of two-dimensional random fields). Therefore, like for the implementations of the simulation of two-dimensional fields, there are four implementations of the simulation algorithm for harmonizable fields (with or without using the HDD during the simulation process, and with or without using parallel threads), and two different implementations for the simulation of moving average OSSRFs (with or without storing data on the HDD).

We tested the needed computing times for the simulation of OSSRFs in different sizes also for this program module. The observed times are listed in the Tables 6.6 to 6.8. Comparing the numbers in theses tables, we can see that the simulation of a three-dimensional harmonizable OSSRF (without using parallel threads) was slower if intermediate results were written to and read from the HDD than in the case of a calculation of an equally-sized OSSRF without caching on the HDD. More precisely, the observed times with caching were about 23% - 77% longer than without it. In the case of the simulation of a moving-average OSSRF, however, the difference were much more serious: When caching data on the HDD, the simulation needed about three to four times as much time as without. The numbers in the tables also show that the simulation of a harmonizable OSSRF which caches data on the HDD needs about 27% - 30% less time if it is calculated with two parallel threads instead of only one thread.

size of the OSSRF		simulation time			simulation time per mio. values		
`M`	mio. values	ha. $\alpha = 2$	ha. $\alpha < 2$	mov. av.	ha. $\alpha = 2$	ha. $\alpha < 2$	mov. av.
50	1.0	2.12	2.70	28.8	2.12	2.70	28.8
64	2.1	3.03	4.54	35.8	1.44	2.16	17.1
80	4.1	6.86	9.86	93.3	1.67	2.41	22.8
100	8.0	16.2	22.0	238.5	2.03	2.75	29.8
128	16.8	25.0	36.8	−−	1.49	2.19	−−
160	32.8	57.0	79.9	−−	1.74	2.44	−−
200	64.0	133.1	179.0	−−	2.08	2.80	−−
225	91.1	229.4	293.4	−−	2.52	3.22	−−

Table 6.6: Time (in s.) for the calculation of 3-dimensional OSSRF with the Java program (without caching data on the HDD and without parallel threads).

size of the OSSRF		simulation time			simulation time per mio. values		
`M`	mio. values	ha. $\alpha = 2$	ha. $\alpha < 2$	mov. av.	ha. $\alpha = 2$	ha. $\alpha < 2$	mov. av.
100	8.0	25	30	900	3.16	3.80	112.5
128	16.8	44	55	1843	2.63	3.30	109.9
160	32.8	95	115	−−	2.88	3.50	−−
200	64.0	198	242	−−	3.10	3.79	−−
225	91.1	312	377	−−	3.42	4.13	−−
320	262.1	775	979	−−	2.96	3.73	−−
512	1073.7	3373	3801	−−	3.14	3.54	−−

Table 6.7: Time (in s.) for the calculation of 3-dimensional OSSRF with the Java program “OSSRFSIM” (caching data on the HDD; without parallel threads).

size of the OSSRF		simulation time		simulation time per mio. values	
`M`	mio. values	$\alpha = 2$	$\alpha < 2$	$\alpha = 2$	$\alpha < 2$
100	8.0	18	20	2.25	2.50
128	16.8	32	38	1.91	2.27
160	32.8	68	81	2.08	2.47
200	64.0	142	177	2.22	2.77
225	91.1	223	255	2.45	2.80
320	262.1	566	704	2.16	2.69
512	1073.7	2440	2711	2.27	2.52

Table 6.8: Time (in s.) for the calculation of 3-dimensional harmonizable OSSRF with the Java program (caching data on the HDD; with 2 parallel threads).

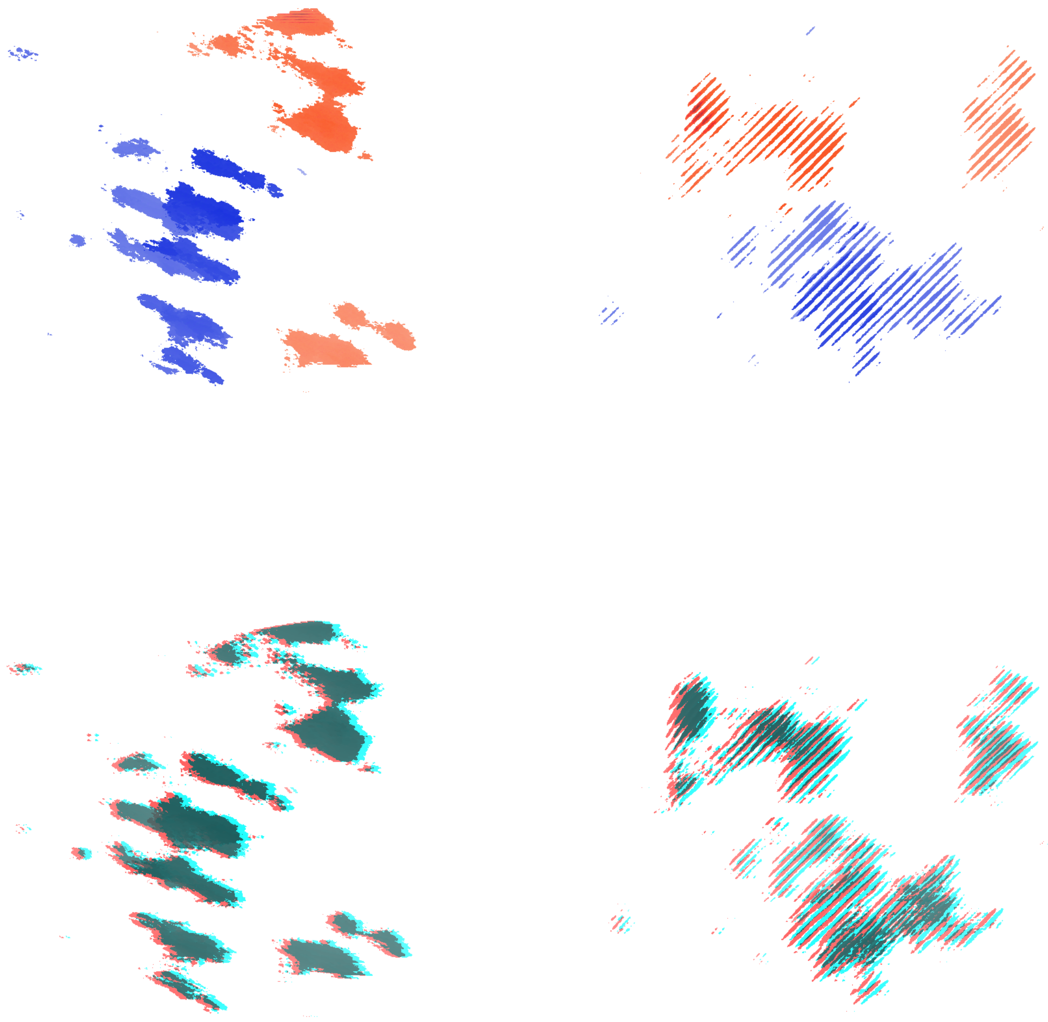


Figure 6.10: Images of a three-dimensional harmonizable OSSRF with parameters $\alpha = 1.5$ and $M = 512$ (i.e. with $2^{30} \approx 1.07$ billion calculated values), simulated with Java. The anaglyph pictures in the second row are intended for observation with red/cyan 3d-glasses.

Chapter 7

Parameter estimation in the harmonizable case

In this chapter, a method for the estimation of some parameters of an OSSRF in harmonizable representation will be presented. This algorithm has been proposed by Prof. H. P. Scheffler.

7.1 Derivation of an estimation algorithm

In order to find a method for the parameter estimation of harmonizable OSSRFs, we first look again at the simulation algorithm for these fields:

(a) Simulation of $(2M)^d$ i.i.d. complex-valued isotropic α-stable random variables (see subsection 5.1.1).

(b) Calculation of $f(\vec{\mathbf{k}}) = \begin{cases} \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}} \text{ for } \xi_{\vec{\mathbf{k}}} = \vec{\mathbf{k}} \cdot D & , \quad \vec{\mathbf{k}} \in J \\ 0 & , \quad \textit{else.} \end{cases}$

(c) Calculation of the products $g_{\vec{\mathbf{k}}} := f(\vec{\mathbf{k}}) \cdot W_\alpha(\Delta_{\vec{\mathbf{k}}})$ for $\vec{\mathbf{k}} \in \{-M, \ldots, M-1\}^d$.

(d) Calculation of $\left(V_\psi^{A,B,D}(x_{\vec{\mathbf{m}}})\right)_{\vec{\mathbf{m}} \in \{-M,\ldots,M-1\}^d} = FFT\left((g_{\vec{\mathbf{k}}})_{\vec{\mathbf{k}} \in \{-M,\ldots,M-1\}^d}\right)$

(e) Calculation of $X_\psi^{A,B,D}(x_{\vec{\mathbf{m}}}) = \text{Re}\left(V_\psi^{A,B,D}(x_{\vec{\mathbf{m}}})\right) - \text{Re}\left(V_\psi^{A,B,D}(\vec{\mathbf{0}})\right)$
for $\vec{\mathbf{m}} \in \{-M, \ldots, M-1\}^d$.

In the following, we consider the case that one or more of the parameters of the function $f(\vec{\mathbf{k}}) = \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}} \cdot 1_J(\vec{\mathbf{k}})$ (i.e. one or more of the parameters $C_1, \ldots, C_d, \lambda_1, \ldots, \lambda_d$, $\theta_1, \ldots, \theta_d, \alpha, H, \rho$) are unknown and should be estimated from a simulated OSSRF (while the remaining parameters are known), and we assume that θ is the vector of unknown parameters of this function.

Let us first assume that the intermediate result $\left(g_{\vec{\mathbf{k}}}\right)_{\vec{\mathbf{k}}\in J}$ of the simulation were known. According to step 3 in the algorithm, $g_{\vec{\mathbf{k}}} := f(\vec{\mathbf{k}}) \cdot W_\alpha(\Delta_{\vec{\mathbf{k}}})$ which implies $\log(|g_{\vec{\mathbf{k}}}|) = \log(|f(\vec{\mathbf{k}})|) + \log(|W_\alpha(\Delta_{\vec{\mathbf{k}}})|)$, for $\vec{\mathbf{k}} \in J$. Thereby, $(f(\vec{\mathbf{k}}))$, for $\vec{\mathbf{k}} \in J$, is a sample of values of the function f, some of whose parameters are to be estimated, and $(\log(|W_\alpha(\Delta_{\vec{\mathbf{k}}})|)_{\vec{\mathbf{k}}\in J}$ are i.i.d. random variables. Thus, the parameter vector θ can be estimated by searching for a vector θ for which the corresponding function $\log(|f_\theta(\vec{\mathbf{k}})|)$ approximates $\log(|g_{\vec{\mathbf{k}}}|) - E(\tilde{W})$ (where $\tilde{W}$ is a random variable with the same distribution as the $\log(|W_\alpha(\Delta_{\vec{\mathbf{k}}})|)$. This means, the searched parameter vector is estimated by the vector θ which minimizes the sum of squares $\bar{S} := \sum_{\vec{\mathbf{k}}\in J} \left(\log(|f_\theta(\vec{\mathbf{k}})|) - (\log(|g_{\vec{\mathbf{k}}}|) - E(\tilde{W}))\right)^2$. As the expectation of $E(\tilde{W})$ can't be calculated easily, it is estimated, using the sample, by

$$\begin{aligned}\frac{1}{|J|} \cdot \sum_{\vec{\mathbf{k}}\in J} \log(|W_\alpha(\Delta_{\vec{\mathbf{k}}})|) &= \frac{1}{|J|} \cdot \sum_{\vec{\mathbf{k}}\in J} \left(\log(|g_{\vec{\mathbf{k}}}|) - \log(|f_\theta(\vec{\mathbf{k}})|)\right) \\ &= \frac{1}{|J|} \cdot \sum_{\vec{\mathbf{k}}\in J} \log(|g_{\vec{\mathbf{k}}}|) - \frac{1}{|J|} \cdot \sum_{\vec{\mathbf{k}}\in J} \log(|f_\theta(\vec{\mathbf{k}})|),\end{aligned}$$

i.e. by the difference of the means of $\log(|g_{\vec{\mathbf{k}}}|)$ and of $\log(|f_\theta(\vec{\mathbf{k}})|)$. Thus, our estimator of θ is the parameter vector $\hat{\theta}$ which minimizes

$$S := \sum_{\vec{\mathbf{k}}\in J} \left(\log\left(|f_\theta(\vec{\mathbf{k}})|\right) - \log(|g_{\vec{\mathbf{k}}}|) + \frac{1}{|J|} \cdot \sum_{\vec{\mathbf{j}}\in J} \log(|g_{\vec{\mathbf{j}}}|) - \frac{1}{|J|} \cdot \sum_{\vec{\mathbf{j}}\in J} \log\left(|f_\theta(\vec{\mathbf{j}})|\right) \right)^2 .$$

In the previous paragraph, the estimation was derived under the assumption that the values of $\left(g_{\vec{\mathbf{k}}}\right)_{\vec{\mathbf{k}}\in J}$ are known, which usually is not true. If instead of this values, the values of $V_\psi^{A,B,D}(x_{\vec{\mathbf{m}}}) = FFT(g_{\vec{\mathbf{k}}})$ are known, then the $g_{\vec{\mathbf{k}}}$ can be calculated from the $V_\psi^{A,B,D}(x_{\vec{\mathbf{m}}})$ by an inverse fast Fourier transform, as the Fourier transform is invertible. Usually also the values of $V_\psi^{A,B,D}(x_{\vec{\mathbf{m}}})$ are unknown, and instead of them the values of the simulated (or observed) OSSRF $X_\psi^{A,B,D}$, which are $X_\psi^{A,B,D}(x_{\vec{\mathbf{m}}}) = \mathrm{Re}\left(V_\psi^{A,B,D}(x_{\vec{\mathbf{m}}})\right) - \mathrm{Re}\left(V_\psi^{A,B,D}(\vec{\mathbf{0}})\right)$, have to be considered as the basis for the estimation. However, this doesn't really make a difference:

- If a complex number v is written in polar representation, i.e. as $v = |v| \cdot e^{i\zeta}$, then

$$\mathrm{Re}(v) = |v| \cdot \cos(\zeta) \Rightarrow |\mathrm{Re}(v)| = |v| \cdot |\cos(\zeta)| \Leftrightarrow |v| = \frac{|\mathrm{Re}(v)|}{|\cos(\zeta)|}.$$

In the case of $v = V_{\psi}^{A,B,D}(x_{\vec{\mathbf{m}}})$, ζ is is a random value, and thus also $|\cos(\zeta)|$ is random. Therefore, considering only the real part of $V_{\psi}^{A,B,D}(x_{\vec{\mathbf{m}}})$ can be seen as multiplication with a random variable $Z = |\cos(\zeta)|$, which is bounded to the interval $[0,1]$. Particularly, $|v| \geq |\mathrm{Re}(v)|$. To compensate the change in magnitude of the absolute values, it seems appropriate to multiply the input data $\mathrm{Re}\left(V_{\psi}^{A,B,D}(x_{\vec{\mathbf{m}}})\right)$ with the constant $E\left(\frac{1}{Z}\right)$ before applying the inverse Fourier transform, or applying this multiplication with $E\left(\frac{1}{Z}\right)$ after the transform (which is equivalent because of the linearity of the Fourier transform). However, it is not necessary to explicitely perform this multiplication with the expectation value, as it is already included in the step of adding the difference between the means of $\log(|g_{\vec{\mathbf{k}}}|)$ and of $\log(|f_{\theta}(\vec{\mathbf{k}})|)$.

- Adding the same (constant) value to each input value $(v_{\vec{\mathbf{m}}})_{\vec{\mathbf{m}}\in J}$ of a FFT or of an inverse FFT doesn't change the result of the output, except in the origin (i.e. if the index is $\vec{\mathbf{k}} = \vec{\mathbf{0}}$), which is not included in the set J, but belongs to the "cut out" center area, and therefore isn't considered anyways.

  For an inverse Fourier transform of an one-dimensional vector, it is easy to see that only the first element of the resulting vector (i.e. the element with index 0) changes, if every element of the input vector is changed by adding a $c \in \mathbb{R}$:

$$\begin{aligned}(IFFT(\vec{\mathbf{v}} + c))_k &= \sum_{m=0}^{N-1} e^{2\pi i \frac{km}{N}} \cdot (v_m + c) \\ &= \sum_{m=0}^{N-1} e^{2\pi i \frac{km}{N}} \cdot v_m + \sum_{m=0}^{N-1} e^{2\pi i \frac{km}{N}} \cdot c \\ &= (IFFT(\vec{\mathbf{v}}))_k + c \cdot \sum_{m=0}^{N-1} e^{2\pi i \frac{km}{N}} \\ &= (IFFT(\vec{\mathbf{v}}))_k + \begin{cases} c \cdot N, & \text{if} \quad k = 0 \\ 0, & \text{else} \end{cases}\end{aligned}$$

  Thus substracting $\mathrm{Re}\left(V_{\psi}^{A,B,D}(\vec{\mathbf{0}})\right)$ only changes the value of the inverse FFT at $g_{\vec{\mathbf{0}}}$, and therefore doesn't influence the result of the estimation.

Following from these considerations, the estimation of some parameters of the function $f(\vec{\mathbf{k}}) = \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{q}{\alpha}} \cdot 1_J(\vec{\mathbf{k}})$, based on the values $X_{\psi}^{A,B,D}(x_{\vec{\mathbf{m}}})$ of an (observed or simulated) OSSRF in harmonizable representation, can be achieved by the following algorithm:

(a) Calculate $\left(\tilde{g}_{\vec{\mathbf{k}}}\right)_{\vec{\mathbf{k}}\in\{-M,\dots,M-1\}^d}$ by performing an inverse FFT on the input field $\left(X_{\psi}^{A,B,D}(x_{\vec{\mathbf{m}}})\right)_{\vec{\mathbf{m}}\in\{-M,\dots,M-1\}^d}$

(b) Set $g_{\vec{\mathbf{k}}} := \tilde{g}_{\vec{\mathbf{k}}} \cdot 1_J(\vec{\mathbf{k}})$, for $\vec{\mathbf{k}} \in \{-M, \dots, M-1\}^d$

(c) Search the parameter vector θ which minimizes the sum

$$S := \sum_{\vec{\mathbf{k}} \in J} \left(l_\varepsilon\left(f_\theta(\vec{\mathbf{k}})\right) - l_\varepsilon(g_{\vec{\mathbf{k}}}) + \frac{1}{|J|} \cdot \sum_{\vec{\mathbf{j}} \in J} l_\varepsilon(g_{\vec{\mathbf{j}}}) - \frac{1}{|J|} \cdot \sum_{\vec{\mathbf{j}} \in J} l_\varepsilon\left(f_\theta(\vec{\mathbf{j}})\right) \right)^2$$

with $l_\varepsilon(\cdot) = \log(\max\{|\cdot|, \varepsilon\})$ for an appropriate (small) $\varepsilon > 0$, e.g. $\varepsilon = e^{-40}$. Thereby, an absolute value of zero is set to the positive value ε before applying the log-function, in order to avoid a function value of $-\infty$.

In the remaining part of this chapter, we show that this algorithm indeed can be used to estimate some unknown parameters of the function f.

7.2 Implementation in Matlab

In this section, an implementation of this estimation method is presented, which tests the estimation of the five parameters α, λ_1, λ_2, v_1 and v_2 for a two-dimensional OSSRF in harmonizable representation (where v_1 and v_2 are the "angles" of the vectors θ_1 and θ_2, i.e. $\theta_k = (\cos(v_k), \sin(v_k))^T,\ k \in \{1, 2\}$). Procedures which estimate certain other combinations of parameters can be developed analogously.

Listing 7.1: A program for the estimation of several parameters of an OSSRF)

```
% set parameter values
%   parameters which will be estimated
alpha = 2.0;     % shape parameter (index of stability)
a1 = 1;          % parameter \lambda_1 (eigenvalue of E)
a2 = 2;          % parameter \lambda_2 (eigenvalue of E)
v1 = 1.0;        % 'angle' of vector \theta_1 (\theta_1 = exp(i*v1) )
v2 = 2.57;       % 'angle' of vector \theta_2 (\theta_2 = exp(i*v2) )
%   other parameters (which are assumed to be known during
%   the estimation process)
H = 1.0 ;
A = 4.0;
N = 2;
C1 = 1;
C2 = 1;
rho = 2;
M = 512;        % Parameter M: determines the size of the random field

iM = 10;          % index of row in stat. tables
loopN = 20;      % number of test runs

```

```
% multiplication of vector 'param' with 'mat_switch'
% exchanges a1 with a2 and v1 with v2
% (i.e. switches indexing of 'eigenvalues' and 'eigenvectors')
mat_switch = [0 1 0 0 0; 1 0 0 0 0; 0 0 0 1 0; 0 0 1 0 0; 0 0 0 0 1];

% table to which the statistical values are saved
deviations = zeros(loopN, 5);

% confirm that a1 <= a2   (if not: exchange them)
if (a1>a2) tmp = a1; a1 = a2; a2 = tmp; clear ('tmp'); end

disp(' ');
disp(' ');
disp(sprintf('Test estimation with parameter M=%d ...', M));

tic;
tocloopstart = toc;

for loopi = 1:loopN

    toc1 = toc;

    % simulation of OSSRF whose parameters should be estimated
    W =gen2Dharmo( alpha, H, A, M, N, a1, a2, v1, v2, C1, C2, rho);

    % inverse FFT
    W = fftshift(W, 1);
    W = fftshift(W, 2);
    W = ifft2(W);
    W = fftshift(W, 1);
    W = fftshift(W, 2);

    % calculation of logAW and meanLogAW as basis for the estimation
    AW = abs(W);
    AW( -N+M+1 : N+M, -N+M+1 : N+M) = 0;
    logAW = max(log(AW), -40);
    meanLogW = mean(mean(logAW));

    %define function 'sum of squares of differences'
    % (param = [a1, a2, v1, v2, alpha] )
    diffSSum = @(param) ( diffSqrSum(param, H, A, M, N, C1, C2, ...
                            rho, meanLogW, logAW) );

    % search a start value (for the parameter vector)
    % (try several randomly chosen param. vectors,
    %  and choose the one with the smallest sum of squares)
    startParam = rand(1,5) .* [3.0 3.0 pi pi 2.0];
    if (startParam(1)>startParam(2))
        startParam = startParam * mat_switch;
    end
    startSSE = diffSSum (startParam);
    for k = 2:10
        testParam = rand(1,5) .* [3.0 3.0 pi pi 2.0];
        if (testParam(1)>testParam(2))
            testParam = testParam * mat_switch;
        end;
        testSSE = diffSSum (testParam);
        if (testSSE < startSSE)
            startParam = testParam;
            startSSE = testSSE;
        end;
    end;

    % estimation of the parameter vector, using 'fminunc'
    y = fminunc(diffSSum, startParam, optimset('GradObj','off', ...
                     'LargeScale', 'off', 'Display', 'notify'));

    % Modification of result for output
    % (a)  y(1) must be less or equal to y(2)
    if (y(1)>y(2)) y = y * mat_switch; end;

    % (b)  y(3) and y(4) can be chosen such thath error is max. pi/2
    %      (because of periodicity)
    while (y(3)<v1-pi/2)  y(3)=y(3)+pi; end;
    while (y(3)>v1+pi/2)  y(3)=y(3)-pi; end;
    while (y(4)<v2-pi/2)  y(4)=y(4)+pi; end;
    while (y(4)>v2+pi/2)  y(4)=y(4)-pi; end;

    deviations(loopi, :) = [( y(1)-a1), ( y(2)-a2), ( y(3)-v1), ...
                              ( y(4)-v2), ( y(5)-alpha)];

    toc2 = toc;              % measure time at the end of the loop
    disp(sprintf( ...
     'Estimation of OSSRF nr. %d (of %d) finished (in %g s.) ... ', ...
     loopi,loopN, toc2-toc1 ));

end; % end loop

tocloopstop = toc;

disp(' ');

timesN(iM) = tocloopstop-tocloopstart;
times1(iM) = timesN(iM)/loopN;
disp(sprintf('Time for %d loops: %g s.;   time per loop: %g s.', ...
                 loopN, timesN(iM), times1(iM)  ));

disp(' ');
disp(' ');
disp('deviations:  ');
```

```
121 disp('            a1        a2        v1        v2      alpha ');
122 for loopj = 1 : loopN
123     disp(sprintf('       %9.4f  %9.4f  %9.4f  %9.4f  %9.4f', ...
124                   deviations(loopj, 1), ...
125                   deviations(loopj, 2), deviations(loopj, 3), ...
126                   deviations(loopj, 4), deviations(loopj, 5) ));
127 end;
128 
129 disp('----------------------------------------------------------------');
130 
131 Ms(iM) = M;
132 mse(iM, :) = mean(deviations.^2);
133 mean_ad(iM, :) =  mean(abs(deviations));
134 median_ad(iM, :) = median(abs(deviations));
135 
136 mean_bs(iM, :) = mean(deviations);
137 median_bs(iM, :) = median(deviations);
138 
139 std_s(iM, :) = std(deviations);
140 iqr_s(iM, :) = iqr(deviations);
141 
142 disp(sprintf('   MSE: %9.4f  %9.4f  %9.4f  %9.4f  %9.4f', ...
143         mse(iM, 1), mse(iM, 2), mse(iM, 3), mse(iM, 4), mse(iM, 5)));
144 disp(sprintf('MEANAD: %9.4f  %9.4f  %9.4f  %9.4f  %9.4f', ...
145         mean_ad(iM, 1), mean_ad(iM, 2), mean_ad(iM, 3), ...
146         mean_ad(iM, 4), mean_ad(iM, 5)));
147 disp(sprintf(' MEDAD: %9.4f  %9.4f  %9.4f  %9.4f  %9.4f', ...
148         median_ad(iM, 1), median_ad(iM, 2), median_ad(iM, 3), ...
149         median_ad(iM, 4), median_ad(iM, 5)));
150 disp(sprintf('  BIAS: %9.4f  %9.4f  %9.4f  %9.4f  %9.4f', ...
151         mean_bs(iM, 1), mean_bs(iM, 2), mean_bs(iM, 3), ...
152         mean_bs(iM, 4), mean_bs(iM, 5)));
153 disp(sprintf('MDBIAS: %9.4f  %9.4f  %9.4f  %9.4f  %9.4f', ...
154         median_bs(iM, 1), median_bs(iM, 2), median_bs(iM, 3), ...
155         median_bs(iM, 4), median_bs(iM, 5)));
156 disp(sprintf('   STD: %9.4f  %9.4f  %9.4f  %9.4f  %9.4f', ...
157         std_s(iM, 1), std_s(iM, 2), std_s(iM, 3), ...
158         std_s(iM, 4), std_s(iM, 5)));
159 disp(sprintf('   IQR: %9.4f  %9.4f  %9.4f  %9.4f  %9.4f', ...
160         iqr_s(iM, 1), iqr_s(iM, 2), iqr_s(iM, 3), ...
161         iqr_s(iM, 4), iqr_s(iM, 5)));
```

The main part of this example program is the loop which starts in line 39 and ends in line 107. In every instance of the loop, an OSSRF is simulated with a set of parameters which has been set before, and the five parameters α, λ_1, λ_2, v_1 and v_2 are estimated from the simulated field. In order to evaluate the quality of the estimation, the deviations of the estimated values from the predefined parameter values which have been used in the simulation are saved in a table. In the code after this loop, these deviations are displayed and some statistical measures are calculated.

The loop begins with the simulation of a two-dimensional harmonizable OSSRF in line 44. The next step of the calculations is the inverse Fourier transform of these simulated random values (lines 47 - 51). Following the algorithm presented in the previous section, the searched parameters are estimated (in a simplified representation) by fitting the logarithm of the function f to the logarithm of the absolute values resulting from this inverse Fourier transform. Therefore, the logarithms of the absolute values of the results from the inverse Fourier transform (and also their mean value) are calculated in the next steps (lines 54-57) and saved in the array `logAW`. The logarithms of function values of f are fitted to these values by searching the vector `param` of the five parameters, which minimizes the function

$$S := \sum_{\vec{\mathbf{k}}\in J} \left(l_\varepsilon\left(f_\theta(\vec{\mathbf{k}})\right) - l_\varepsilon(g_{\vec{\mathbf{k}}}) + \frac{1}{|J|}\cdot \sum_{\vec{\mathbf{j}}\in J} l_\varepsilon(g_{\vec{\mathbf{j}}}) - \frac{1}{|J|}\cdot \sum_{\vec{\mathbf{j}}\in J} l_\varepsilon\left(f_\theta(\vec{\mathbf{j}})\right)\right)^2$$

(compare the previous section). This function is defined as a function of the vector `param` (in line 61f.) using the function `diffSqrSum` whose implementation is listed below. The minimum of this function is searched with the Matlab function `fminunc` (line 85f.), which requires as input not only the function that should be minimized (i.e. S), but also an initial parameter vector. Therefore, an initial parameter vector for the minimizing process is chosen before (in the lines 67 to 82) by testing some random vectors, and selecting the one with the smallest sum of squares. After estimating the parameter values, the deviations of the estimations from the true values of the parameters are calculated (in line 99f.).

After repeating the process of simulation of an OSSRF and estimation of the parameters α, λ_1, λ_2, v_1 and v_2 several times (in this example: 20 times), and thus obtaining a sample of 20 deviations between estimated value and true value of each parameter, several statistics of this deviations are calculated (lines 132-140) and displayed (lines 142-161) for each parameter, in order to evaluate the quality of the estimation.

Listing 7.2: The function 'diffSqrSum' (a modified sum of squares of deviations)

```
% The parameter 'param' is a vector of length 5:
% param= [a1 a2 v1 v2 alpha]    - the parameters of phi
%                                 which are to be estimated
% H, A, M, N, C1, C2, rho       - the other parameters of phi
%                                 (which are supposed to be known)
% meanLogAW                     - the mean value of logAW
% logAW                         - the sample for which the parameters of
%                                 phi should be estimated
function fval=diffSqrSum(param, H, A, M, N, C1, C2, rho, meanLogAW, logAW)

    a1 = param(1);
    a2 = param(2);
    v1 = param(3);
    v2 = param(4);
    alpha = param(5);

    % create the grid ( with 2M x 2M data points)
    [km,lm]=meshgrid(-A:A/M:A-A/M,-A:A/M:A-A/M);

    % calculate  log(phi) with the given parameters
    phi_exponent = (-H-(a1+a2)/alpha)/rho;
    phi_base = C1*(abs(km*cos(v1) + lm*sin(v1))).^(rho/a1) ...
             + C2*(abs(km*cos(v2) + lm*sin(v2))).^(rho/a2);
    logphi =  phi_exponent .* log( phi_base );
    logphi( M-N+1 : M+N, M-N+1 : M+N) = -40;

    % calculate the mean value of log(phi)
    meanLogPhi = mean(mean(  logphi   ));

    % calculate the difference of log(phi) and logAW,
    % minus its mean value,
    % i.e.      (  logphi - mean(mean(logphi)) )
    %          - (  logAW  - mean(mean(logAW))  )
    diffm =  logphi +  meanLogAW - meanLogPhi - logAW ;
    diffm( M-N+1 : M+N, M-N+1 : M+N) = 0;

    % calculate the sum of squares
    fval =   sum(sum( diffm.^2 ))   ;
end
```

This implementation uses the Matlab function `fminunc` in order to find a minimum of the sum of squares `diffSSum`. In a first version, the function `fminsearch` was used for this task, but Prof. H. P. Scheffler suggested to replace it by `fminunc`, because this function is faster. Indeed, tests comparing these two methods on the same example random fields showed that `fminunc` and `fminsearch` found estimations of the searched parameters with similar precisions, but usually `fminunc` needed considerably less time for this task.

For example, the simulation and estimation (both methods applied to the same random fields) of 10 random fields with parameter $M = 200$ (e.g. with 160000 data elements) resulted in the sums of absolute deviations which are given in table 7.1. Here the sum of absolute deviations for an estimation means the sum - over the different estimated parameters - of absolute values of differences between estimated and true value of the corresponding parameter, or: $s = |\hat{a}_1 - a_1^*| + |\hat{a}_2 - a_2^*| + |\hat{v}_1 - v_1^*| + |\hat{v}_2 - v_2^*| + |\hat{\alpha} - \alpha^*|$, where $\hat{a}_1$ is the estimated value and a_1^* is the actual value of the parameter a_1, etc.

In this example, both methods gave estimates with similar deviations from the true parameter values (with average error of the `fminunc` method being a bit better than the one of the `fminsearch` method), but the `fminunc` function usually calculated this

fminsearch	fminunc
0.063113	0.063926
0.063878	0.063741
0.056464	0.055868
0.019942	0.020176
0.055384	0.055667
0.025237	0.024853
0.068784	0.061975
0.025333	0.025504
0.080791	0.078862
0.081495	0.075756
mean values:	
0.054042	0.052633

(a) sums of estimation errors

fminsearch	fminunc
73.6	32.8
85.9	36.6
53.9	34.2
32.3	42.3
38.6	25.3
43.9	36.7
55.9	24.0
70.2	34.1
63.8	36.8
81.1	38.6
mean values:	
59.9	34.1

(b) times for search for minimum (in seconds)

Table 7.1: Sums of absolute approximation errors, and elapsed times of estimation process (Comparison of `fminsearch` and `fminunc`).

results much faster than the `fminsearch` function: The average time for the calculation was 59.9 seconds when using `fminsearch`, compared to only 34.1 seconds with `fminunc`.

7.3 Numerical study

In order to test and to demonstrate the accuracy of the presented estimation procedure, it was applied to a series of simulated random fields, and the estimated parameter values were compared to the parameter values which were used to simulate the field. For several different values of the parameter `M` (i.e. for different sizes of the simulated field), a series of 20 harmonizable OSSRFs with the same combination of parameters each (here: `alpha`= 2.0, `a1`= 1.0, `a2`= 2.0, `v1`= 1.0, `v2`= 2.0, `H`= 1.0, `A`= 4.0, `N`= 2, `C1`= 1.0, `C2`= 1.0 and `rho`= 2.0) was simulated, and the parameters `a1`, `a2`, `v1`, `v2` and `alpha` were estimated by the program from the previous section.

In this section, be n the number of simulated random fields (in this example: $n = 20$), α^* be the actual value of the parameter α which has been used for the simulation of these random fields, and $\hat{\alpha}_k$ denotes the value of α which has been estimated from the kth simulated random field ($1 \leq k \leq n$). The deviation of α for the kth simulation is $\delta_k^\alpha := \hat{\alpha}_k - \alpha^*$ and the absolute deviation is $|\delta_k^\alpha| = |\hat{\alpha}_k - \alpha^*|$. For the other estimated parameters, these notations are analogous. From the sample of 20 deviations, the following statistics are calculated by the estimation program (here presented for the parameter α, for the other parameters analogous):

- the MSE (mean squared error): $\mathrm{mse}(\alpha) := \frac{1}{n}\sum_{k=1}^{n}(\delta_k^\alpha)^2$
- the mean absolute deviation: $\mathrm{mean\_ad}(\alpha) := \frac{1}{n}\sum_{k=1}^{n}|\delta_k^\alpha|$
- the median absolute deviation: $\mathrm{median\_ad}(\alpha) := \mathrm{median}(\{|\delta_1^\alpha| \ldots, |\delta_n^\alpha|\})$
- the mean bias: $\mathrm{mean\_bs}(\alpha) := \frac{1}{n}\sum_{k=1}^{n}\delta_k^\alpha = \left(\frac{1}{n}\sum_{k=1}^{n}\hat{\alpha}_k\right) - \alpha^*$
- the median bias: $\mathrm{median\_bs}(\alpha) := \mathrm{median}_{1\le k\le n}(\delta_k^\alpha) = \mathrm{median}_{1\le k\le n}(\hat{\alpha}_k) - \alpha^*$
- the sample standard deviation of the deviations (which is the same as the sample standard deviation of the estimated values)
- the interquartile range, i.e the difference between the upper and the lower quartile, of the deviations, which is also the same as the interquartile range of the estimated values.

The figures 7.1 and 7.2 show boxplots of the deviations of the 20 estimations for different values of M (i.e., for different sizes of the simulated random fields). These plots demonstrate that the estimation returns quite useful results (with absolute deviations below 0.1 in most cases) already for relative small random fields, and that the accuracy of the estimation improves with increasing values of M. Speaking more precisely, the plots indicate that the magnitude of the absolute deviations is proportional to M^{-1}, and therefore converges to zero for $M \to \infty$. In other words, the observed results give empirical evidence to the assumption that the estimated parameter values converge towards the parameter values which were used to simulate the fields, when $M \to \infty$, i.e. that the estimator is consistent (a theoretical proof of this assumption is a possible topic for further research). Indeed, the calculated statistics of the deviations, like e.g. the mean absolute deviation, the mean bias or the interquartile range of the deviations, seem to be approximately proportional to M^{-1}, and the mean squared error, being defined as the mean of the squares of the deviations, is proportional to M^{-2}. As examples for these statistics, the values of the MSE and of the mean absolute deviation for different parameter values are presented in the graphs in figures 7.3 and 7.4 and in the tables 7.2 and 7.3.

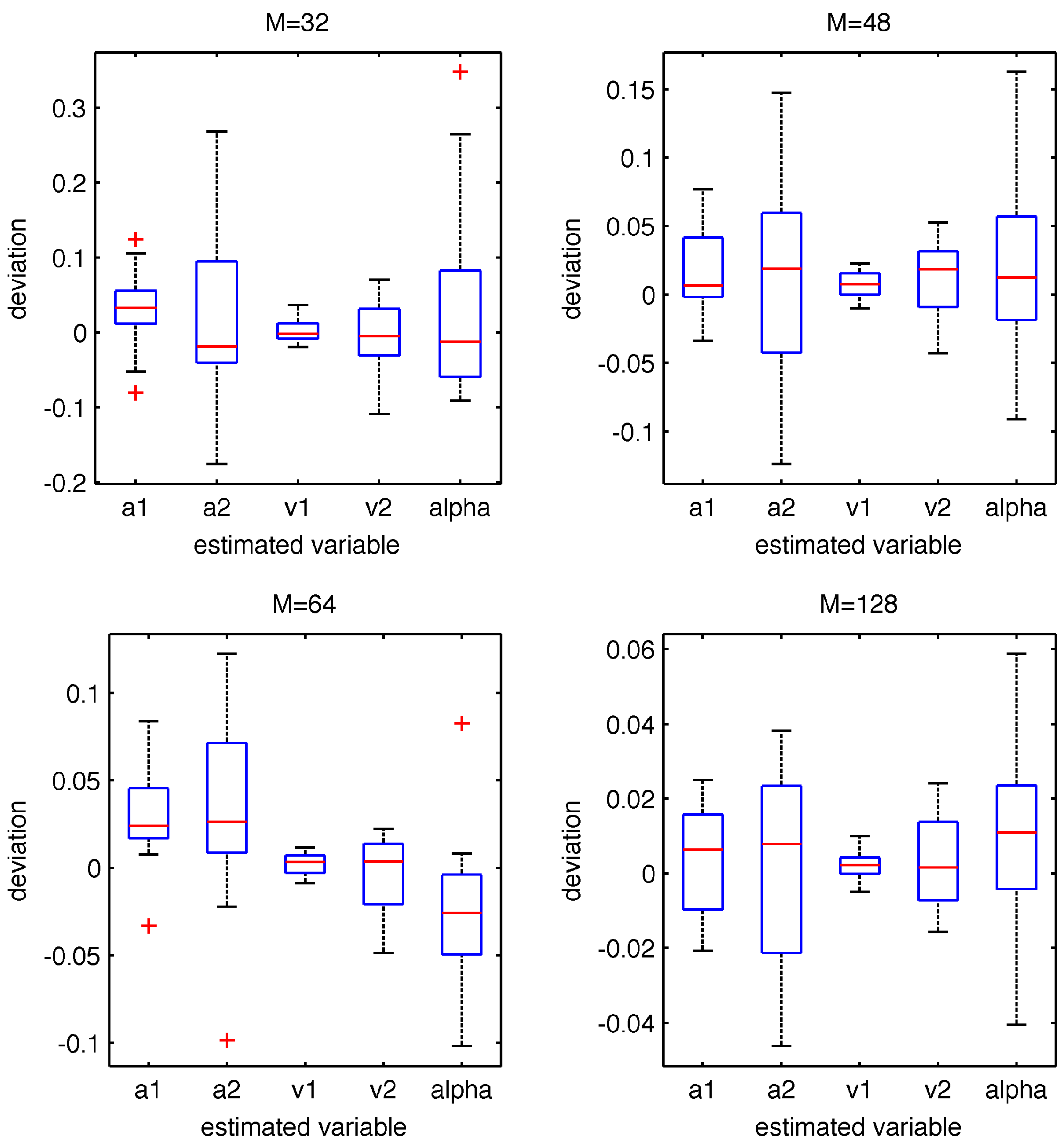


Figure 7.1: Boxplots of deviations of estimated parameter values.

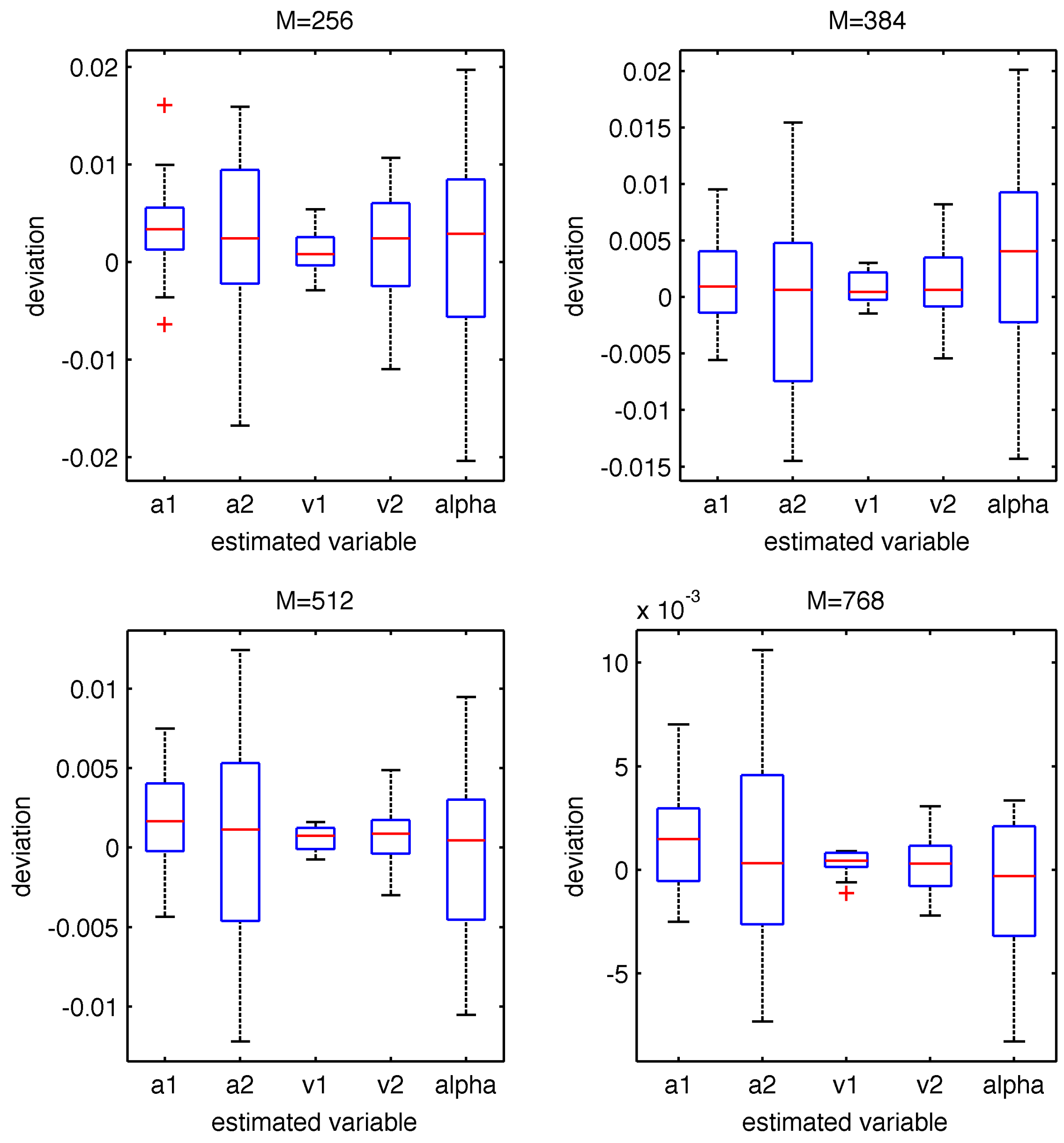


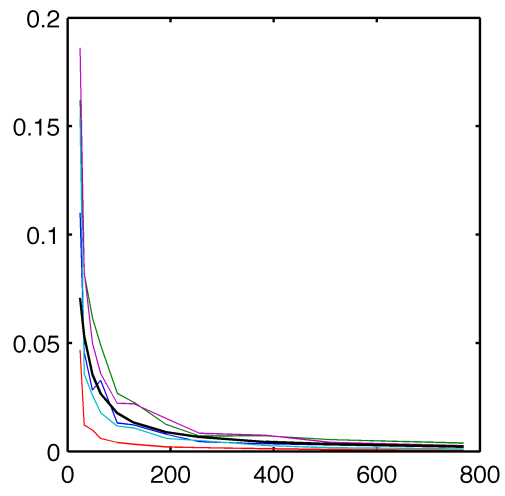
Figure 7.2: Boxplots of deviations of estimated parameter values (for large fields).

size of the random field		mean absolute deviation					
M	elements ($= 4M^2$)	a1	a2	v1	v2	alpha	mean
24	2304	0.1101	0.1620	0.0469	0.1537	0.1861	0.1317
32	4096	0.0457	0.0823	0.0123	0.0358	0.0829	0.0518
48	9216	0.0283	0.0616	0.0098	0.0259	0.0498	0.0351
64	16384	0.0327	0.0489	0.0060	0.0178	0.0358	0.0283
96	36864	0.0131	0.0269	0.0041	0.0117	0.0222	0.0156
128	65536	0.0123	0.0227	0.0033	0.0108	0.0220	0.0142
192	147456	0.0078	0.0123	0.0021	0.0060	0.0151	0.0087
256	262144	0.0044	0.0071	0.0018	0.0050	0.0084	0.0053
384	589824	0.0035	0.0072	0.0013	0.0027	0.0075	0.0044
512	1048576	0.0030	0.0054	0.0008	0.0017	0.0041	0.0030
768	2359296	0.0021	0.0039	0.0005	0.0012	0.0027	0.0021

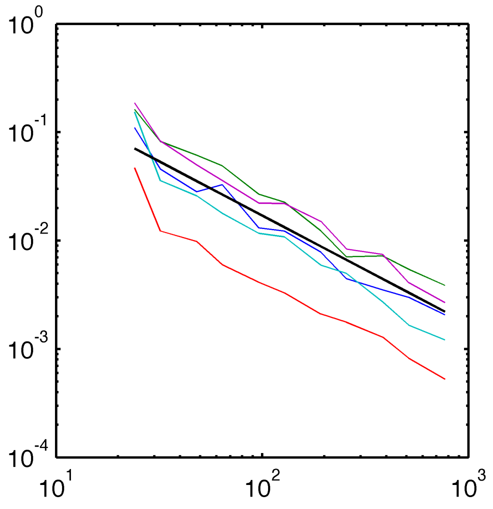
Table 7.2: Mean absolute deviation depending on the size of the random field.

size of the r. field		mean squared error					
M	elements	a1	a2	v1	v2	alpha	mean
24	2304	0.056073	0.053518	0.007040	0.081894	0.089297	0.057564
32	4096	0.003167	0.011863	0.000247	0.001892	0.013551	0.006144
48	9216	0.001345	0.005186	0.000136	0.000894	0.004608	0.002434
64	16384	0.001438	0.003878	0.000046	0.000504	0.002187	0.001611
96	36864	0.000281	0.001202	0.000024	0.000205	0.000878	0.000518
128	65536	0.000190	0.000710	0.000018	0.000162	0.000737	0.000363
192	147456	0.000105	0.000311	0.000008	0.000059	0.000344	0.000165
256	262144	0.000032	0.000077	0.000005	0.000036	0.000103	0.000051
384	589824	0.000020	0.000074	0.000003	0.000012	0.000085	0.000039
512	1048576	0.000013	0.000044	0.000001	0.000004	0.000026	0.000018
768	2359296	0.000007	0.000022	0.000000	0.000002	0.000011	0.000008

Table 7.3: Mean squared error (MSE) depending on the size of the random field.

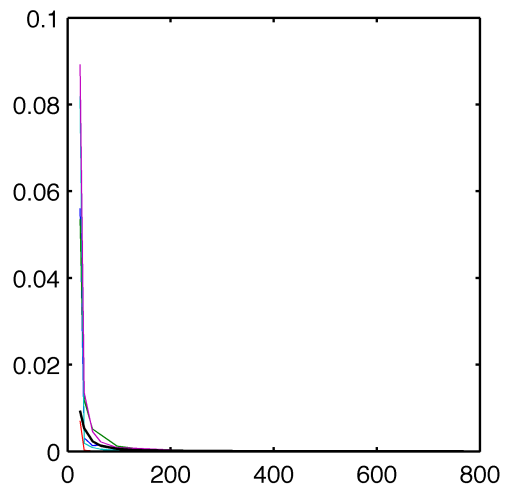


(a) In a cartesian coordinate system.

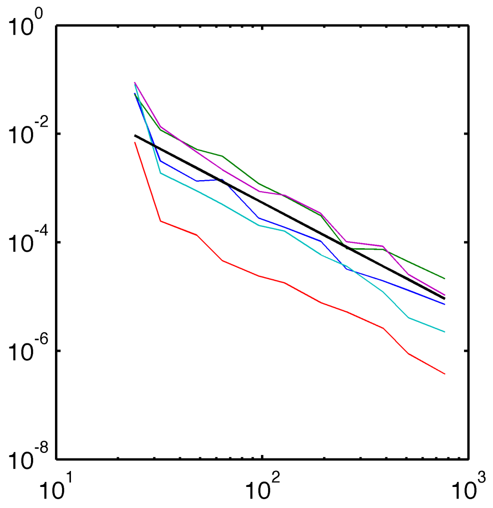


(b) In a log-log coordinate system.

Figure 7.3: The mean absolute deviation in dependency of the parameter M. The five thin lines show the median absolute deviations for the different estimated parameters. The thick, black line is the graph of $h(M) = 1.7 \cdot M^{-1}$.



(a) In a cartesian coordinate system.



(b) In a log-log coordinate system.

Figure 7.4: The mean squared error in dependency of the parameter M. The five thin lines show the MSE for the different estimated parameters. The thick, black line is the graph of $h(M) = 5.4 \cdot M^{-2}$.

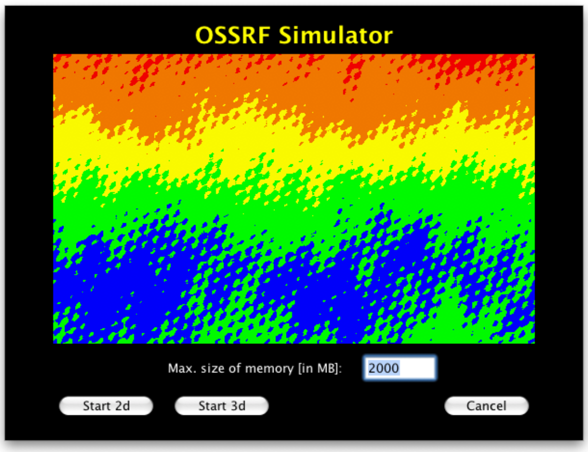
Appendix A

Manual of the Java program "OSSRFSIM"

A.1 Program start

The program is opened by the command line call "java START" (in the program directory, which contains the file `START.class`) - or when using MacOS, by a simple double click on the file `START.class`. After a waiting time of a few seconds, the following window appears (see figure A.1).

In the text field after **"Max. size of memory"** , the maximum amount of system memory (RAM) which shall be used by the program, can be set (the minimum amount is 10 MB and the maximum is 2048 MB, i.e. two gigabytes). The amount of memory should be chosen according to the available free memory on the computer which the program is running on: For example, if it has three gigabyte - or more - of RAM, then a choice of 2048 MB is preferable, in order to avoid an unnecessary limitation of the program. However, if the machine, for example, has 512 MB RAM and half of it is occupied by the operating system and other programs running in the background, then it is recommended not to use more than about 300 MB for this program.



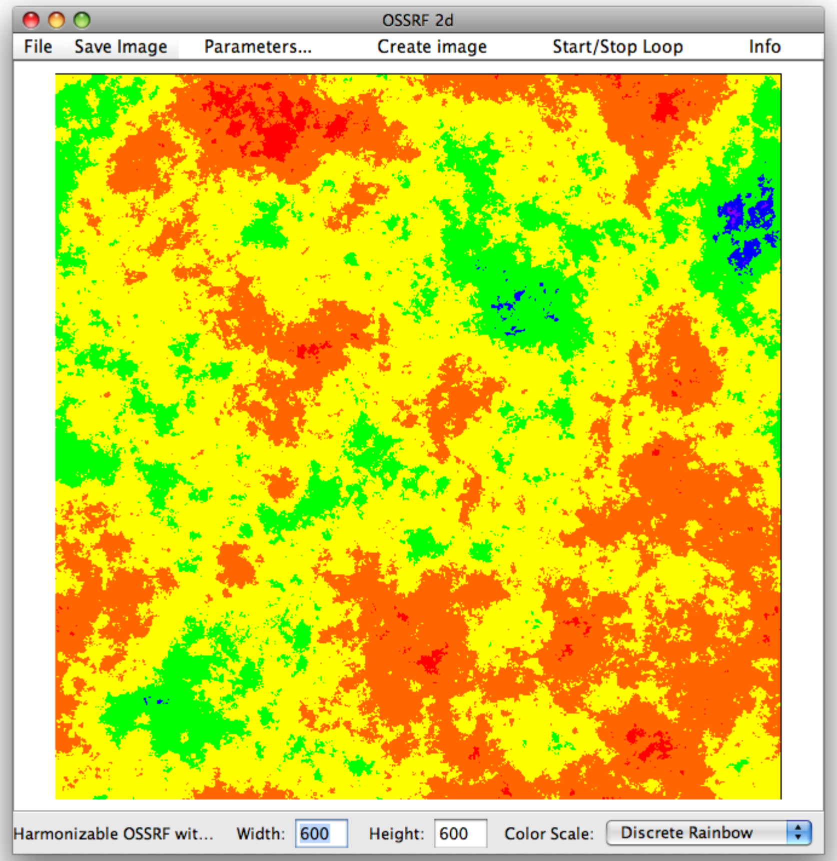
Max. size of memory [in MB]: 2000

Figure A.1: The start screen.

If two-dimensional OSSRFs should be simulated, then the program module for this task can be started by a click on the "Start 2d" button. In the same way, the module for the simulation of three-dimensional OSSRFs is started with a click on the "Start 3d" button. Alternatively, the program may be simply closed by clicking on the "Cancel" button.

A.2 2-dimensional OSSRF

After a click on "Start 2d", the program module for the simulation of two-dimensional OSSRF starts, which shows after a few seconds the image of an OSSRF (see figure A.2).



Harmonizable OSSRF wit... Width: 600 Height: 600 Color Scale: Discrete Rainbow

Figure A.2: The Program window with a simulated 2-dimensional OSSRF.

A.2.1 The main menu

The main menu contains the submenus “File” and “Save Image”, and additionally the menu items “Parameters...”, “Create Image”, “Start/Stop Loop” and “Info”.

A.2.2 The File menu

The menu “File” contains the menu items “Read OSSRF data file”, “Save OSSRF data file” and “Close”. A click on “Save OSSRF data file” opens a dialog which enables the user to choose a directory and specify a filename to which the numerical data of the currently displayed OSSRF should be saved (see figure A.3). This enables the user to view a certain simulated OSSRF again later, or to share it with other users. In the same way, a click on “Read OSSRF data file” opens a dialog which allows to choose a file from which a saved OSSRF can be loaded (instead of simulating a new one). A click on the “Close” menu item closes the program (without asking for confirmation!).

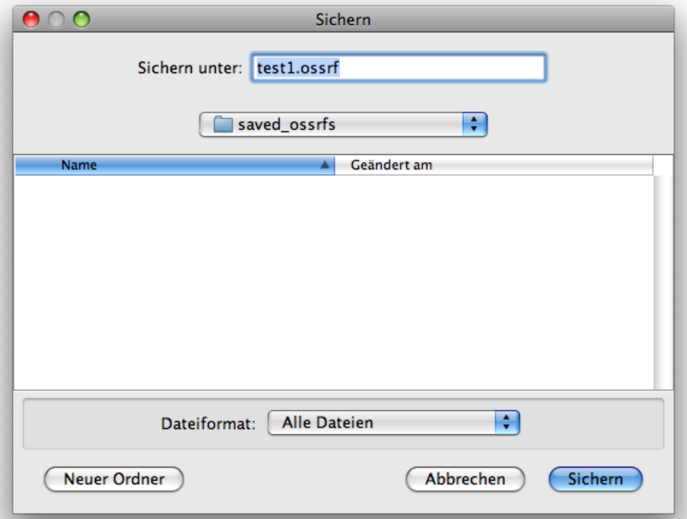


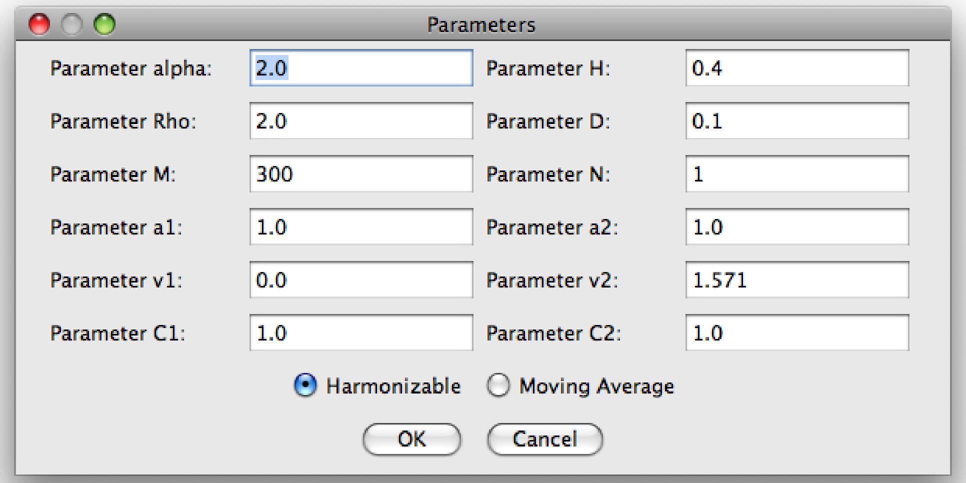
Figure A.3: The file chooser dialog which opens after click on “Save OSSRF data file” (this picture shows the dialog window on a German version of MacOS).

A.2.3 The Save Image menu

The menu "Save Image" contains some menu items which can be used to save the currently displayed image of an OSSRF in an ".png" or ".jpg" image file: "Save JPG (A. Nr.)", "Save PNG (A. Nr.)", "Save JPG as ..." and "Save PNG as ...". When clicking on one of the first two items, a picture is saved immediately into the program folder, with an automatically generated, unique number (and, of course, the corresponding file extension) as filename. When using one one of the latter two, however, a dialog window, like the one shown above in figure A.3, opens so that the user can specify a filename to which the image should be saved. The image is saved as a ".jpg" file when using "Save JPG (A. Nr.)" or "Save JPG as ...", and in the ".png" format when one of the other items is used.

A.2.4 The parameter dialog

A click on the menu item "Parameters ..." opens a dialog window, which allows the user to set the parameters of the simulated OSSRF and of the simulation procedure (figure A.4):


Parameters

Parameter alpha:	2.0	Parameter H:	0.4
Parameter Rho:	2.0	Parameter D:	0.1
Parameter M:	300	Parameter N:	1
Parameter a1:	1.0	Parameter a2:	1.0
Parameter v1:	0.0	Parameter v2:	1.571
Parameter C1:	1.0	Parameter C2:	1.0

[✓ Harmonizable] Moving Average

Figure A.4: The parameter dialog for the simulation of two-dimensional OSSRFs.

At the bottom of the dialog, you can choose wether an OSSRF in harmonizable representation or one in moving average representation should be simulated (At any time, exactly one of these two possibilites is chosen. The default setting is "harmonizable").

If the harmonizable representation is chosen, then the following random field is approximated:

$$X_\psi(x) = \operatorname{Re} \int_{\mathbb{R}^2} \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-(a_1+a_2)/\alpha} W_\alpha(d\xi), \qquad x \in \mathbb{R}^2$$

and in the moving average case the following random field:

$$X_\varphi(x) = \int_{\mathbb{R}^2} \left(\varphi(x-y)^{H-(a_1+a_2)/\alpha} - \varphi(-y)^{H-(a_1+a_2)/\alpha}\right) Z_\alpha(dy), \qquad x \in \mathbb{R}^2$$

Thereby, in any case the occuring integral over $\mathbb{R}^2$ is approximated by a finite sum: In the harmonizable case this is

$$X_\psi^{A,M}(x) = \operatorname{Re} \sum_{(k,l)\in J} \left(e^{i<x,\xi_{k,l}>} - 1\right) \psi(\xi_{k,l})^{-H-(a_1+a_2)/\alpha} W_\alpha(\Delta_{k,l})$$

with $J := \{-M, \dots, M-1\}^2 \backslash \{-N, \dots, N-1\}^2$, $\xi_{k,l} = (kD, lD)^T$ and $\Delta_{k,l} = [kD, (k+1)D) \times [lD, (l+1)D)$. In the moving average case the approximating sum is

$$X_{\varphi,A,M}(x) = \sum_{k,l=-M}^{M-1} \left(\varphi(x - y_{k,l})^{H-(a_1+a_2)/\alpha} - \varphi(-y_{k,l})^{H-(a_1+a_2)/\alpha}\right) Z_\alpha(\Delta_{k,l}) \approx X_{\varphi,A}(x)$$

(with $y_{k,l} = (kD, lD)^T$ and $\Delta_{k,l} = [kD, (k+1)D) \times [lD, (l+1)D)$).

In both cases, an E-homogeneous function of the following form is used:

$$\psi(x) = \varphi(x) = \left(C_1 \,|< x, \theta_1 >|^{\rho/a_1} + C_2 \,|< x, \theta_2 >|^{\rho/a_2}\right)^{1/\rho}$$

The meaning of the parameters `alpha` (=α), `H` (=H), `a1` (= a_1), `a2` (=a_2), `C1` (= C_1), `C2` (=C_2) and `rho` (=ρ) can be read from these formulas. The vectors θ_1 and θ_2, which are contained in the last formula, are calculated from the parameters `v1` (= v_1) and `v2` (= v_2) by $\theta_1 = (\cos(v_1), \sin(v_1))^T$ and $\theta_1 = (\cos(v_2), \sin(v_2))^T$.

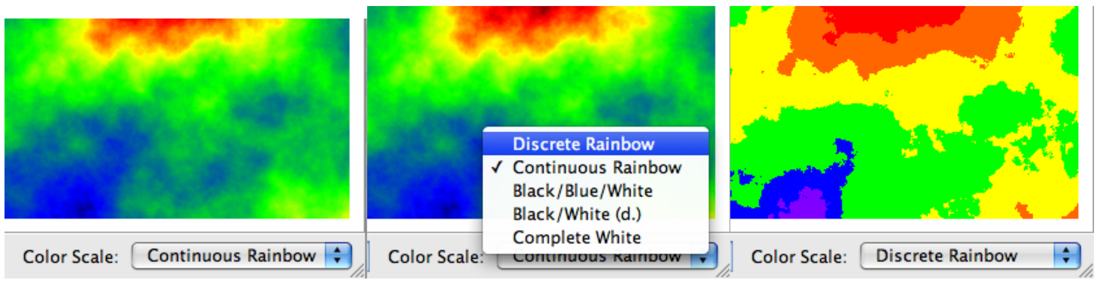
With a click on “OK” the parameter dialog can be closed, and changes to the parameters are saved to the program if they are valid parameter values (however, if e.g. the input in a parameter input field is not a number, or if the input for `M` is not an integer, then the new value will not be accepted, and the text in the input field will be reset to the previous value). Alternatively, the window can be closed without saving the input by a click on “Cancel”.

Simulation of OSSRF

A click on the menu item "Create image" starts the simulation of a new OSSRF with the currently set parameters. As an alternative, the repeated simulation of OSSRF with identical parameter values can be started and stopped with the menu item "Start/Stop Loop".

Display settings: Color scale and size of image

The simulated two-dimensional OSSRF are shown in the program window as a two-dimensional bitmap image, in which each pixel is painted according to the a value in the simulated OSSRF. The color map, which maps each value in the normed OSSRF to a corresponding color, can be chosen in the lower right corner of the program window. A click on the button behind "Color scale" opens a list of availabel color scales, from which a one entry may be chosen (see A.5).



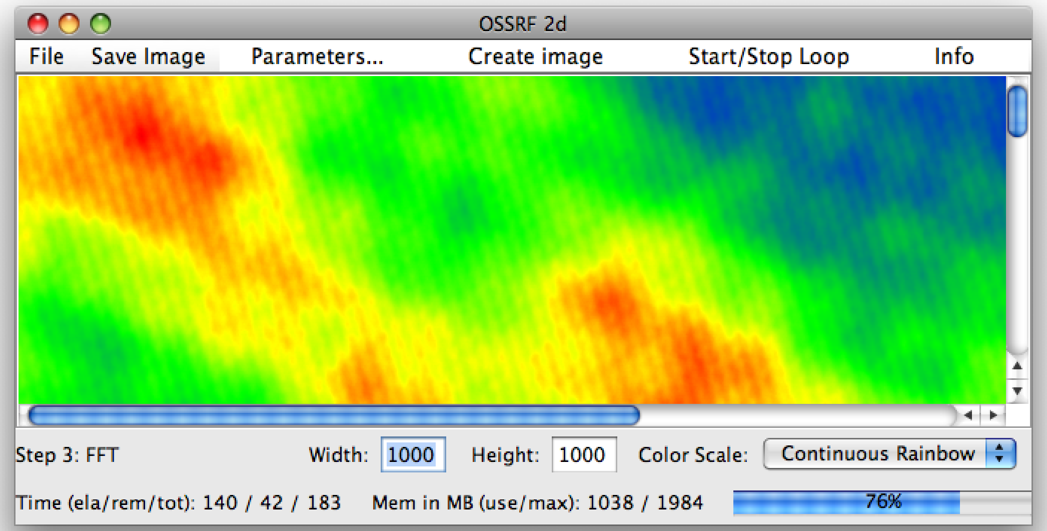
(a) before change (b) changing the map (c) after the change

Figure A.5: Changing the color map.

A.2.5 Progress display

During the simulation of an OSSRF, some information about the progress of the calculations and about the memory usage are displayed in the status bar at the lower edge of the program window (see A.6):

In the upper line (next to the display settings), on the left side a short information about the current task is given (e.g., in the picture A.6: FFT). In the line below, the time since the begin of the calculation, and an estimate of the remaining computing time is shown on the left: From left to right the <u>elap</u>sed time, the estimated <u>rem</u>aining time, and the



Step 3: FFT   Width: 1000   Height: 1000   Color Scale: Continuous Rainbow

Time (ela/rem/tot): 140 / 42 / 183   Mem in MB (use/max): 1038 / 1984   76%

Figure A.6: Display of progress during the simulation (at the lower edge of the window).

resulting total computing time are displayed. The estimated ratio of elapsed time and total time (e.g. the percentage of the calculations which have been done yet) is shown in the progress bar on the right. In the middle, the amount of system memory currently in use by the program ("use") and the maximum amount of memory ("max") which it can use (according to the parameter value which was set in the start window) are displayed.

In the screen shot shown in image A.6, the program is currently calculating a FFT, the total simulation process already took 140 seconds, and the program estimates that this is about 76 percent of the whole simulation. This numbers imlpy a total computation time of about 183 seconds, so that 42 seconds are remaining until the end of the calculation. The program has been started with a limit of 1984 MB of RAM, but is currently using only 1038 MB (in this example, an OSSRF is simulated with parameter $M = 4000$, i.e. the approximated OSSRF is approximated in 64 million points.

A.2.6 Mouse and keyboard commands

Mouse commands

The image of the OSSRF which is shown in the program window can be scrolled and zoomed by moving the mouse over the image while pressing a mouse button. If the right button is pressed and the mouse is dragged vertically, then the image is zoomed in or out. If the mouse is moved up, then the picture is zoomed out (showing a larger part of the OSSRF, if its projection is larger than the displayed image), and if the mouse is moved down, then the image is zoomed in (i.e. magnified). However, zooming can be disabled: The program is always in one of the three zoom modes, which can be changed by the “z” key:

- “0” (no zooming, the projection is always shown at 100% zoom, i.e. one data point is shown as one pixel)
- “1” (squares: In a high zoom factor, each data point of the OSSRF is shown as a small square),
- “2” (interpolation: the color of each pixel is determined by interpolation between the neighboring data points).

If the left button is pressed, then the picture is shifted, following the movement of the mouse. However, the movement of the image is limited: If the image is smaller than the projection of the OSSRF, then the movement of the image stops at the edge of the projection, so that it doesn’t “run out of the OSSRF”, and if the image is larger, than the projection is not moved out of it.

Keyboard commands

The program window recognizes the following keyboard commands:

Key	Function
+	zoom in (magnify image)
-	zoom out (reduce image)
C, c	center the image
I, i	reset zoom factor to 100 %
Q, q	Close the program (“Quit”) (the same as the menu item “File - Close”)
Z	switch to next zoom mode (0 → 1 → 2 → 0)
z	switch to prev. zoom mode (0 → 2 → 1 → 0)

Table A.1: Keyboard commands for the display of 2d OSSRFs.

A.3 3-dimensional OSSRF

The program module for the simulation of three-dimensional OSSRF is started by a click on the “Start 3d” button in the start window. It first shows the parameter dialog, giving the opportunity to adjust the simulation parameters before the first simulation of an OSSRF. After closing the parameter dialog, the simulation of an OSSRF can be started by a click on the “Simulate OSSRF” menu item. To the left side of the program window, the display parameter dialog also appears. It is recommended not to close this window, as it is quite small and can stay on the side of the main window, thus permitting to change the display parameters without requiring to open and close the window every time. However, if it is closed, it is also possible to open it again (by the main menu).

A.3.1 The parameter dialog

The (simulation) parameter dialog can be opened by a click on the menu item “Parameters (Sim.)” in the program window, and allows the user to set the parameters of the simulated OSSRF and of the simulation procedure (see figure A.7).

Below the text fields for the input of the simulation parameters, the choice of “Harmonizable” or “Moving Average” determines, if an OSSRF in harmonizable or in moving average representation should be simulated (only one of these choices can be checked; the default setting is “Harmonizable”).

If the box before “Save 2d Pictures” is checked, then the program exports the OSSRF (immediately after its simulation) to a series of PNG-pictures (in the “pictures”-folder, or in the folder which is specified in the file `options.txt` - if it exists - after `DIRNAME_PICTURES`). Thereby each plane of points with identical z-coordinates is saved in a PNG file, creating $2M$ pictures of size $2M \times 2M$, where M is the parameter of the OSSRF which determines the size of the simulated sample. The x and y coordinates of a point in the OSSRF are also his x and y coordinate in the picture, and the z coordinate is translated to the number of the file. However, saving the series of pictures takes about as much time as the whole simulation process, which is the reason for this function to be switched off by default.

If “Harmonizable” is chosen, then the random field

$$X_\psi(x) = \operatorname{Re} \int_{\mathbb{R}^3} \left(e^{i<x,\xi>} - 1\right) \psi(\xi)^{-H-(a_1+a_2+a_3)/\alpha} W_\alpha(d\xi), \qquad x \in \mathbb{R}^3$$

is simulated, and if “Moving Average” is chosen, the simulated random field is the following:

$$X_\varphi(x) = \int_{\mathbb{R}^3} \left(\varphi(x-y)^{H-(a_1+a_2+a_3)/\alpha} - \varphi(-y)^{H-(a_1+a_2+a_3)/\alpha}\right) Z_\alpha(dy), \qquad x \in \mathbb{R}^3.$$

In each case, the occuring integral over $\mathbb{R}^3$ is approximated by a finite sum, namely in the harmonizable case by

$$X_\psi^{A,B,D}(x) = \mathrm{Re} \sum_{\vec{\mathbf{k}} \in J} \left(e^{i<x,\xi_{\vec{\mathbf{k}}}>} - 1\right) \psi(\xi_{\vec{\mathbf{k}}})^{-H-\frac{(a_1+a_2+a_3)}{\alpha}} W_\alpha(\Delta_{\vec{\mathbf{k}}})$$

with $J := \{-M, \ldots, M-1\}^3 \backslash \{-N, \ldots, N-1\}^3$, and in the moving average case by

$$X_{\varphi,A,M}(x) = \sum_{\vec{\mathbf{k}} \in \{-M,\ldots,M-1\}^3} \left(\varphi(x - y_{\vec{\mathbf{k}}})^{H-\frac{a_1+a_2+a_3}{\alpha}} - \varphi(-y_{\vec{\mathbf{k}}})^{H-\frac{a_1+a_2+a_3}{\alpha}}\right) Z_\alpha(\Delta_{\vec{\mathbf{k}}}).$$

In both cases, an E-homogeneous function in the following form is used:

$$\psi(x) = \varphi(x) = \left(C_1 \,|< x, \theta_1 >|^{\rho/a_1} + C_2 \,|< x, \theta_2 >|^{\rho/a_2} + C_3 \,|< x, \theta_3 >|^{\rho/a_3}\right)^{1/\rho}$$

The meaning of the parameters `alpha` ($=\alpha$), `H` ($=H$), `rho` ($=\rho$), `D` ($=D = \frac{A}{M} = \frac{B}{N}$), `M` ($=M$), `N` ($=N$, used only in the "harmonizable" case), `a1` ($= a_1$), `a2` ($=a_2$), `a3` ($=a_3$), `C1` ($= C_1$), `C2` ($=C_2$) und `C3` ($=C_3$) are to be read from these formulas.

The coordinates of each of the vectors $\theta_1, \theta_2, \theta_3 \in \mathbb{R}^3$ are set in three neighboring input fields (i.e. in the form of a row vector), each vector θ_j to the left of the corresponding eigenvalue a_j.

A click on "OK" closes the dialog, and the changes are saved (if the new input values are valid). If the button "Cancel" is clicked, then the dialog is closed without saving changes to the parameters.

A.3.2 The display parameter dialog

The display parameter dialog is a small window which usually appears at the left side of the screen (see figure A.8). It permits to specify whether a projection of the three-dimensional OSSRF should be displayed in the main window, for which values of the normed OSSRF (values in the range $[0.0, 1.0]$) the corresponding points should be shown, and which color map is to be used (about the color map, see also the subsection A.2.4 "display settings" for two-dimensional OSSRF).

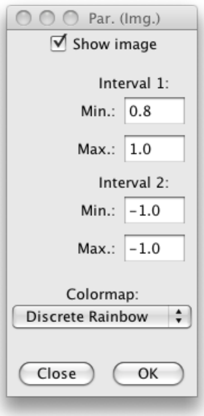

Par. (Img.)
☑ Show image
Interval 1:
Min.: 0.8
Max.: 1.0
Interval 2:
Min.: -1.0
Max.: -1.0
Colormap: Discrete Rainbow

Figure A.8

A.3.3 The main menu

The main menu contains the submenus “File” and “Save Image”, and the additional menu items “Simulate OSSRF”, “Parameters (Sim.)”, “Parameters (Image)” and “Info”.

A click on “Simulate OSSRF” starts a the simulation of a new OSSRF with the currently set parameters. If the OSSRF is too large to fit into the the available RAM (i.e. if the parameter M is too big), then the program automatically uses a method which stores parts of the data on the hard disk between different steps of the calculation. Thereby, the program needs more time for the calculation, but is able to calculate also random fields which need more than the available memory space (especially: which need more than 2 Gigabyte of RAM). Thus, the size of the simulated OSSRF is now limited by the amount of free space on the hard disk. The needed disk space is 24 bytes per data point, i.e. if $M = 512$, then the OSSRF has $(2 * M)^3 = 1024^3$ points (about 1.074 billion), and therefore needs 24 GB of disk space.

The menu item “Parameters (Sim).” opens the simulation parameter dialog (see subsection A.3.1) if it is closed. Analoguously, the menu item “Parameters (Image)” opens the display parameter dialog (see subsection A.3.2).

The submenus “File” and “Save Image” are discribed in the following paragraphs:

The File menu

The menu “File” contains the menu items “Read OSSRF data file”, “Save OSSRF data file”, “Save 2d Pictures” and “Close”. A click on “Save OSSRF data file” opens a dialog which enables the user to choose a directory and specify a filename to which the numerical data of the currently displayed OSSRF should be saved (see figure A.3). This enables the user to view a certain simulated OSSRF again later, or to share it with other users. In the same way, a click on “Read OSSRF data file” opens a dialog which allows to choose a file from which a saved OSSRF can be loaded (instead of simulating a new one). With the menu item “Save 2d Pictures”, the user can export the OSSRF to a series of .png picture files (if the corresponding option has been chosen in the simulation parameters dialog, this action is started automatically after the simulation of the OSSRF, compare the description in subsection A.3.1. Using this menu item, the export tho the picture files can also be done at a later time). A click on the “Close” menu item closes the program (without asking for confirmation!).

The Save Image menu

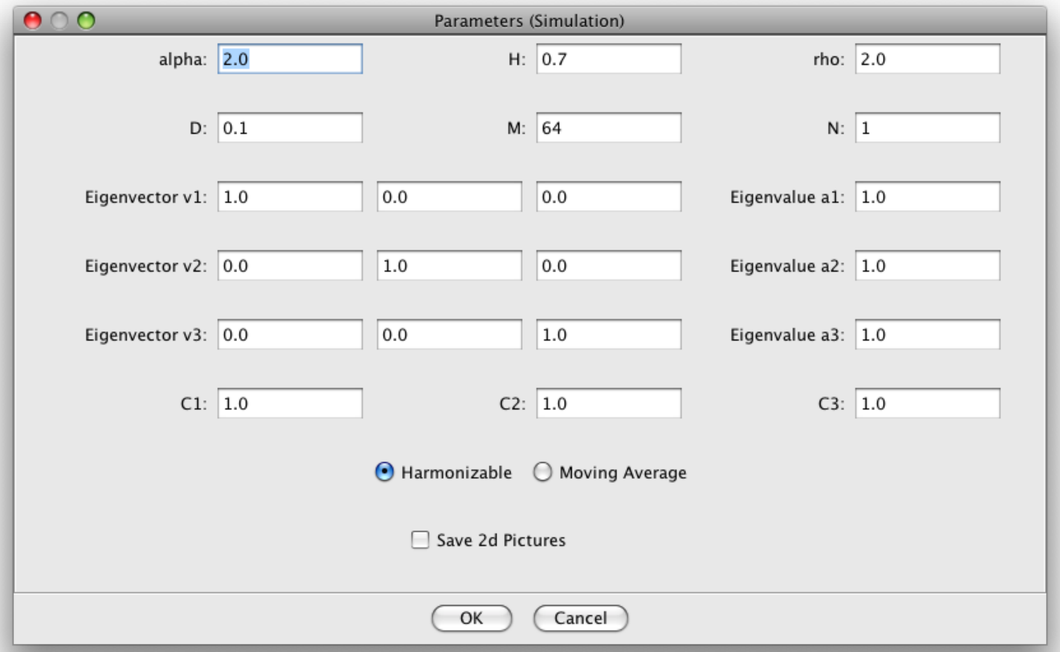
Using the items in the "Save Image" menu, the projection of the three-dimensional OSSRF, which is shown in the program window, can be saved to a .jpg or a .png file. The menu and its items is identical to the corresponding menu in the module for two-dimensional OSSRFs (compare subsection A.2.3).

A.3.4 Mouse and keyboard commands in the main window

The projection of the OSSRF in the program window can be changed by mouse movements while a mouse button is pressed, and by keyboard commands. The image of the OSSRF is rotated by moving the mouse over the window while the left mouse button is pressed. If the right mouse button is pressed, then a vertical movement of the mouse increases or decreases the size of the projection (zoom in / out). The keys which are accepted by the program window as commands to the OSSRF projection are listed in table A.2 (the program window must have the input focus, which means for example that after an input to the image parameter dialog, the main window must be clicked in order to regain the focus, before accepting keyboard commands).

A.3.5 Progress display

In the status bar (at the lower edge of the program window), the progress is displayed during calculations. The progress display in the 3d module is the same as for the 2d module (see subsection A.2.5).

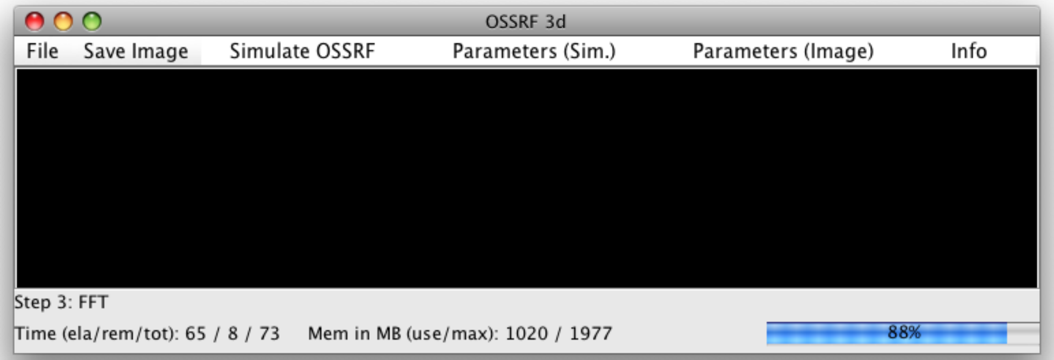


alpha:	2.0	H:	0.7	rho:	2.0
D:	0.1	M:	64	N:	1
Eigenvector v1:	1.0	0.0	0.0	Eigenvalue a1:	1.0
Eigenvector v2:	0.0	1.0	0.0	Eigenvalue a2:	1.0
Eigenvector v3:	0.0	0.0	1.0	Eigenvalue a3:	1.0
C1:	1.0	C2:	1.0	C3:	1.0

● Harmonizable ○ Moving Average

☐ Save 2d Pictures

Figure A.7: The Parameter dialog for the simulation of three-dimensional OSSRFs.



Step 3: FFT

Time (ela/rem/tot): 65 / 8 / 73 Mem in MB (use/max): 1020 / 1977

Figure A.9: Main menu and status bar.

Key	Function
1	Switch to parallel projection.
2	Switch to perspectivic projection.
3	Switch to anaglyph projection (red/cyan 3D glasses needed)
B, b	Switch between black and white background.
0	Reset transformation matrix to unit matrix,
	i.e. undo all rotations
+	Magnify image (zoom in)
-	Reduce image (zoom out)
	(only relevant in anaglyph projection mode:)
e	Decrease distance between eyes.
E	Increase distance betwee eyes.
f	Leave full screen mode.
F	Enter full screen mode.
p	reduce size of pixels.
P	increase size of pixels.
Q	Close (Quit) program.
	(not relevant in parallel projection mode:)
w	Decrease distance between eyes and origin.
	(stronger perspectivic deformation)
W	Increase distance between eyes and origin.
	(less perspectivic deformation)
x	Rotation of the OSSRF about the X-axis
X	Rotation of the OSSRF about the X-axis (opposite direction)
y	Rotation of the OSSRF about the Y-axis
Y	Rotation of the OSSRF about the Y-axis (opposite direction)
c	Rotation of the OSSRF about the Z-axis
C	Rotation of the OSSRF about the Z-axis (opposite direction)

Table A.2: Keyboard commands for the display of 3d OSSRFs.

A.4 The option.txt file

Some additional constants of the program, which are not set in a parameter dialog (e.g. the name of the directory in which the temporary files are stored, or the filenames of temporary files), can be read from a text file called "option.txt" which has to be in the program folder in order to be found. This file allows to change these settings without editing the source code and compiling again. If this file is not found, certain default settings are used.

The file "option.txt" may, for example, look like this:

```
## DO NOT EDIT THE TEXT BEFORE THE '='-SIGN !!!
##
DIRNAME_PICTURES=pictures
DIRNAME_CACHE=cache
FILENAME_OSSRF_DATA=ossrf_data3d
FILENAME_POINTS=points3d
FILENAME_PICTURES=img
## simulation mode for  2d ossrf:
## "8byte" / "64bit"  or "4byte" / "32bit"
2D_VARIANT=8byte
## number of parallel processes in simulation methods
## which support parallel computation
## (an integer number)
NUMBER_OF_PARALLEL_PROC=1
```

Lines which set a constant, have the format "KEY=value", without any whitespaces. The key, including the "="-sign, has to be written exactly as in the example file, only the values after the "="-sign may be changed. The constants to be read from the "option.txt"-file are summarized in table A.3.

Key	Meaning
DIRNAME_PICTURES	Name of the directory to which the series of 2d pictures are exported from a 3d OSSRF.
FILENAME_PICTURES	Name of the files of the 2d pictures which are generated from the 3d OSSRFs.
DIRNAME_CACHE	Name of the directory into which temporary files are stored (the directory and its contents may be deleted after closing the program).
FILENAME_OSSRF_DATA	Name of the (temporary) file(s) into which large 3d OSSRFs are stored.
FILENAME_POINTS	The name of the file to which the displayed points in the projection of a three-dimensional OSSRF are saved.
2D_VARIANT	Sets whether the OSSRF data of two-dimensional OSSRF should be stored in 32bit (`float`) or in 64bit (`double`) floating point variables. Using 32bit (4byte) variables allows the simulation of larger OSSRF in a certain amount of RAM.
NUMBER_OF_PARALLEL_PROC=1	Some parts of the simulation of harmonizable OSSRF can be split into several parallel processes, in order to use more than one CPU core, and thus accelerate the computation. This parameter determines into how many parallel threads the computation is distributed (a value of 1 means "no parallel computation", a value of 2 uses both cores of a dual-core system, etc.)

Table A.3: Parameter names in the file "option.txt".

Appendix B

Contents of the attached CD

- **Java**: This directory contains the Java implementation ("OSSRFSIM") of the simulation algorithms.
  - **intro_pictures**: This directory contains some example pictures which are used in the "Start"-window (when starting the program) and in the "Info"-window.
  - **javadoc**: This directory contains the javadoc-files (short documentation of the java classes, in the HTML format) which have been generated from the source files (start with *index.html* to view the documentation).
  - **ossrfsim**: This directory contains the "*.class" Java bytecode files of the Java implementation.

    Under MacOS, the module for the simulation of two-dimensional OSSRF can be started by a double click on *OSSRFWindow2d.class*, and the module for three-dimensional OSSRF by a double click on *OSSRFWindow3d.class.* Under all common operating systems (including all versions of Windows, MacOS or Linux), each module can be started in a command line shell, e.g. the "2d"-module by the command "java OSSRFWindow2d". Using the parameter "-Xmx*MSIZE*", the maximal available memory space for this program can be specified to be *MSIZE*, for example a maximal amount of 2 GB by the call "java -Xmx2048M OSSRFWindow2d". However, it is recommended to start the program using the "START"-file in the program directory (then the maximum amount of memory can be specified in the start window, see section A.1).
  - **src**: This directory contains the "*.java" source files of the Java project.
  - **options.txt**: A textfile which can be used to set some options for the Java implementation (see section A.4).

  - **START.class**: This file can be used to start the Java program: Open a command line shell, go to this folder and call “java START”. Under MacOS, the program can alternatively also be started by a double click on this file (see section A.1).

- **Matlab**: This directory contains the Matlab implementations of the simulation algorithms (see section 6.1) and of the estimation algorithm of chapter 7.

  - **ossrf_2d**: This directory contains the Matlab implementations of the simulation algorithms for two-dimensional OSSRFs (see subsection 6.1.1).

    * **ossrfgui2d.fig**: GUI definitions for *ossrfgui2d.m*.

    * **ossrfgui2d.m**: A graphical user interface to input the parameters for a 2d OSSRF, and start its simulation. It uses *simuH2dl* or *simuM2d* (depending on the type of the OSSRF) for the simulation.

    * **ossrfH2d.m**: Simulates a harmonizable OSSRF. Parameter values are defined in the source code at the beginning of the file. It uses *simuH2ds* or *simuH2dl* (depending on the size of the OSSRF) for the simulation.

    * **ossrfHi2d.m**: The same as *ossrfH2d.m*, except that parameter values are requested as user input on the Matlab command line rather than specified in the source code.

    * **ossrfHs2d.m**: Simpler version of *ossrfH2d.m* for a small OSSRF. It uses the function *simuH2ds* for the simulation and should not be used for large OSSRF with $M > 2000$.

    * **ossrfM2d.m**: Simulates a harmonizable OSSRF. Parameter values are defined in the source code at the beginning of the file. It uses *simuM2d* for the simulation.

    * **ossrfMi2d.m**: The same as *ossrfM2d.m*, except that parameter values are requested as user input on the Matlab command line rather than specified in the source code.

    * **simuH2dl.m**: Function for the simulation of middle and large 2d harmonizable OSSRFs. It should be used by calling one of the *ossrf*.m*-files.

    * **simuH2ds.m**: Function for the simulation of small 2d harmonizable OSSRFs. It should be used by calling one of the *ossrf*.m*-files.

    * **simuM2d.m**: Function for the simulation of 2d moving-average OSSRFs. It should be used by calling one of the *ossrf*.m*-files.

- **ossrf_3d**: This directory contains the Matlab implementations of the simulation algorithms for three-dimensional OSSRFs (see subsection 6.1.2).
  - **displayossrf3d.m**: Function for the display of 3d OSSRFs. It is used by *ossrf3d.m* and *ossrfgui3d.m*.
  - **ossrf3d.m**: Simulates a 3d OSSRF. Parameter values are defined in the source code at the beginning of the file. It uses one of the *simu**-functions (depending on the specified parameters) for the simulation, and *displayossrf3d* for the presentation of the OSSRF.
  - **ossrfgui3d.fig**: GUI definitions for *ossrfgui3d.m*.
  - **ossrfgui3d.m**: A graphical user interface to input the parameters for a 3d OSSRF, and start its simulation. It uses one of the *simu**-functions (depending on the specified parameters) for the simulation, and *displayossrf3d* for the presentation of the OSSRF.
  - **simuH3dl.m**: Function for the simulation of middle and large 3d harmonizable OSSRFs. It should be used by calling *ossrf3d.m* or *ossrfgui3d.m*.
  - **simuH3ds.m**: Function for the simulation of small 3d harmonizable OSSRFs. It should be used by calling *ossrf3d.m* or *ossrfgui3d.m*.
  - **simuM3d.m**: Function for the simulation of 3d moving-average OSSRFs. It should be used by calling *ossrf3d.m* or *ossrfgui3d.m*.
- **ossrfestim_a1a2v1v2alpha**: This directory contains the Matlab implementation of the estimation algorithm (for two-dimensional, harmonizable OSSRFs, see section 7.2).
  - **diffSqrSum.m**: The function for the calculation of the "sum of squares", which has to be minimized. It is used by *runEstimation.m*.
  - **gen2Dharmo.m**: The function for the simulation of 2d harmonizable OSSRFs. It is used by *runEstimation.m* to create the OSSRF whose parameters have to be estimated.
  - **runEstimation.m**: The main file of the estimation test program.

Additionally, each directory contains the three files of the *rstab*-package by Prof. Vandev from the University of Sofia (i.e. the files *d2.m*, *rstab.m* and *tan2.m*).

- **thesis.pdf**: This thesis in the form of a PDF file.

List of symbols and abbreviations

Special symbols

$\mathbb{N}$	The positive integers $\{1, 2, 3, \ldots\}$
$\mathbb{N}_0$	The non-negative integers $\mathbb{N} \cup \{0\} = \{0, 1, 2, 3, \ldots\}$
$\mathbb{Z}$	The integer numbers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
$\mathbb{R}$	The real numbers
$\mathbb{C}$	The complex numbers
$\|\| \cdot \|\|_\rho$	The ρ-norm $\|\|\vec{\mathbf{x}}\|\|_\rho := \left(\sum_{j=1}^{d} x_j^\rho\right)^{1/\rho}$ for $\vec{\mathbf{x}} \in \mathbb{R}^d$, $\rho \in [1, \infty)$
$\|\| \cdot \|\|_\infty$	The maximum-norm $\|\|\vec{\mathbf{x}}\|\|_\infty := \max_{1 \leq j \leq d}(x_j)$ for $\vec{\mathbf{x}} \in \mathbb{R}^d$
S_∞	The $\|\| \cdot \|\|_\infty$-unit-sphere ($S_\infty := \{x \in \mathbb{R}^d : \|\|x\|\|_\infty = 1\}$)
S_2	The $\|\| \cdot \|\|_2$-unit-sphere ($S_2 := \{x \in \mathbb{R}^d : \|\|x\|\|_2 = 1\}$)
$\{k\}^n$	The vector of length n, all of whose elements have the value k, i.e. $\{k\}^n = (k, \ldots, k)^T$ (n components)
$\vec{\mathbf{k}}$	The vector $(k_1, \ldots, k_d)^T$
$\xi_{\vec{\mathbf{k}}}$	The vector $\vec{\mathbf{k}} \cdot D = (k_1 \cdot D, \ldots, k_d \cdot D)^T$
$\Delta_{\vec{\mathbf{k}}}$	The hypercube $[k_1 \cdot D, (k_1 + 1) \cdot D) \times \ldots \times [k_d \cdot D, (k_d + 1) \cdot D)$
$< x, y >$	The scalar product ("dot product") of the vectors x and y.
$X \sim S_\alpha(\sigma, \beta, \mu)$	The random variable X is distributed according to an α-stable distribution with shape parameter β, scaling parameter σ and location parameter μ.
Γ	$\mathbb{R}^d \backslash \{0\}$.

Frequently used abbreviations

OSSRF	operator scaling stable random field
r.v.	random variable
DFT	discrete Fourier transform
FFT	fast Fourier transform
mio.	million ($= 10^6$)
bio.	billion ($= 10^9$)

Bibliography

[1] Hermine Biermé, Mark M. Meerschaert, Hans-Peter Scheffler: Operator scaling stable random fields, Stochastic Processes and their Applications 117, pages 312-332, Elsevier, 2007

[2] Michael Clausen, Ulrich Baum: Fast Fourier Transforms BI-Wissenschafts-Verlag, 1993

[3] J.W. Cooley, J.W. Tukey: An algorithm for the machine calculation of complex Fourier series, Math. Comput. 19 (90): 297301, 1965. *http://dx.doi.org/10.1090%2FS0025-5718-1965-0178586-1*

[4] J.M. Chambers, C.L. Mallows, B.W. Stuck: A Method for Simulating Stable Random Variables, Journal of the American Statistical Association, vol. 71, nr. 354, pages 340-344, 1976

[5] JAVA 2 SE 5.0 API Specification, *http://download.oracle.com/javase/1.5.0/docs/api/index.html*

[6] G. Krüger, T. Stark: Handbuch der Java-Programmierung, 6. Auflage, Addison-Wesley, 2009; free-of-charge download version: *www.javabuch.de*

[7] MATLAB Online Function Reference, *http://www.mathworks.com/help/techdoc/ref/*

[8] M.M. Meerschaert, H.P. Scheffler, Limit Distributions for Sums of Independent Random Vectors: Heavy Tails in Theory and Practice, Wiley-Interscience, New York, 2001.

[9] W. Schweizer: MATLAB kompakt, 2. überarbeitete Aufl. Oldenbourg, München, 2007

[10] G. Samorodnitsky, M.S. Taqqu: Stable Non-Gaussian Random Processes, Chapman and Hall, New York, 1994

[11] James S. Walker: Fast Fourier Transforms, CRC Press, 1991