A Lanczos-based Method for Computing Ornstein-Uhlenbeck Representations for the Generalized Langevin Equation

Dissertation

zur Erlangung des Grades
Doktor der Naturwissenschaften

am Fachbereich Physik, Mathematik und Informatik
der Johannes Gutenberg-Universität
in Mainz

vorgelegt von

Niklas Johannes Bockius

geboren in Wiesbaden, Deutschland

Mainz, den 10. Oktober 2025

1. Berichterstatter: Prof. Dr. Martin Hanke-Bourgeois

2. Berichterstatterin: Prof. Dr. Friederike Schmid

Datum der mündlichen Prüfung: 12. Januar 2026

Abstract

The generalized Langevin equation is a common equation for modeling the motion of particles in molecular dynamics. It describes the velocity of a macromolecule in a medium consisting of many smaller particles via a convolution over the history of its velocity and a memory kernel, which encodes the medium's physical properties and allows to take into account long-term consequences such as the influence of collisions on the motion of the medium's particles, which can indirectly influence the macromolecule in the long term. The generalized Langevin equation employs the coarse graining approach of only simulating the macromolecule's velocity explicitly while collisions with the particles of the medium are modeled via a random force process. It is known that if the memory is a Prony series, the generalized Langevin equation can be represented by an Ornstein-Uhlenbeck process by introducing auxiliary variables, which simplifies both its theoretical analysis and its numerical simulation.

While the forward problem, generating sample paths to a given generalized Langevin equation, is interesting in itself, we are more interested in the inverse problem, which consists in determining a good approximation of the memory kernel given a solution of a generalized Langevin equation. By a solution we here mean the distribution of the solution process, which is given by the auto-covariance function of the velocity. This function is deterministic and can be easily estimated given a sufficient number of sample paths. The inverse problem is known to be ill-posed, therefore stability is a major issue with methods which solve this problem. In this thesis we introduce a new method for solving the inverse problem. Instead of first approximating the memory kernel in order to determine a suitable Prony series and an Ornstein-Uhlenbeck process approximation, this method directly yields an Ornstein-Uhlenbeck process which approximates the given velocity auto-covariance function of a generalized Langevin equation. It is based on a variant of the Lanczos algorithm which allows interpolating the velocity auto-covariance function by a Prony series and determines the corresponding Ornstein-Uhlenbeck process, followed by several post-processing steps. By using the multi-dimensional Lanczos algorithm, it can also be applied to the multi-dimensional generalized Langevin equation, where the velocity is a vector.

We test this method using data from several molecular dynamics simulations, including noisy data to assess the method's robustness. For the one-dimensional case we introduce and assess an extension of the algorithm for the setting of a generalized Langevin equation where the macromolecule is subject to an additional constant external force. In this case it is essential to obtain an Ornstein-Uhlenbeck process whose mobility is close to the actual mobility of the generalized Langevin equation.

Acknowledgment

Work on this project, which resulted in this thesis, was funded by Deutsche Forschungsgemeinschaft (DFG), project A3 “Coarse-graining frequency-dependent phenomena and memory in colloidal systems” of the Collaborative Research Center SFB TRR 146 “Multiscale Simulation Methods for Soft Matter Systems”; corresponding financial support was granted by Deutsche Forschungsgemeinschaft (DFG) via Grant 233630050.

For reasons of privacy protection, the remaining acknowledgements are left blank in the electronic version of this thesis.

Contents

1. Introduction

1.1. Overview over this Thesis

The generalized Langevin equation (GLE) is a commonly used equation in molecular dynamics. It describes the motion of a macromolecule (or several macromolecules) in a medium. While it simulates only the macromolecule's velocity directly, the motion of the medium's molecules is taken into account, too, albeit indirectly. For this the GLE contains a memory term, which enables it to depend on the history of the particle's velocity, and a stochastic force term. While the stochastic term simulates the immediate impact of collisions with the mediums particles, the memory term takes into account long-term consequences such as the influence of collisions on the motion of the medium's particles, which can indirectly influence the macromolecule in the long term.

In this thesis we are interested in the inverse problem, which consists in determining the memory kernel given a solution of a GLE. By a solution we here mean the stochastic distribution of the velocity. Since the velocity is a continuous, centered Gaussian process, the distribution is given by the auto-covariance function. We assume that the particle's velocity is a stationary process, hence this is a function of one variable. Often the complete velocity auto-covariance function (VACF) is not available, but its values at certain times, usually in an equidistant time grid, are. On top of determining the memory kernel, we are also interested in writing the corresponding GLE as an Ornstein-Uhlenbeck process. In an Ornstein-Uhlenbeck process the memory term is represented by auxiliary variables. Since it is a Markov process, this significantly simplifies numerical simulation of the corresponding GLE and computations of its analytical properties. Our goal is developing a method which computes an Ornstein-Uhlenbeck representation of a GLE without first having to determine an approximation of the memory kernel in order to avoid known stability problems of the latter problem. (Note, however, that the inverse problem itself is ill-posed, therefore no completely stable algorithm for solving it exists.)

This thesis is organized as follows: In the remaining chapter we will clarify the used notation and give an introduction into the stochastic foundations required to define Ornstein-Uhlenbeck processes. In Chapter 2 we give a detailed introduction into the generalized Langevin equation and the corresponding inverse problem and explain how it can be represented by an Ornstein-Uhlenbeck process. In Chapter 3 we present a method for solving the one-dimensional inverse problem and generating an Ornstein-Uhlenbeck system whose VACF approximates the given VACF in a given equidistant grid. In Chapter 4 we will investigate an extension of the generalized Langevin equation, obtained by adding a constant external force which affects the macromolecule, and assess an adapted version of this method to obtain an extended Ornstein-Uhlenbeck representation for the corresponding process. In Chapters 5 and 6 we will extend the algorithm from

Chapter 3 to the multidimensional generalized Langevin equation with a matrix-valued VACF and assess its numerical results.

1.2. Notation and Useful Lemmas

1.2.1. Notation

If A is a matrix, A^T denotes its transposed matrix while $A^* = \overline{A}^T$ denotes the conjugate transposed matrix. $A_{i,j}$ is the matrix entry at position (i, j) while $A_{i,\cdot}$ is its ith row and $A_{\cdot,j}$ its jth column. $\mathbf{I}_n$ denotes the $n \times n$ identity matrix. $\mathbf{0}_{m,n}$ is the $m \times n$ matrix containing only zeros and $\mathbf{0}_n$ is the zero vector in $\mathbb{R}^n$. We sometimes omit the dimensions when they are obvious from the context or irrelevant. e_k is the kth unit vector where the dimension is again clear from the context when it is not explicitly mentioned.

We will often write matrices in block notation, i.e. write them as blocks of smaller matrices. To simplify the specification of the blocks' dimensions, we introduce the following notation:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \in \mathbb{R}^{(m|n)\times(p|q)}$$

means that A is a matrix consisting of the blocks A_{11}, A_{12}, A_{21}, and A_{22} where $A_{11} \in \mathbb{R}^{m\times p}$, $A_{12} \in \mathbb{R}^{m\times q}$, $A_{21} \in \mathbb{R}^{n\times p}$, and $A_{22} \in \mathbb{R}^{n\times q}$. So $\mathbb{R}^{(m|n)\times(p|q)}$ is actually $\mathbb{R}^{(m+n)\times(p+q)}$, but used in the above context it also specifies the dimensions of the blocks of A. Of course this notation depends not only on A itself but also on the used block notation and is thus not strictly well-defined. However, we will only use it where it is clear from the block notation what we mean.

$\sigma(A)$ denotes the set of all eigenvalues of a quadratic matrix A. By $\operatorname{Log} x$, $x \in \mathbb{C} \setminus \{0\}$, we denote the principal branch of the complex logarithm, i.e. $\operatorname{Im} \operatorname{Log} x \in (-\pi, \pi]$. For non-singular quadratic matrices A we write $\operatorname{Log} A$ for the matrix logarithm which satisfies $\operatorname{Im} \lambda \in (-\pi, \pi]$ for each $\lambda \in \sigma(\operatorname{Log} A)$.

For a suitable function f, the notation $\mathcal{F}f$ denotes its Fourier transform, defined by

$$\mathcal{F}f(\xi) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\mathrm{i}t\xi} f(t)\,\mathrm{d}t,$$

while $\mathcal{L}f$ denotes its Laplace transform

$$\mathcal{L}f(\xi) := \int_{0}^{\infty} e^{-\xi t} f(t)\,\mathrm{d}t.$$

$\mathcal{N}(\mu, \Sigma)$ denotes the normal distribution with expectation $\mu \in \mathbb{R}^m$ and covariance matrix $\Sigma \in \mathbb{R}^{m\times m}$. For a real random variable $X_1 \in L^1(\mathbf{P})$ on a probability space $(\Omega, \mathcal{A}, \mathbf{P})$ with $X_1(\omega) \in \mathbb{R}^m$ for every $\omega \in \Omega$, $\mathbf{E}[X_1] \in \mathbb{R}^m$ denotes its expectation. For real random variables $X_1, X_2 \in L^2(\mathbf{P})$ with $X_1(\omega) \in \mathbb{R}^m$, $X_2(\omega) \in \mathbb{R}^n$ for every $\omega \in \Omega$,

$$\mathbf{Cov}[X_1, X_2] := \mathbf{E}\big[(X_1 - \mathbf{E}[X_1])(X_2 - \mathbf{E}[X_2])^T\big] \in \mathbb{R}^{m\times n}$$

denotes the covariance matrix of X_1 and X_2 and $\mathbf{Var}[X_1] := \mathbf{Cov}[X_1, X_1]$ denotes the covariance matrix of X_1 (or variance in the one-dimensional case).

1.2.2. Two Useful Lemmas

In the following we state two well-known identities, which we will use several times throughout this thesis.

Definition and Lemma 1.1 (Schur Complement)**.** *Let*

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \in \mathbb{R}^{(m|n)\times(m|n)}$$

with H_{22} non-singular. Then $S := H_{11} - H_{12}H_{22}^{-1}H_{21}$ is the Schur complement *of H_{22} in H. If S and H are non-singular, then*

$$H^{-1} = \begin{bmatrix} S^{-1} & -S^{-1}H_{12}H_{22}^{-1} \\ -H_{22}^{-1}H_{21}S^{-1} & H_{22}^{-1} + H_{22}^{-1}H_{21}S^{-1}H_{12}H_{22}^{-1} \end{bmatrix}. \tag{1.1}$$

Proof. One directly verifies $H^{-1}H = \mathbf{I}$. □

The following lemma is a special case of the Leibniz integral rule.

Lemma 1.2. *Let $t > 0$ and let $D \subseteq \mathbb{R}^2$ be open with $\{(t_1, t_2) \in \mathbb{R}^2 : 0 \le t_2 \le t_1 \le t\} \subseteq D$. Let further $f\colon D \to \mathbb{R}^{d\times d}$ be continuous and continuously differentiable w.r.t. the first argument with derivative f_1 and define $g(t) := \int_0^t f(t,s)\,\mathrm{d}s$. Then g is differentiable and*

$$g'(t) = \int_0^t f_1(t,s)\,\mathrm{d}s + f(t,t).$$

Proof. The formal definition of the derivative of f yields

$$\begin{aligned} g'(t) &= \lim_{t'\to t} \frac{g(t) - g(t')}{t - t'} = \lim_{t'\to t} \frac{1}{t-t'}\left(\int_0^t f(t,s)\,\mathrm{d}s - \int_0^{t'} f(t',s)\,\mathrm{d}s\right) \\ &= \lim_{t'\to t} \frac{1}{t-t'}\left(\int_0^t (f(t,s) - f(t',s))\,\mathrm{d}s - \int_t^{t'} f(t',s)\,\mathrm{d}s\right) \\ &= \int_0^t \lim_{t'\to t} \frac{f(t,s) - f(t',s)}{t-t'}\,\mathrm{d}s + \lim_{t'\to t}\frac{1}{t-t'}\int_{t'}^t f(t',s)\,\mathrm{d}s \\ &= \int_0^t f_1(t,s)\,\mathrm{d}s + f(t,t) \end{aligned}$$

□

1.3. Recapitulation of Stochastic Foundations

The following introduction is based on [39, Chapters 1–5] and [26, Chapters 21 and 25].

1.3.1. The Brownian Motion

Definition 1.3. *Let $(\Omega, \mathcal{A}, \mathbf{P})$ be a probability space and $I \subseteq \mathbb{R}$ a (finite or infinite) interval.*

(i) A d-dimensional (real-valued) stochastic process *with time domain I is a map $X\colon I \times \Omega \to \mathbb{R}^d$ such that $X(t, \cdot)$ is measurable w.r.t. $\mathcal{A}$ for every $t \in I$.*

(ii) A set $(\mathcal{A}_t)_{t\in I}$ of σ-algebras with $\mathcal{A}_t \subseteq \mathcal{A}$ for every t is called a filtration *if $\mathcal{A}_t \subseteq \mathcal{A}_{t'}$ holds for every $t < t'$.*

(iii) A stochastic process is called adapted *to the filtration $(\mathcal{A}_t)_{t\in I}$ if $X(t, \cdot)$ is measurable w.r.t. $\mathcal{A}_t$ for every $t \in I$.*

(iv) Two stochastic processes X and Y with time domain I are called equivalent *if for each $t \in I$ they satisfy $X(t) = Y(t)$ almost surely.*

The interpretation of a filtration $\mathcal{A}$ is that $\mathcal{A}_t$ determines which information is known at time t. An adapted process can be seen as a stochastic process such that $X(t)$ is known at time t.

Definition 1.4. *Let $X = (X(t))_{t\in I}$ be a stochastic process.*

(i) X is called centered *if $\mathbf{E}[X(t)] = \mathbf{0}$ for each $t \in I$.*

(ii) Let $I = [0, \infty)$ or $I = \mathbb{R}$. X is called (strictly) stationary *if for every $n \in \mathbb{N}$ and $t_1, t_2, \ldots, t_n \in I$ the distribution of $\big(X(t_1 + t, \cdot), X(t_2 + t, \cdot), \ldots, X(t_n + t, \cdot)\big)$ does not depend on $t \in I$.*

Definition 1.5. *Let X, Y be stochastic processes with time domains I_X and I_Y, respectively, which satisfy $X(t, \cdot) \in L^2(\mathbf{P})$ for each $t \in I_X$ and $Y(t, \cdot) \in L^2(\mathbf{P})$ for each $t \in I_Y$. The function*

$$C_{X,Y}(s, t) := \mathbf{Cov}\big[X(s, \cdot), Y(t, \cdot)\big]$$

is called the covariance function *of X and Y. If the joint process $\big[X^T, Y^T\big]^T$ is stationary, $C_{X,Y}(s, t)$ depends only on $t - s$ and we thus write $C_{X,Y}(t) := C_{X,Y}(t, 0)$.*

$$C_X(s, t) := C_X(s, t)$$

is the auto-covariance function *of X. If X is stationary, we write $C_X(t) := C_{X,X}(t, 0)$.*

Definition 1.6. *A d-dimensional stochastic process X with time domain I is called a* Gaussian process *if $\big[X(t_1, \cdot)^T, \ldots, X(t_n, \cdot)^T\big]^T \in \mathbb{R}^{nd}$ is multivariate normally distributed for every $n \in \mathbb{N}$ and every $t_1, \ldots, t_n \in I$.*

Definition 1.7. *A one-dimensional stochastic process* $W = (W(t,\cdot))_{t\in[0,\infty)}$ *is called a* Brownian motion *if:*

- $W(0,\omega) = 0$ *almost surely.*
- $W(t,\cdot) \sim \mathcal{N}(0,t)$ *for each* $t \geq 0$.
- *The increments* $\left(W(t_k,\cdot) - W(t_{k-1},\cdot)\right)_{k=1\ldots,n}$ *are independent for every* $n \in \mathbb{N}$ *and all* $0 \leq t_0 < t_1 < \ldots < t_n$.
- *The path* $t \mapsto W(t,\omega)$ *is almost surely continuous.*

A d*-dimensional stochastic process* W *is called a* d-dimensional Brownian motion *if each component* W_i *is a Brownian motion and all components are independent.*

Theorem 1.8. *There exists a probability space on which a Brownian motion exists.*

Theorem 1.9. *A stochastic process* W *is a Brownian motion if and only if it is a centered Gaussian process with auto-covariance function* $C_W(s,t) = \min\{s,t\}$ *whose paths* $t \mapsto W(t,\omega)$ *are almost surely continuous.*

Definition 1.10. *A* Brownian motion on $\mathbb{R}$ *is a stochastic process* $(W(t))_{t\in\mathbb{R}}$ *such that* $(W(t))_{t\in[0,\infty)}$ *and* $(W(-t))_{t\in[0,\infty)}$ *are independent Brownian motions.*

Obviously every Brownian motion can be extended to a Brownian motion on $\mathbb{R}$ by defining $W(t) := \widetilde{W}(-t)$ for $t < 0$ with an independent Brownian motion $\widetilde{W}$.

1.3.2. The Itô Integral with Respect to a Brownian Motion

For this section let $(\Omega, \mathcal{A}, \mathbf{P})$ be a probability space and $W = (W(t,\cdot))_{t\in[0,\infty)}$ a Brownian motion in $(\Omega, \mathcal{A}, \mathbf{P})$. Let further $(\mathcal{A}_t)_{t\in[0,\infty)}$ be a filtration w.r.t. which W is a martingale, i.e. W is adapted to $(\mathcal{A}_t)_{t\in[0,\infty)}$ with $W(t,\cdot) \in L^1(\mathbf{P})$ and $\mathbf{E}[W(t',\cdot)\,|\,\mathcal{A}_t] = W(t,\cdot)$ for every $t, t' \in [0,\infty)$ with $t < t'$, where $\mathbf{E}[W(t',\cdot)\,|\,\mathcal{A}_t]$ denotes the conditional expectation of $W(t',\cdot)$ given the information $\mathcal{A}_t$.

Since W has almost surely infinite variation in every interval $[t,t']$, $t < t'$ (cf. [26, Theorem 21.55]), the Itô integral w.r.t. W cannot be defined as a Riemann-Stieltjes integral. Instead we first define the Itô integral for so-called "elementary functions" and then employ a limit transition in order to generalize the definition to limits of such functions.

Definition 1.11. *Let* $S, T \in [0,\infty)$ *with* $S < T$, $I = [S,T]$, $n \in \mathbb{N}$, *and let*

$$t_i^{(n)} := \begin{cases} S & \text{if } i\cdot 2^{-n} < S, \\ i\cdot 2^{-n} & \text{if } S \leq i\cdot 2^{-n} \leq T, \\ T & \text{if } T < i\cdot 2^{-n}. \end{cases}$$

A function $X\colon I \times \Omega \to \mathbb{R}$*,*

$$X(t,\omega) = \sum_{i=0}^{\infty} \xi_i(\omega) \cdot \mathbb{1}_{[t_i^{(n)}, t_{i+1}^{(n)})}(t) \tag{1.2}$$

with $\mathcal{A}_{t_i^{(n)}}$*-measurable, bounded random variables* ξ_i*,* $i \in \mathbb{N}_0$*, is called an* elementary function*. Here* $\mathbb{1}_A(t) := 1$ *if* $t \in A$ *and* $\mathbb{1}_A(t) := 0$ *otherwise.*

Note that the sum in (1.2) is actually a finite sum as $\mathbb{1}_{[t_i^{(n)}, t_{i+1}^{(n)})}$ is non-zero only for finitely many i.

Definition 1.12. *The* Itô integral *of an elementary function* X *from* (1.2) *is the random variable*

$$\int_S^T X(t,\omega)\,\mathrm{d}W(t,\omega) := \sum_{i=0}^{n} \xi_i(\omega)\Big(W\big(t_{i+1}^{(n)},\omega\big) - W\big(t_i^{(n)},\omega\big)\Big). \tag{1.3}$$

Remark 1.13. According to [39, Lemma 3.1.5], the Itô integral satisfies

$$\mathbf{E}\left[\left(\int_S^T X(t,\omega)\,\mathrm{d}W(t,\omega)\right)^2\right] = \mathbf{E}\left[\int_S^T X(t,\omega)^2\,\mathrm{d}t\right] \tag{1.4}$$

for bounded elementary functions f. If we consider

$$\sqrt{\mathbf{E}\left[\int_S^T X(t,\omega)^2\,\mathrm{d}t\right]}$$

as a seminorm on the vector space $\mathcal{U}(S,T)$ of all bounded elementary functions, the Itô integral is consequently an isometry from $\mathcal{U}(S,T)$ to $L^2(\mathbf{P})$.

We now generalize the Itô integral to the following vector space:

Definition 1.14. *For* $A \subseteq \mathbb{R}$ *denote by* $\mathcal{B}(A)$ *the Borel* σ*-algebra of* A*. Then* $\mathcal{V}(S,T)$ *is the vector space of all stochastic processes* X *such that*

(i) X *is* $\mathcal{B}([0,\infty)) \otimes \mathcal{A}$*-measurable where* $\mathcal{B}([0,\infty)) \otimes \mathcal{A}$ *denotes the product* σ*-algebra of* $\mathcal{B}([0,\infty))$ *and* $\mathcal{A}$*.*

(ii) X *is adapted to the filtration* $(\mathcal{A}_t)_{t\in[0,\infty)}$ *from above.*

(iii) $\mathbf{E}\left[\int_S^T X(t,\omega)^2\,\mathrm{d}t\right] < \infty$*.*

According to [39, p. 47 ff.], for every $X \in \mathcal{V}(S,T)$ there exists a sequence $(X_n)_{n\in\mathbb{N}}$ of bounded elementary functions such that

$$\mathbf{E}\left[\int_S^T \big(X(t,\omega) - X_n(t,\omega)\big)^2\,\mathrm{d}t\right] \xrightarrow{n\to\infty} 0.$$

Consequently $(X_n)_{n\in\mathbb{N}}$ is a Cauchy sequence in $\mathcal{V}(S,T)$ and thus (due to (1.4))

$$\left(\int_S^T X_n(t,\omega)\,\mathrm{d}W(t,\omega)\right)_{n\in\mathbb{N}}$$

is a Cauchy sequence in $L^2(\mathbf{P})$. Since $L^2(\mathbf{P})$ is complete, this sequence converges and we denote the limit

$$\int_S^T X(t,\omega)\,\mathrm{d}W(t,\omega) := \lim_{n\to\infty}\int_S^T X_n(t,\omega)\,\mathrm{d}W(t,\omega),$$

as the *Itô integral* of $X \in \mathcal{V}(S,T)$. By this definition the Itô integral is uniquely determined up to almost sure equality.

The following important properties of the Itô integral can be found in [39, Corollary 3.1.7 and Theorem 3.2.1]:

Theorem 1.15. *Let $X \in \mathcal{V}(S,T)$ and $0 \le S < U < T$.*

(i) $\int_S^T X(t,\omega)\,\mathrm{d}W(t,\omega)$ *is $\mathcal{A}_T$-measurable.*

(ii) $\int_S^T X(t,\omega)\,\mathrm{d}W(t,\omega)$ *is linear in X.*

(iii) $\int_S^T X(t,\omega)\,\mathrm{d}W(t,\omega) = \int_S^U X(t,\omega)\,\mathrm{d}W(t,\omega) + \int_U^T X(t,\omega)\,\mathrm{d}W(t,\omega).$

(iv) $\mathbf{E}\left[\int_S^T X(t,\omega)\,\mathrm{d}W(t,\omega)\right] = 0.$

(v) $\mathbf{E}\left[\left(\int_S^T X(t,\omega)\,\mathrm{d}W(t,\omega)\right)^2\right] = \mathbf{E}\left[\int_S^T X(t,\omega)^2\,\mathrm{d}t\right].$

By replacing $X(t,\omega)$ by $X(t,\omega)+Y(t,\omega)$ and $X(t,\omega)-Y(t,\omega)$ in Theorem 1.15(v) and taking the difference of the resulting equations, we obtain the useful identity

$$\mathbf{E}\left[\left(\int_S^T X(t,\omega)\,\mathrm{d}W(t,\omega)\right)\left(\int_S^T Y(t,\omega)\,\mathrm{d}W(t,\omega)\right)\right] = \mathbf{E}\left[\int_S^T X(t,\cdot)Y(t,\cdot)\,\mathrm{d}t\right] \quad (1.5)$$

for $X, Y \in \mathcal{V}(S,T)$.

The stochastic process $(I(t,\cdot))_{t\in[0,T]}$ defined by

$$I(t,\omega) := \int_0^t X(s,\omega)\,\mathrm{d}W(s,\omega)$$

with $X \in \mathcal{V}(0,T)$ does not necessarily have continuous paths. However, there exists an equivalent continuous process (i.e. a process with continuous paths), cf. [39, Theorem 3.2.5]

and [26, Theorem 25.11]. We will denote this continuous process as the Itô integral from now on, as it is usually done.

As mentioned in [39, p. 35] and [26, Lemma 25.15], it is possible to replace the condition (iii) in Definition 1.14 by the less strict condition

$$\int_S^T X(t,\omega)^2 \,\mathrm{d}t < \infty \qquad \text{almost surely}$$

by applying a suitable limit procedure; however, we omit this generalization since we will not need it.

For the generalization to the multidimensional case, one interprets $X(t,\omega)\,\mathrm{d}W(t,\omega)$ as a matrix-vector multiplication: Let $X\colon [0,\infty)\times\Omega\to\mathbb{R}^{d\times m}$ with $X_{i,j}\in\mathcal{V}(S,T)$ for every i and j and let $W=(W_1,\dots,W_m)$ be an m-dimensional Brownian motion. Then the Itô integral of X w.r.t. W is defined as

$$\left(\int_S^T X(t,\omega)\,\mathrm{d}W(t,\omega)\right)_i := \sum_{j=1}^m \int_S^T X_{i,j}(t,\omega)\,\mathrm{d}W_j(t,\omega),$$

cf. [39, Definition 3.3.1]. Analogously to (1.5) we have in this case

$$\mathbf{E}\left[\left(\int_S^T X(t,\omega)\,\mathrm{d}W(t,\omega)\right)\left(\int_S^T Y(t,\omega)\,\mathrm{d}W(t,\omega)\right)^T\right] = \mathbf{E}\left[\int_S^T X(t,\omega)Y(t,\omega)^T\,\mathrm{d}t\right]. \tag{1.6}$$

If the elementary function X is deterministic (i.e. it does not depend on ω), (1.3) is a sum of independent, centered, normally distributed random variables and is therefore centered and normally distributed. Since L^2-convergence implies convergence in distribution, the L^2-limit of the Itô integrals of such functions is also normally distributed (if it exists). Consequently the Itô integral of a $\mathcal{B}([0,\infty))$-measurable function $X\colon[0,\infty)\to\mathbb{R}$ with $\int_S^T X(t)^2\,\mathrm{d}t<\infty$ is centered and normally distributed. For functions $X\colon[0,\infty)\to\mathbb{R}^{d\times m}$, every linear combination of components of the Itô integral is centered and normally distributed, hence the Itô integral is a centered, multivariate normally distributed random variable whose covariance matrix is given by (1.6).

Deterministic integrands also satisfy the following identity from [39, Theorem 4.1.5] (the generalization to the multidimensional case is straightforward), which corresponds to integration by parts known from Riemann integrals:

Theorem 1.16. *Let $X\colon[S,T]\to\mathbb{R}^{m\times n}$ be a continuous function with bounded variation and W an n-dimensional Brownian motion. Then*

$$\int_S^T X(t)\,\mathrm{d}W(t,\omega) = X(T)W(T,\omega) - X(S)W(S,\omega) - \int_S^T \mathrm{d}X(t)W(t,\omega).$$

(We write $\mathrm{d}X(t)W(t,\omega)$ in the second integral to retain the correct order of the factors

in the matrix-vector multiplication.) If X is differentiable, this implies

$$\int_S^T X(t)\,\mathrm{d}W(t,\omega) = X(T)W(T,\omega) - X(S)W(S,\omega) - \int_S^T X'(s)W(t,\omega)\,\mathrm{d}t.$$

In the remainder of this thesis we will usually omit ω as function argument for better readability, as is common for random variables. A notation such as $X \in \mathbb{R}^d$ with a random variable X thus means $X(\omega) \in \mathbb{R}^d$ for all (or almost all) $\omega \in \Omega$. On top of that, when speaking about equality of stochastic processes, we usually mean equivalence as defined in Definition 1.3(iv), but we will often not mention this explicitly.

1.3.3. Ornstein-Uhlenbeck Processes

Definition 1.17. *Let $A \in \mathbb{R}^{d\times d}$ and $K \in \mathbb{R}^{d\times m}$. Let further $X_0 \in \mathbb{R}^d$ be a random variable, W an m-dimensional Brownian motion, and $\mathcal{A}_t$ the σ-algebra generated by X_0 and $(W(s))_{s\in[0,t]}$. The stochastic differential equation*

$$\mathrm{d}X(t) = AX(t)\,\mathrm{d}t + K\,\mathrm{d}W(t), \qquad X(0) = X_0 \tag{1.7}$$

is called the Ornstein-Uhlenbeck equation *with* drift matrix A *and* diffusion matrix K. *A continuous stochastic process $(X(t))_{t\geq 0}$ which is adapted to $(\mathcal{A}_t)_{t\in[0,\infty)}$ and which satisfies*

$$X(t) = X_0 + \int_0^t AX(s)\,\mathrm{d}s + KW(t), \tag{1.8}$$

for every $t \geq 0$ is called a (d-dimensional) Ornstein-Uhlenbeck process. *We call the pair (A, K) an* Ornstein-Uhlenbeck system.

(1.7) can be seen as a formal notation of the integral equation (1.8), where the final term $KW(t)$ can be rewritten as $KW(t) = \int_0^t K\,\mathrm{d}W(t)$.

For every A and K, a unique Ornstein-Uhlenbeck process X satisfying (1.8) exists, cf. [26, Theorem 26.8] or [39, Theorem 5.2.1], where uniqueness means that X is uniquely determined (up to equivalence of processes) for each X_0 and W. The solution can even be given in an explicit form:

Theorem 1.18. *The solution of* (1.7) *and* (1.8) *is given by*

$$X(t) = e^{tA}X_0 + \int_0^t e^{(t-s)A}K\,\mathrm{d}W(s). \tag{1.9}$$

Proof. Let X be defined by (1.9). Integrating (1.9) and employing Theorem 1.16 (where $W(0) = 0$) yields

$$\begin{aligned}\int_0^t X(s)\,\mathrm{d}s &= \int_0^t e^{sA}X_0\,\mathrm{d}s + \int_0^t \int_0^s e^{(s-u)A}K\,\mathrm{d}W(u)\,\mathrm{d}s \\ &= A^{-1}e^{sA}\Big|_0^t X_0 + \int_0^t \left(KW(s) + \int_0^s Ae^{(s-u)A}KW(u)\,\mathrm{d}u\right)\mathrm{d}s\end{aligned}$$

$$= A^{-1}(e^{tA} - \mathbf{I})X_0 + \int_0^t KW(s)\,\mathrm{d}s + \int_0^t \int_u^t Ae^{sA}\,\mathrm{d}s\, e^{-uA}KW(u)\,\mathrm{d}u$$
$$= A^{-1}(e^{tA} - \mathbf{I})X_0 + \int_0^t KW(s)\,\mathrm{d}s + \int_0^t e^{(t-u)A}KW(u)\,\mathrm{d}u - \int_0^t KW(u)\,\mathrm{d}u.$$

Inserting into (1.8) shows that X indeed solves the equation:

$$\begin{aligned} X_0 + \int_0^t AX(s)\,\mathrm{d}s + KW(t) &= X_0 + A\Big(A^{-1}(e^{tA} - \mathbf{I})X_0 + \int_0^t e^{(t-s)A}KW(s)\,\mathrm{d}s\Big) + \\ &\quad KW(t) \\ &= e^{tA}X_0 + A\int_0^t e^{(t-s)A}KW(s)\,\mathrm{d}s + KW(t) \\ &= e^{tA}X_0 + \int_0^t e^{(t-s)A}K\,\mathrm{d}W(s) \\ &= X(t). \end{aligned}$$

□

An Ornstein-Uhlenbeck process' distribution is uniquely determined by A, K, and the distribution of X_0. While an Ornstein-Uhlenbeck process exists for every distribution of X_0, we are mainly interested in a particular starting distribution for which the solution is a stationary process.

Definition 1.19. *A quadratic matrix is called* stable *if all its eigenvalues have negative real part.*

If A in (1.7) and (1.8) is stable, there is a stationary Ornstein-Uhlenbeck process corresponding to the Ornstein-Uhlenbeck system (A, K), which is a centered Gaussian process, as we will see in Theorem 1.20. To obtain its covariance matrix, we employ (1.6) and exploit the fact that X_0 and W are independent, resulting in

$$\begin{aligned} \mathbf{Var}[X(t)] &= \mathbf{Var}[e^{tA}X_0] + \mathbf{Cov}\Big[e^{tA}X_0, \int_0^t e^{(t-s)A}K\,\mathrm{d}W(s)\Big] + \\ &\quad \mathbf{Cov}\Big[\int_0^t e^{(t-s)A}K\,\mathrm{d}W(s), e^{tA}X_0\Big] + \mathbf{Var}\Big[\int_0^t e^{(t-s)A}K\,\mathrm{d}W(s)\Big] \\ &= \mathbf{Var}[e^{tA}X_0] + \int_0^t e^{(t-s)A}K\big(e^{(t-s)A}K\big)^T\,\mathrm{d}s \\ &= e^{tA}\mathbf{Var}[X_0]e^{tA^T} + \int_0^t e^{uA}KK^Te^{uA^T}\,\mathrm{d}u. \end{aligned}$$

In the stationary case, $\mathbf{Var}[X(t)]$ is constant and we can thus let $t \to \infty$. Then the first summand vanishes while the second converges towards

$$\int_0^\infty e^{sA}KK^Te^{sA^T}\,\mathrm{d}s =: \Sigma, \tag{1.10}$$

which is finite since A is stable.

Theorem 1.20. *Let (A, K) be an Ornstein-Uhlenbeck system with A stable. If $X_0 \sim \mathcal{N}(\mathbf{0}, \Sigma)$ with Σ from (1.10), then the Ornstein-Uhlenbeck process X from (1.8) is stationary.*

Proof. For every $t > 0$, the summands from (1.9) have the distributions

$$e^{tA}X_0 \sim \mathcal{N}\Big(e^{tA}\mathbf{E}[X_0], e^{tA}\mathbf{Var}[X_0]e^{tA^T}\Big) = \mathcal{N}\Big(\mathbf{0}, e^{tA}\Sigma e^{tA^T}\Big)$$

and

$$\int_0^t e^{(t-s)A}K \,\mathrm{d}W(s) \sim \mathcal{N}\Big(\mathbf{0}, \int_0^t e^{(t-s)A}KK^T e^{(t-s)A^T}\,\mathrm{d}s\Big).$$

Since these two random variables are independent, the sum's distribution is

$$\begin{aligned}
X(t) &\sim \mathcal{N}\Big(\mathbf{0}, e^{tA}\Sigma e^{tA^T} + \int_0^t e^{(t-s)A}KK^T e^{(t-s)A^T}\,\mathrm{d}s\Big)\\
&= \mathcal{N}\Big(\mathbf{0}, \int_0^\infty e^{tA}e^{sA}KK^T e^{sA^T}e^{tA^T}\,\mathrm{d}s + \int_0^t e^{(t-s)A}KK^T e^{(t-s)A^T}\,\mathrm{d}s\Big)\\
&= \mathcal{N}\Big(\mathbf{0}, \int_t^\infty e^{sA}KK^T e^{sA^T}\,\mathrm{d}s + \int_0^t e^{sA}KK^T e^{sA^T}\,\mathrm{d}s\Big)\\
&= \mathcal{N}(\mathbf{0}, \Sigma).
\end{aligned}$$

Therefore if $X_0 \sim \mathcal{N}(\mathbf{0}, \Sigma)$, then $X(t) \sim \mathcal{N}(\mathbf{0}, \Sigma)$ for each $t \geq 0$.

Given $t > 0$ and $0 < t_1 < t_2 < \ldots < t_n$, (1.9) yields

$$\begin{aligned}
X(t+t_i) &= e^{(t+t_i)A}X_0 + \int_0^{t+t_i} e^{(t+t_i-s)A}K \,\mathrm{d}W(s)\\
&= e^{(t_i-t_1)}\Big(e^{(t+t_1)A}X_0 + \int_0^{t+t_1} e^{(t+t_1-s)A}K\,\mathrm{d}W(s)\Big) + \int_{t+t_1}^{t+t_i} e^{(t+t_i-s)A}K\,\mathrm{d}W(s)\\
&= e^{(t_i-t_1)}X(t+t_1) + \int_{t_1}^{t_i} e^{(t_i-s)A}K\,\mathrm{d}W(s),
\end{aligned}$$

for every i. Consequently the distribution of $\big(X(t_1+t), X(t_2+t), \ldots, X(t_n+t)\big)$ depends only on the distribution of $X(t_1+t)$ and the distribution of the increments of $W(s)$ for $s \in [t_1, t_n]$. Since $X(t+t_1) \sim \mathcal{N}(\mathbf{0}, \Sigma)$ and W has stationary increments, the distribution of $\big(X(t_1+t), X(t_2+t), \ldots, X(t_n+t)\big)$ is independent of t. □

Due to

$$\begin{aligned}
A\Sigma + \Sigma A^T &= \int_0^\infty \Big(Ae^{sA}KK^T e^{sA^T} + e^{sA}KK^T e^{sA^T}A^T\Big)\,\mathrm{d}s = \int_0^\infty \frac{\mathrm{d}}{\mathrm{d}s}\Big(e^{sA}KK^T e^{sA^T}\Big)\,\mathrm{d}s\\
&= \lim_{s\to\infty} e^{sA}KK^T e^{sA^T} - e^{0\cdot A}KK^T e^{0\cdot A^T} = -KK^T,
\end{aligned}$$

Σ is the symmetric, positive semi-definite solution of the *Lyapunov equation*

$$A\Sigma + \Sigma A^T = -KK^T. \tag{1.11}$$

Theorem 1.21. *(i) The equation* (1.11) *has a unique solution Σ if and only if A has no eigenvalue λ such that $-\lambda$ is also an eigenvalue of A. This solution is symmetric.*

(ii) If A is stable (in which case (i) holds), the solution Σ is positive semi-definite.

The existence and uniqueness of Σ are proven, e.g., in [24, Theorem 2.4.4.1]. For the symmetry and positive semi-definiteness see [29, Theorems XI and XII]. ([29] proves that Σ is positive definite if KK^T is positive definite; the positive semi-definite case follows directly via a limit procedure since Σ in (1.11) depends continuously on K as (1.11) is a system of linear equations in the entries of Σ.) Given an eigenvalue decomposition of A, Σ can be computed analytically, as described in [17, Appendix C.2]. (1.6) yields the auto-covariance function of X (using the fact that X_0 and W are independent):

$$\begin{aligned} C_X(s,t) &= \mathbf{Cov}[X(s), X(t)] \\ &= \mathbf{Cov}\big[e^{sA}X_0, e^{tA}X_0\big] + \mathbf{Cov}\bigg[\int_0^s e^{uA}K\,\mathrm{d}W(u), \int_0^t e^{uA}K\,\mathrm{d}W(u)\bigg] \\ &= e^{sA}\mathbf{Var}[X_0]e^{tA^T} + \int_0^{\min\{s,t\}} e^{uA}KK^Te^{uA^T}\,\mathrm{d}u. \end{aligned}$$

In the stationary case $\mathbf{Var}[X(t)] = \Sigma$ implies

$$C_X(t) = C_X(t,0) = e^{tA}\Sigma \qquad \text{for } t \geq 0 \tag{1.12}$$

and $C_X(t) = C_X(-t)^T = \Sigma e^{-tA^T}$ for $t < 0$.

Let X be a stationary Ornstein-Uhlenbeck process and $\widetilde{X}$ an Ornstein-Uhlenbeck process corresponding to the same Ornstein-Uhlenbeck system (A, K) and using the same Brownian motion W, but with a different initial value $\widetilde{X}_0$ with arbitrary distribution. Then

$$\|X(t) - \widetilde{X}(t)\| = \|e^{tA}(X_0 - \widetilde{X}_0)\| \xrightarrow{t\to\infty} 0 \quad \text{almost surely;}$$

in particular (cf. [26, Theorem 13.18]), the distribution of $\widetilde{X}(t)$ converges to $\mathcal{N}(\mathbf{0}, \Sigma)$.

Remark 1.22. Since the finite-dimensional distributions of a Gaussian process X are multivariate normal distributions, they are uniquely determined by its expectation function and its auto-covariance function. Kolmogorov's extension theorem (cf. [26, Theorem 14.36] and [11, p. 33]) implies that for each countable set $Q = \{t_i : i \in \mathbb{N}\}$ the distribution of $(X(t_i))_{i\in\mathbb{N}}$ is uniquely defined by the distributions of all finite tuples $(X(t_i))_{i=1,\dots,N}$, $N \in \mathbb{N}$, and if X is continuous, its path is given by its values $(X(t_i))_{i\in\mathbb{N}}$ if Q is dense in $[0,\infty)$. Hence every stationary, centered, and continuous Gaussian process $X \in \mathbb{R}^d$ with auto-covariance function $C_X(t) = e^{tA}\Sigma$ for $t \geq 0$ (with $A, \Sigma \in \mathbb{R}^{d\times d}$) such that $A\Sigma + \Sigma A^T$ is negative semi-definite is an Ornstein-Uhlenbeck process. For the same reason, every stationary, centered, and continuous Gaussian process $X \in \mathbb{R}^m$ with auto-covariance function $C_X(t) = P^Te^{tA}\Sigma P$ for $t \geq 0$ with arbitrary $P \in \mathbb{R}^{m\times d}$ has the form $X = PY$ with a d-dimensional Ornstein-Uhlenbeck process Y.

Further if a Gaussian process X with such an auto-covariance function is not continuous, there exists an equivalent Gaussian process $\widetilde{X}$ which is continuous, cf. [11, Section 9.2].

Remark 1.23. Instead of defining $X(t)$ for $t \geq 0$ using an initial value X_0, an Ornstein-Uhlenbeck process with stable A can also be defined on the whole real line similarly to (1.9): Extend W to a Brownian motion on $\mathbb{R}$ (see Definition 1.10) and define

$$X(t) := \int_{-\infty}^{t} e^{(t-s)A} K \, \mathrm{d}s.$$

(The filtration to which X is adapted is $(\mathcal{A}_t)_{t\in\mathbb{R}}$ where $\mathcal{A}_t$ is the σ-algebra generated by $(W(s))_{s\in(-\infty,t]}$.) Then according to (1.6) and (1.10),

$$C_X(t) = \int_{-\infty}^{0} e^{(t-s)A} K K^T e^{-sA} \, \mathrm{d}s = e^{tA}\Sigma$$

for $t \geq 0$. Hence, according to Remark 1.22, each sub-process $(X(t + t_0))_{t\in[0,\infty)}$ is a stationary Ornstein-Uhlenbeck process with initial value $X(t_0)$; thus X can be interpreted as a stationary Ornstein-Uhlenbeck process on $\mathbb{R}$.

2. The Generalized Langevin Equation

2.1. Introduction to the Generalized Langevin Equation

2.1.1. Basic Information

The (simple) *Langevin equation* is a physically motivated stochastic differential equation which describes the motion of a macromolecule in a medium. The molecule's velocity $V(t) \in \mathbb{R}^d$ is subject to friction, which slows down the molecule, and collisions with the medium's particles. Exactly simulating these collisions would require simulating the motion of each single particle in the medium. Due to the complex systems and the enormous computational cost this entails, the Langevin equation employs the coarse graining approach of simulating only the macromolecule's motion explicitly while the collisions are modeled by a random term. Here we assume that collisions are independent of each other and only have an immediate impact on the molecule's velocity, but no long-term effects. Collisions are not regarded as a discrete (countable) number of events but as a stochastic process whose impact is modeled by an Itô integral with a Brownian motion. This leads to the following equation for the macromolecule's velocity (cf. [17, Equation (2.1)], [23, Equation (7)]):

$$m\,\mathrm{d}V(t) = -AV(t)\,\mathrm{d}t + K\,\mathrm{d}W(t), \qquad V(0) = V_0 \tag{2.1}$$

where $A \in \mathbb{R}^{d\times d}$ is the friction coefficient and $K \in \mathbb{R}^{d\times n}$. m denotes the macromolecule's mass and d the dimension of V. In many applications A is a diagonal matrix but we do not require this to be the case. V_0 is the (random) initial velocity and $W(t)$ is an n-dimensional Brownian motion which is independent of V_0.

(2.1) is obviously the Ornstein-Uhlenbeck equation (1.7). Consequently if $-A$ is stable, (2.1) has a unique stationary solution, which is a Markov process, and the distribution of $V(t)$ converges towards the stationary distribution for every initial distribution V_0 according to Theorem 1.20. We denote by the name "Ornstein-Uhlenbeck equation" the theoretical mathematical equation without any physical meaning while we call this specific application to molecular dynamics the "Langevin equation".

(2.1) can also describe the motion of several macromolecules, in which case V contains all molecules' coordinates. Therefore $d > 3$ can make sense physically. However, we restrict ourselves to the case where all considered macromolecules have the same mass.

The *generalized Langevin equation (GLE)* expands on this notion. It keeps the idea of coarse grained variables representing the motion of the medium's particles, but in contrast to (2.1), it takes into account that collisions not only influence the macromolecule but

also the other particles, which has an impact on their dynamics and thus potentially on their influence on the macromolecule at a later time. Hence collisions not only have an immediate effect on the velocity but can also influence it on the long term via their impact on the other particles. The generalized Langevin equation deals with this through a convolution term with a memory kernel which encodes the long-term effects of collisions.

This leads to the following equation for the macromolecule's velocity $V(t) \in \mathbb{R}^d$:

$$mV'(t) = -\int_0^t \widetilde{\gamma}(t-s)V(s)\,\mathrm{d}s + \widetilde{F}(t) \qquad \text{for } t > 0, \qquad V(0) = V_0 \tag{2.2}$$

with a memory kernel $\widetilde{\gamma} \in L^1\big([0,\infty), \mathbb{R}^{d\times d}\big)$ and a centered, stationary Gaussian process $\widetilde{F}$ representing the random collisions which is independent of V_0. m again denotes the macromolecule's mass and d denotes the dimension of V. As in (2.1), $d > 3$ can make sense physically if there are several macromolecules whose velocities are combined in V. The distribution of $\widetilde{F}$ is given by a relation to $\widetilde{\gamma}$ known as the *fluctuation-dissipation theorem*, cf. [28], which states that the auto-covariance function of F equals $\widetilde{\gamma}$ up to a constant:

$$C_{\widetilde{F}}(t) := \mathbf{Cov}\big[\widetilde{F}(t), \widetilde{F}(0)\big] = \mathbf{Cov}\big[\widetilde{F}(s+t), \widetilde{F}(s)\big] = \frac{1}{\beta}\widetilde{\gamma}(t) \qquad \text{for } t \geq 0.$$

Here $\beta := \frac{1}{k_B T} > 0$ is the "inverse temperature" where T is the temperature and k_B the Boltzmann constant. In this thesis we will always assume m and β to be known scalar constants. Their exact values are of no importance to us. The notation in (2.2) using the derivative is valid as the right-hand side is continuous and V is therefore almost surely differentiable (contrarily to V from (2.1)) as proven in [35, Theorem 5.6] for the one-dimensional case.

(2.2) can be regarded as a generalization of (2.1) in a distributional sense: By omitting the condition $\widetilde{\gamma} \in L^1\big([0,\infty), \mathbb{R}^{d\times d}\big)$ and instead taking as $\widetilde{\gamma}$ the distribution defined by $\widetilde{\gamma}(f) := Af(0)$, the integral in (2.2) is replaced by $AV(t)$ and $\widetilde{F}$ in (2.2) can be represented by a stationary stochastic process $\widetilde{F}$ of independent and (identically) normally distributed random variables. Since this requires using distributions, (2.1) and (2.2) differ in some analytical properties, such as differentiability and the fact that the solution of (2.2) is no longer a Markov process.

In the remaining thesis we will always use the following rescaled version of the GLE obtained by dividing $\widetilde{F}$, $\widetilde{\gamma}$, and both sides of (2.2) by m:

$$V'(t) = -\int_0^t \gamma(t-s)V(s)\,\mathrm{d}s + F(t) \qquad \text{for } t > 0, \qquad V(0) = V_0 \tag{2.3}$$

with $\gamma(t) := \frac{1}{m}\widetilde{\gamma}(t)$ and $F(t) := \frac{1}{m}\widetilde{F}(t)$. The corresponding fluctuation-dissipation theorem reads

$$C_F(t) = \frac{1}{\beta m}\gamma(t) \qquad \text{for } t \geq 0. \tag{2.4}$$

In the course of this thesis we will also consider the following version of the GLE, which

combines the long-term friction from (2.3) and the instantaneous friction from (2.1):

$$\mathrm{d}V(t) = \left(DV(t) - \int_0^t \gamma(t-s)V(s)\,\mathrm{d}s + F(t)\right)\mathrm{d}t + \frac{1}{\sqrt{\beta m}}L\,\mathrm{d}W(t) \tag{2.5}$$

with $\gamma \in L^1\big([0,\infty), \mathbb{R}^{d\times d}\big)$ and $D, L \in \mathbb{R}^{d\times d}$; cf. also [23, Equation (14)] and [35, Equation (1)]. Concerning the stochastic dependence of F and W, the following two cases are of interest: F and W are independent or F has the form

$$F(t) = \int_{-\infty}^{\infty} \varphi(t-s)\,\mathrm{d}W(s) \tag{2.6}$$

with a function $\varphi \in L^2\big(\mathbb{R}, \mathbb{R}^{d\times d}\big)$ and the same Brownian motion W extended to the whole real line (see Definition 1.10). We restrict ourselves to the second case here; see [21] for a comparison of both cases.

Since the right-hand side of (2.5) equals the sum of the right-hand sides of (2.1) and (2.3) up to constant factors, (2.5) can be considered as a GLE where the memory kernel is the sum of a function $\gamma \in L^1\big([0,\infty), \mathbb{R}^{d\times d}\big)$ and a distribution. Consequently the solution V of (2.5) is no longer differentiable but it is still almost surely continuous under weak conditions on γ (cf. [35, Theorem 5.8] for the case $d = 1$). The fluctuation-dissipation theorem (2.4) has to be reformulated in this case: The relation of the terms $\int_0^t \gamma(t-s)V(s)\,\mathrm{d}s\,\mathrm{d}t$ and $F(t)\,\mathrm{d}t$, which describe the long-term behavior of the GLE, now reads

$$C_F(t) = \frac{1}{\beta m}\gamma_*(t) - \frac{1}{\sqrt{\beta m}}\varphi(t)L^T - \frac{1}{\sqrt{\beta m}}L\varphi(-t)^T \tag{2.7}$$

where

$$\gamma_*(t) = \begin{cases} \gamma(t) & \text{if } t > 0, \\ \frac{1}{2}\big(\gamma(t) + \gamma(-t)^T\big) & \text{if } t = 0, \\ \gamma(-t)^T & \text{if } t < 0. \end{cases}$$

On top of that, the instantaneous terms $DV(t)\,\mathrm{d}t$ and $\frac{1}{\sqrt{\beta m}}L\,\mathrm{d}W(t)$ are related via

$$D + D^T = -LL^T.$$

The latter relation results from (1.11) when inserting $\Sigma = \mathbf{Var}[V(t)] = \frac{1}{\beta m}\mathbf{I}$ since the instantaneous terms describe an Ornstein-Uhlenbeck process. A rigorous proof of these relations may be found in [21, Theorem 2.3].

2.1.2. The Stationary Solution

Just as the (simple) Langevin equation, the generalized Langevin equation has a stationary solution, as mentioned in [28, Section 4], which is a centered stationary Gaussian process as we will see now. We consider the more general equation (2.5) with instantaneous friction; (2.3) is obviously a special case.

Definition and Theorem 2.1. *Let $\gamma \in L^1([0,\infty), \mathbb{R}^{d\times d})$. The equation*

$$r'(t) = Dr(t) - \int_0^t \gamma(t-s)r(s)\,\mathrm{d}s \qquad \text{for } t > 0, \qquad r(0) = \mathbf{I}_d. \tag{2.8}$$

is called the (first kind) integro-differential Volterra equation *(we say briefly* Volterra equation*). This equation has a unique solution r, which is called the* differential resolvent *of the memory kernel γ and the matrix D.*

The existence and uniqueness of the differential resolvent are proven in [19, Theorem 3.3.1]. Note that although (2.8) looks like a deterministic "version" of the generalized Langevin equation, its dimension is $d \times d$. Using the differential resolvent, the solution of (2.5) can be written explicitly as follows, cf. [19, Theorem 3.3.3] and [21, Theorem 2.1]:

Theorem 2.2. *Let r be the differential resolvent of γ. Then*

$$V(t) = r(t)V_0 + \int_0^t r(t-s)F(s)\,\mathrm{d}s + \int_0^t r(t-s)\frac{1}{\sqrt{\beta m}}L\,\mathrm{d}W(s), \tag{2.9}$$

is the unique solution of (2.5) *(up to equivalence of stochastic processes). If V_0 is normally distributed with $\mathbf{E}[V_0] = \mathbf{0}$, V is a centered Gaussian process.*

Proof. Let V be defined as in (2.9). To verify that (2.5) holds, we rewrite the first and second summand as integral of their derivatives and apply Lemma 1.2 to the second summand; for the third term we employ Theorem 1.16:

$$\begin{aligned} V(t) &= r(t)V_0 + \int_0^t r(t-u)F(u)\,\mathrm{d}u + \int_0^t r(t-u)\frac{1}{\sqrt{\beta m}}L\,\mathrm{d}W(u) \\ &= r(0)V_0 + \int_0^t r'(s)V_0\,\mathrm{d}s + \int_0^t \left(\int_0^s r'(s-u)F(u)\,\mathrm{d}u + r(0)F(s)\right)\mathrm{d}s + \\ &\quad r(0)\frac{1}{\sqrt{\beta m}}LW(t) - \int_0^t -r'(t-u)\frac{1}{\sqrt{\beta m}}LW(u)\,\mathrm{d}u. \end{aligned}$$

By inserting (2.8) for each occurrence of r', substituting $w := v+u$, and applying Fubini's theorem, we obtain

$$\begin{aligned} V(t) &= V_0 + \int_0^t \left(Dr(s) - \int_0^s \gamma(s-u)r(u)\,\mathrm{d}u\right)V_0\,\mathrm{d}s + \\ &\quad \int_0^t \left(\int_0^s \left(Dr(s-u) - \int_0^{s-u} \gamma(s-u-v)r(v)\,\mathrm{d}v\right)F(u)\,\mathrm{d}u + F(s)\right)\mathrm{d}s + \\ &\quad \frac{1}{\sqrt{\beta m}}LW(t) + \int_0^t \left(Dr(t-u) - \int_0^{t-u} \gamma(t-u-v)r(v)\,\mathrm{d}v\right)\frac{1}{\sqrt{\beta m}}LW(u)\,\mathrm{d}u \\ &= V_0 + \int_0^t \left(Dr(s) - \int_0^s \gamma(s-u)r(u)\,\mathrm{d}u\right)V_0\,\mathrm{d}s + \\ &\quad \int_0^t \left(\int_0^s \left(Dr(s-u) - \int_u^s \gamma(s-w)r(w-u)\,\mathrm{d}w\right)F(u)\,\mathrm{d}u + F(s)\right)\mathrm{d}s + \\ &\quad \frac{1}{\sqrt{\beta m}}LW(t) + \int_0^t \left(Dr(t-u) - \int_u^t \gamma(t-w)r(w-u)\,\mathrm{d}w\right)\frac{1}{\sqrt{\beta m}}LW(u)\,\mathrm{d}u \end{aligned}$$

$$
\begin{aligned}
&= V_0 + \int_0^t Dr(s)V_0 \,\mathrm{d}s - \int_0^t \int_0^s \gamma(s-u)r(u)V_0 \,\mathrm{d}u \,\mathrm{d}s + \\
&\int_0^t \left(\int_0^s Dr(s-u)F(u) \,\mathrm{d}u - \int_0^s \gamma(s-w) \int_0^w r(w-u)F(u) \,\mathrm{d}u \,\mathrm{d}w + F(s) \right) \mathrm{d}s + \\
&\frac{1}{\sqrt{\beta m}} LW(t) + \int_0^t Dr(t-u) \frac{1}{\sqrt{\beta m}} LW(u) \,\mathrm{d}u - \\
&\int_0^t \gamma(t-w) \int_0^v r(w-u) \frac{1}{\sqrt{\beta m}} LW(u) \,\mathrm{d}u \,\mathrm{d}w.
\end{aligned}
$$

We rewrite the last three summands as follows: For the first of them, Theorem 1.16 implies

$$
\frac{1}{\sqrt{\beta m}} LW(t) = \int_0^t \frac{1}{\sqrt{\beta m}} L \,\mathrm{d}W(s).
$$

Rewriting the next summand as the integral of its derivative via Lemma 1.2 and subsequently employing Theorem 1.16 yields

$$
\begin{aligned}
&\int_0^t Dr(t-u) \frac{1}{\sqrt{\beta m}} LW(u) \,\mathrm{d}u \\
&\quad = \int_0^t \left(\int_0^s Dr'(s-u) \frac{1}{\sqrt{\beta m}} LW(u) \,\mathrm{d}u + Dr(0) \frac{1}{\sqrt{\beta m}} LW(s) \right) \mathrm{d}s \\
&\quad = \int_0^t \int_0^s Dr(s-u) \frac{1}{\sqrt{\beta m}} L \,\mathrm{d}W(u) \,\mathrm{d}s.
\end{aligned}
$$

In the last summand we rewrite the inner integral in the same way:

$$
\begin{aligned}
&\int_0^t \gamma(t-w) \int_0^w r(w-u) \frac{1}{\sqrt{\beta m}} LW(u) \,\mathrm{d}u \,\mathrm{d}w \\
&\quad = \int_0^t \gamma(t-w) \int_0^w \left(\int_0^v r'(v-u) \frac{1}{\sqrt{\beta m}} LW(u) \,\mathrm{d}u + r(0) \frac{1}{\sqrt{\beta m}} LW(v) \right) \mathrm{d}v \,\mathrm{d}w \\
&\quad = \int_0^t \gamma(t-w) \int_0^w \int_0^v r(v-u) \frac{1}{\sqrt{\beta m}} L \,\mathrm{d}W(u) \,\mathrm{d}v \,\mathrm{d}w.
\end{aligned}
$$

Employing Fubini's theorem twice and substituting $s := t + v - w$ in-between yields

$$
\begin{aligned}
&\int_0^t \gamma(t-w) \int_0^w \int_0^v r(v-u) \frac{1}{\sqrt{\beta m}} L \,\mathrm{d}W(u) \,\mathrm{d}v \,\mathrm{d}w \\
&\quad = \int_0^t \int_v^t \gamma(t-w) \int_0^v r(v-u) \frac{1}{\sqrt{\beta m}} L \,\mathrm{d}W(u) \,\mathrm{d}w \,\mathrm{d}v \\
&\quad = \int_0^t \int_v^t \gamma(s-v) \int_0^v r(v-u) \frac{1}{\sqrt{\beta m}} L \,\mathrm{d}W(u) \,\mathrm{d}s \,\mathrm{d}v \\
&\quad = \int_0^t \int_0^s \gamma(s-v) \int_0^v r(v-u) \frac{1}{\sqrt{\beta m}} L \,\mathrm{d}W(u) \,\mathrm{d}v \,\mathrm{d}s.
\end{aligned}
$$

By inserting these results yields, we obtain

$$\begin{aligned}
V(t) &= V_0 + \int_0^t Dr(s)V_0 \,\mathrm{d}s - \int_0^t \int_0^s \gamma(s-u)r(u)V_0 \,\mathrm{d}u \,\mathrm{d}s + \\
&\quad \int_0^t \left(\int_0^s Dr(s-u)F(u) \,\mathrm{d}u - \int_0^s \gamma(s-w) \int_0^w r(w-u)F(u) \,\mathrm{d}u \,\mathrm{d}w + F(s) \right) \mathrm{d}s + \\
&\quad \int_0^t \frac{1}{\sqrt{\beta m}} L \,\mathrm{d}W(s) + \int_0^t \int_0^s Dr(s-u) \frac{1}{\sqrt{\beta m}} L \,\mathrm{d}W(u) \,\mathrm{d}s - \\
&\quad \int_0^t \int_0^s \gamma(s-v) \int_0^v r(v-u) \frac{1}{\sqrt{\beta m}} L \,\mathrm{d}W(u) \,\mathrm{d}v \,\mathrm{d}s \\
&= V_0 + D \int_0^t \left(r(s)V_0 + \int_0^s r(s-u)F(u) \,\mathrm{d}u + \int_0^s r(s-u) \frac{1}{\sqrt{\beta m}} L \,\mathrm{d}W(u) \right) \mathrm{d}s - \\
&\quad \int_0^t \int_0^s \gamma(s-u) \Bigg(r(u)V_0 + \int_0^u r(u-v)F(v) \,\mathrm{d}v + \\
&\qquad\qquad \int_0^u r(u-v) \frac{1}{\sqrt{\beta m}} L \,\mathrm{d}W(v) \Bigg) \mathrm{d}u \,\mathrm{d}s + \\
&\quad \int_0^t F(s) \,\mathrm{d}s + \int_0^t \frac{1}{\sqrt{\beta m}} L \,\mathrm{d}W(s) \\
&= V_0 + D \int_0^t V(s) \,\mathrm{d}s - \int_0^t \int_0^s \gamma(s-u)V(u) \,\mathrm{d}u \,\mathrm{d}s + \int_0^t F(s) \,\mathrm{d}s + \\
&\quad \int_0^t \frac{1}{\sqrt{\beta m}} L \,\mathrm{d}W(s),
\end{aligned}$$

which is the integral version of (2.5).

If V_0 is normally distributed with $\mathbf{E}[V_0] = \mathbf{0}$, (2.9) directly implies that V is a centered Gaussian process.

If there are two solutions of (2.5), their difference $\widetilde{V}$ satisfies

$$\widetilde{V}'(t) = Dr(t) - \int_0^t \gamma(t-s)\widetilde{V}(s) \,\mathrm{d}s \qquad \text{for } t > 0, \qquad \widetilde{V}(0) = \mathbf{0}_d.$$

Due to the uniqueness of the differential resolvent, the stochastic process $\widetilde{V}$ is equivalent to a deterministic process which is a linear combination of the columns of r. Due to $\widetilde{V}(0) = \mathbf{0}_d$, it is the trivial linear combination, $\widetilde{V}(t) = \mathbf{0}_d$ for all t. □

If V_0 is chosen appropriately, the process auto-covariance function of V satisfies

$$C_V(t,s) = \frac{1}{\beta m} r(t-s) \qquad \text{for } t \geq s \geq 0$$

and is therefore stationary:

Theorem 2.3. *Let r be the differential resolvent of γ and D. If $V_0 \sim \mathcal{N}(\mathbf{0}, \frac{1}{\beta m}\mathbf{I})$, the process V given by (2.5) is stationary with $C_V(t) = \frac{1}{\beta m} r(t)$ for $t \geq 0$.*

We omit the proof hereof and refer to [21, Theorem 2.3] instead.

Remark 2.4. According to [19, Theorem 3.3.17], $r \in L^1([0,\infty), \mathbb{R}^{d\times d})$ if and only if $r(t) \xrightarrow{t\to\infty} \mathbf{0}$. In this case, comparing (2.9) for the initial distribution from Theorem 2.3 and for any other initial distribution (using the same paths F and W) shows that the difference converges to $\mathbf{0}$ almost surely; consequently for any initial distribution, the distribution of $V(t)$ converges towards the stationary distribution.

Theorem 2.3 implies that C_V is also the unique solution of a first kind Volterra equation with an adapted initial condition:

$$C_V'(t) = DC_V(t) - \int_0^t \gamma(t-s)C_V(s)\,\mathrm{d}s \qquad \text{for } t > 0, \qquad C_V(0) = \frac{1}{\beta m}\mathbf{I}_d. \tag{2.10}$$

While both V and F have a stationary distribution, the joint process $X := \begin{bmatrix} V \\ F \end{bmatrix}$ does not, cf. [23, Proposition 2.8]. To remedy this shortcoming, one can extend F to a stationary stochastic process with domain $\mathbb{R}$ via (2.6) and consider the following stationary version of (2.5), where the integral starts at $-\infty$ instead of 0:

$$\mathrm{d}V(t) = \left(DV(t) - \int_{-\infty}^t \gamma(t-s)V(s)\,\mathrm{d}s + F(t)\right)\mathrm{d}t + \frac{1}{\sqrt{\beta m}}L\,\mathrm{d}W(t), \qquad t \in \mathbb{R}. \tag{2.11}$$

The joint process X in (2.11) has a stationary distribution, which is a centered Gaussian process. In many applications $C_{F,V}(t)$ is assumed to be 0 for $t > 0$, but this is not the case for the stationary GLE (2.11) as is proven in [23, Theorem 2.7], i.e. $V(s)$ and $F(t)$ are not uncorrelated for $t > s$ (in contrast to (2.5)). It can be proven that C_V from (2.11) satisfies (2.10) (with the same integral limits), too, cf. [20]. For $t \to \infty$, the difference between the solutions of (2.5) and (2.11) using the same process F vanishes (regardless of V_0) in the sense of L^2-convergence, cf. [21, Theorem 4.1].

While the existence of a stationary solution of (2.11) is a useful and desirable property, this version of the GLE is unsuited for practical simulation as evaluating the integral for $t > 0$ requires an already existing path $(X(t))_{t\in(-\infty,0)}$ instead of a single initial value. Therefore numerical simulation is only possible for (2.5). Since the difference of the solutions of (2.5) and (2.11) becomes negligible for $t \to \infty$, one can obtain a good approximation to (2.11) by discarding the simulation results up to some "equilibration time" $t_0 > 0$ and only using $(V(t))_{t\geq t_0}$. For the remaining thesis we will therefore not pay much attention to the difference between (2.5) and (2.11) and work only with (2.5) for simplicity.

2.1.3. The Inverse Problem

We are interested in the problem of determining the memory kernel in (2.5) corresponding to a given velocity auto-covariance function (VACF). To achieve this, several methods have already been proposed: The classical method consists in determining γ from the

Volterra equation (2.10) numerically by discretizing the integral, resulting in a system of linear equations which can subsequently be solved, cf. e.g. [47, 30]. The system's matrix is a triangular matrix and it can thus be solved efficiently.

However, the corresponding matrix often has a large condition. After rewriting (2.10) as

$$C_V'(t) = DC_V(t) - \int_0^t \gamma(s) C_V(t-s)\,\mathrm{d}s$$

for $t > 0$, differentiating via Lemma 1.2 leads to the *second kind Volterra equation*

$$C_V''(t) = DC_V'(t) - \frac{1}{\beta m}\gamma(t) - \int_0^t \gamma(t-s) C_V'(s)\,\mathrm{d}s \qquad \text{for } t \geq 0 \tag{2.12}$$

with $C_V(0) = \frac{1}{\beta m}\mathbf{I}_d$, $C_V'(0) = D$. This equation can also be discretized, resulting in another system of linear equations (again with a triangular matrix) which allows determining γ. Solving (2.12) is more stable than (2.10) (cf. [20, p. 485], [47, p. 319]) but requires numerically computing the second derivative of the data if it is not available, which can introduce an additional numerical error. [33] presents numerical methods for solving the first and second kind Volterra equation.

In physical simulations, however, the derivatives of C_V are often known since $C_V'(t) = C_{V',V}(t)$ and $C_V''(t) = C_{V'}(t)$, where V' is the total force acting on the particle (up to scaling by the mass m), which can be directly observed. This fact is also exploited by [47]. It is also possible to take the Laplace transform of both sides of (2.10) and employ the convolution theorem in order to obtain an equation for $\mathcal{L}\gamma$ as described e.g. in [32, p. 14184] although the numerical Laplace transform and inverse Laplace transform can introduce additional numerical errors, cf. [47, p. 319]. [25] proposes another method, which employs an iterative reconstruction of the memory kernel. A second iterative method, which is also applicable to the non-stationary case, is described in [36].

2.2. The Generalized Langevin Equation as an Ornstein-Uhlenbeck Process

2.2.1. Prony Series

Simulating a path of (2.3) or (2.5) numerically requires the numerical evaluation of the convolution integral in each time step. This is expensive, resulting in a simulation time of $O(n^2)$ for n simulation steps. Reducing the computational cost to $O(n)$ is possible by cutting off the kernel after some time, but this introduces an additional error and can still entail large computational costs depending on where the kernel is cut off. This is a problem especially for slowly decaying memory kernels. Therefore the question arises if the generalized Langevin equation can be approximated by a simpler and less expensive process.

Indeed this is possible if the memory kernel has the form

$$\gamma(t) = B^T e^{tA_0} C \quad \text{with} \quad B, C \in \mathbb{R}^{n\times d}, \quad A_0 \in \mathbb{R}^{n\times n}, \quad n > d \tag{2.13}$$

where we require B, C, and A_0 to have full rank. In this case we can introduce an auxiliary process Z and set

$$X(t) := \begin{bmatrix} V(t) \\ Z(t) \end{bmatrix} \in \mathbb{R}^{(d|n)}$$

with the velocity V from (2.5) such that X is an Ornstein-Uhlenbeck process (1.7) with certain matrices $A \in \mathbb{R}^{(d+n)\times(d+n)}$ and $K \in \mathbb{R}^{(d+n)\times d}$, as we will see in Section 2.2.2; cf. also [17] and [34, Theorem 1]. The process Z contains auxiliary variables representing the memory term as well as the random force F. (Representing both terms by the same auxiliary variables is possible due to the fluctuation-dissipation relation (2.7).) The auxiliary variables have no physical meaning for themselves. The Ornstein-Uhlenbeck process (1.7) is far easier to simulate than (2.5). Numerical simulation is possible using e.g. the Euler-Maruyama method or even by sampling paths of its exact solution, see Theorem 1.18.

If γ does not have the form (2.13) but can be approximated by such a function, an Ornstein-Uhlenbeck process can nonetheless be a good approximation of the solution of (2.5). In this thesis we are interested in finding an Ornstein-Uhlenbeck process such that the auto-covariance function of the V component approximates a given velocity auto-covariance function of a generalized Langevin equation.

If A_0 has pairwise different eigenvalues $\lambda_1, \dots, \lambda_n \in \mathbb{C}$ and $A_0 = U\Lambda U^{-1}$ is an eigenvalue decomposition of A_0, $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$, (2.13) means that each component of γ has the form

$$\gamma_{i,j}(t) = \sum_{k=1}^{n} b_{i,k} c_{k,j} e^{\lambda_k t} \tag{2.14}$$

with $b_{i,j} = (B^T U)_{i,j}$, $c_{i,j} = (U^{-1}C)_{i,j}$, which is a special case of a *Prony series*

$$\gamma(t) = \sum_{k=1}^{n} w_k e^{\lambda_k t}, \qquad w_k \in \mathbb{C}^{d\times d}, \tag{2.15}$$

i.e. a linear combination of exponential functions (with $\lambda_k \in \mathbb{C}$). If the original memory kernel has this form, the Ornstein-Uhlenbeck representation of V is exact, meaning that the paths V from (2.5) and from (1.7) (where V denotes the first d components of X) are equivalent when using the same V_0 and W. $\gamma \in L^1([0,\infty), \mathbb{R}^{d\times d})$ holds if A_0 is stable.

If $d > 1$, the coefficient matrix $w_k = [b_{i,k}c_{k,j}]_{i,j=1,\dots,d}$ of (2.14) can be written as the product of the vectors $[b_{i,k}]_{i=1,\dots,d}$ and $[c_{k,j}]_{j=1,\dots,d}$ and therefore has rank 1 for each fixed k. If a coefficient matrix of a Prony series (2.15) has higher rank, it can be written as a sum of several rank 1 matrices. Consequently every Prony series can be written in the form $B^T e^{tA_0} C$ where the multiplicity of an eigenvalue λ_k in A_0 equals the rank of its coefficient matrix.

If the eigenvalues of A_0 are not pairwise different, it is also possible (depending on the Jordan form of A_0) that γ has the form

$$\gamma(t) = \sum_{k=1}^{\nu} \sum_{l=0}^{\nu_k - 1} w_{k,l} t^l e^{\lambda_k t}$$

with coefficients $w_{k,l} \in \mathbb{C}^{d\times d}$ depending on B and C where ν denotes the number of pairwise different eigenvalues of A_0 and ν_k the maximum size of a Jordan block of A_0 corresponding to the eigenvalue λ_k. We will, however, restrict ourselves to the case of a diagonalizable A_0.

If γ has the form (2.13), then $C_X(t) = e^{tA}\Sigma$ and $C_V(t) = P^T e^{tA}\Sigma P$ with $P = [\mathbf{I}_d, \mathbf{0}]^T$. Consequently C_V is a Prony series, too, whose exponents λ_k are the eigenvalues of A. Conversely, if C_V is such Prony series, γ has the form (2.13). Usually we do not know the exact form of γ but we can still try to approximate the given function C_V by a Prony series.

Approximation of functions via Prony series has already been extensively studied. The "classical" approach, the so-called *Prony method*, dates back to 1795 and is described e.g. in [42, Section 10.1]. To apply this method, one first needs to determine the memory kernel of the studied generalized Langevin equation, which one then approximates by a Prony series. Subsequently an Ornstein-Uhlenbeck system is computed from the Prony series. Algorithms which circumvent the intermediate step of determining the memory kernel have been proposed in [45, 8, 44, 38, 13]. The application of the ESPRIT method from [45] to the problem of Prony series approximation is described e.g. in [43]. All of these methods allow for approximating a function by a Prony series and can therefore by applied to both the VACF and the memory kernel; when applying them to the VACF, the memory kernel does not need to be determined. However, these methods still only determine a Prony series approximation of the velocity auto-covariance function, which then needs to be used in a second step to obtain a corresponding Ornstein-Uhlenbeck system. [17] proposed a method which directly generates an Ornstein-Uhlenbeck representation by employing a maximum likelihood estimator for the matrix entries and a Kalman filter.

In this thesis we introduce another method which directly generates an Ornstein-Uhlenbeck system whose velocity auto-covariance function (VACF) approximates the given VACF in a given equidistant grid. It is not necessary to explicitly determine the memory kernel first. While we already treated the case $d = 1$ in [10], we will present a slightly improved version here and extend it to the multidimensional case.

2.2.2. Representing the Generalized Langevin Equation by an Ornstein-Uhlenbeck Process

If γ from the GLE (2.3) is a Prony series (2.13), V can be represented by an Ornstein-Uhlenbeck process (1.7). For this we introduce an auxiliary process $Z = (Z(t))_{t\in[0,\infty)}$ and consider the joint process $X := \begin{bmatrix} V \\ Z \end{bmatrix}$, cf. [17]. Let $P := \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(d|n)\times d}$ be the projection from X onto V, where n is the dimension of Z. It is always possible to choose the basis of Z such that the covariance matrix Σ of the stationary distribution of X has the form

$$\Sigma = \frac{1}{\beta m}\begin{bmatrix} \mathbf{I}_d & \mathbf{0} \\ \mathbf{0} & S \end{bmatrix} \in \mathbb{R}^{(d|n)\times(d|n)} \tag{2.16}$$

with S symmetric and positive definite, i.e. $V(t)$ and $Z(t)$ are independent for every $t \geq 0$ with $C_V(t) = P^T e^{tA} \Sigma P = \frac{1}{\beta m} P^T e^{tA} P$. (The components of a multidimensional normally distributed random variable are independent if and only if they are uncorrelated.) Therefore we will always require Σ to have the form (2.16).

Theorem 2.5. *Let $V = (V(t))_{t\in\mathbb{R}}$ be the stationary solution of the generalized Langevin equation* (2.3) *with a memory kernel γ of the form* (2.13)*, and let*

$$A := \begin{bmatrix} \mathbf{0} & B^T \\ -C & A_0 \end{bmatrix} \in \mathbb{R}^{(d|n)\times(d|n)} \tag{2.17}$$

with $n \geq d$ where B, C, and A_0 are such that the Lyapunov equation (1.11) *has a solution*

$$K = \frac{1}{\sqrt{\beta m}} \begin{bmatrix} \mathbf{0} \\ G \end{bmatrix} \in \mathbb{R}^{(d|n)\times d} \tag{2.18}$$

and Σ from (2.16) *symmetric and positive definite. Then there exists a stochastic process $Z = (Z(t))_{t\in\mathbb{R}}$ such that the joint process X with $X(t) := \begin{bmatrix} V(t) \\ Z(t) \end{bmatrix} \in \mathbb{R}^{(d|n)}$ is equivalent to an Ornstein-Uhlenbeck process* (1.7) *with drift matrix A and diffusion matrix K from* (2.17) *and* (2.18)*.*

Proof. We first note that (1.11) is in this case equivalent to the *singular Lur'e equations*

$$\begin{aligned} A_0 S + S A_0^T &= -GG^T \\ SB - C &= \mathbf{0} \end{aligned} \tag{2.19}$$

with unknown G and S as one can directly verify by inserting (2.16) and (2.17) into (1.11). Let F be the Gaussian process from (2.3). (2.4) and $SB = C$ imply

$$C_F(t) = \frac{1}{\beta m}\gamma(t) = \frac{1}{\beta m} B^T e^{tA_0} SB$$

for $t \geq 0$. According to Remark 1.22, there exists a stationary Ornstein-Uhlenbeck process $\widetilde{F}$ such that $F = B^T \widetilde{F}$, i.e.

$$\mathrm{d}\widetilde{F}(t) = A_0 \widetilde{F}(t)\,\mathrm{d}t + \frac{1}{\sqrt{\beta m}} G\,\mathrm{d}W(t),$$

with $C_{\widetilde{F}}(t) = \frac{1}{\beta m} e^{tA_0} S$ for $t \geq 0$.

Let $\widetilde{Z}(t) := \int_0^t e^{(t-s)A_0} C V(s)\,\mathrm{d}s$ for $t \geq 0$. This implies

$$V'(t) = -\int_0^t B^T e^{(t-s)A_0} C V(s)\,\mathrm{d}s + F(t) = -B^T \widetilde{Z}(t) + B^T \widetilde{F}(t).$$

Lemma 1.2 yields

$$\widetilde{Z}'(t) = \int_0^t A_0 e^{(t-s)A_0} C V(s)\,\mathrm{d}s + CV(t) = A_0\widetilde{Z}(t) + CV(t)$$

and thus the joint process can be written as an Ornstein-Uhlenbeck process

$$\mathrm{d}\begin{bmatrix} V(t) \\ \widetilde{Z}(t) \\ \widetilde{F}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{0} & -B^T & B^T \\ C & A_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & A_0 \end{bmatrix} \begin{bmatrix} V(t) \\ \widetilde{Z}(t) \\ \widetilde{F}(t) \end{bmatrix} \mathrm{d}t + \frac{1}{\sqrt{\beta m}} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ G \end{bmatrix} \mathrm{d}W(t)$$

or equivalently

$$\mathrm{d}\begin{bmatrix} V(t) \\ Z(t) \end{bmatrix} = \begin{bmatrix} \mathbf{0} & B^T \\ -C & A_0 \end{bmatrix} \begin{bmatrix} V(t) \\ Z(t) \end{bmatrix} \mathrm{d}t + \frac{1}{\sqrt{\beta m}} \begin{bmatrix} \mathbf{0} \\ G \end{bmatrix} \mathrm{d}W(t)$$

where $Z(t) := -\widetilde{Z}(t) + \widetilde{F}(t)$. Therefore $X(t) = \begin{bmatrix} V(t) \\ Z(t) \end{bmatrix}$ is an Ornstein-Uhlenbeck process with drift matrix A and diffusion matrix K as in (2.17) and (2.18). □

While we need the auxiliary variables Z from Theorem 2.5, they are only used to define the Ornstein-Uhlenbeck system. They do not have any physical meaning and cannot be observed in a physical simulation of the generalized Langevin equation.

Theorem 2.5 can be extended to include the extra terms in (2.5):

Theorem 2.6. *Let $V = (V(t))_{t\in\mathbb{R}}$ be the stationary solution of the generalized Langevin equation* (2.5) *with a memory kernel γ of the form* (2.13)*. Let further*

$$A := \begin{bmatrix} D & B^T \\ -C & A_0 \end{bmatrix} \in \mathbb{R}^{(d|n)\times(d|n)} \quad \text{and} \quad K = \frac{1}{\sqrt{\beta m}} \begin{bmatrix} L \\ G \end{bmatrix} \in \mathbb{R}^{(d|n)\times d} \tag{2.20}$$

with $n \geq d$, where and D and L are the matrices from (2.5) *and B, C, A_0, and G are such that the Lyapunov equation* (1.11) *has a symmetric and positive definite solution Σ with the form* (2.16)*. Then there exists a stochastic process $Z = (Z(t))_{t\in\mathbb{R}}$ such that the joint process $X(t) := \begin{bmatrix} V(t) \\ Z(t) \end{bmatrix} \in \mathbb{R}^{(d|n)}$ is equivalent to an Ornstein-Uhlenbeck process* (1.7) *with drift matrix A and diffusion matrix K. φ from* (2.7) *is given by*

$$\varphi(t) = \begin{cases} \frac{1}{\sqrt{\beta m}} B^T e^{tA_0} G & \text{if } t > 0, \\ \frac{1}{2\sqrt{\beta m}} B^T G & \text{if } t = 0, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Proof. We first prove that F given by (2.6) satisfies (2.7) if φ is chosen this way.

In this case (2.20) implies that (1.11) is equivalent to the *regular Lur'e equations*

$$A_0 S + S A_0^T = -GG^T \tag{2.21a}$$
$$SB - C = -GL^T \tag{2.21b}$$
$$D + D^T = -LL^T \tag{2.21c}$$

with unknown matrices G, L, and S. Inserting φ into (2.6) yields

$$F(t) = \int_{-\infty}^{\infty} \varphi(t-s)\,\mathrm{d}W(s) = \frac{1}{\sqrt{\beta m}} \int_{-\infty}^{t} B^T e^{(t-s)A_0} G\,\mathrm{d}W(s).$$

Since (2.21a) is another Lyapunov equation, its solution S can be written via (1.10) as $S = \int_0^\infty e^{sA_0} GG^T e^{sA_0}\,\mathrm{d}s$. Together with (1.6), this implies

$$C_F(t) = \frac{1}{\beta m} \int_{-\infty}^{0} B^T e^{(t-s)A_0} GG^T e^{-sA_0} B\,\mathrm{d}s = \frac{1}{\beta m} B^T e^{tA_0} SB$$

for $t \geq 0$. Consequently F can again be written as $F = B^T \widetilde{F}$ with an Ornstein-Uhlenbeck process $\widetilde{F}$ on $\mathbb{R}$ (see Remark 1.23). Inserting (2.21b) shows that F satisfies (2.7):

$$\begin{aligned} C_F(t) &= \frac{1}{\beta m} B^T e^{tA_0} SB = \frac{1}{\beta m} B^T e^{tA_0} C - \frac{1}{\beta m} B^T e^{tA_0} GL^T \\ &= \frac{1}{\beta m} \gamma_*(t) - \frac{1}{\sqrt{\beta m}} \varphi(t) L^T - \frac{1}{\sqrt{\beta m}} L \varphi(-t)^T \end{aligned}$$

for $t > 0$. (One easily verifies that (2.7) also holds for $t < 0$ and $t = 0$.) (2.21c) is the fluctuation-dissipation relation of the instantaneous friction terms, which thus holds, too.

We now continue analogously to the proof of Theorem 2.5. Define $\widetilde{Z}$ as there; consequently

$$\begin{aligned} \mathrm{d}V(t) &= \left(DV(t) - \int_0^t B^T e^{(t-s)A_0} CV(s)\,\mathrm{d}s + F(t) \right) \mathrm{d}t + \frac{1}{\sqrt{\beta m}} L\,\mathrm{d}W(t) \\ &= \left(DV(t) - B^T \widetilde{Z}(s) + B^T \widetilde{F}(t) \right) \mathrm{d}t + \frac{1}{\sqrt{\beta m}} L\,\mathrm{d}W(t) \end{aligned}$$

implies that the joint process is an Ornstein-Uhlenbeck process

$$\mathrm{d} \begin{bmatrix} V(t) \\ \widetilde{Z}(t) \\ \widetilde{F}(t) \end{bmatrix} = \begin{bmatrix} D & -B^T & B^T \\ C & A_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & A_0 \end{bmatrix} \begin{bmatrix} V(t) \\ \widetilde{Z}(t) \\ \widetilde{F}(t) \end{bmatrix} \mathrm{d}t + \frac{1}{\sqrt{\beta m}} \begin{bmatrix} L \\ \mathbf{0} \\ G \end{bmatrix} \mathrm{d}W(t).$$

Setting $Z(t) := -\widetilde{Z}(t) + \widetilde{F}(t)$ yields the desired Ornstein-Uhlenbeck process representation with the matrices A and K from (2.20). □

Remark 2.7. V, $\widetilde{Z}$, and $\widetilde{F}$ from Theorem 2.5 (or Theorem 2.6) have a joint stationary distribution according to Theorem 1.20 and hence V and F have a joint stationary distribution, too, which at first glance seems to contradict our remark in Section 2.1.2 that no joint stationary distribution exists. For the stationary distribution, however,

$\widetilde{Z}(0) \neq 0$ holds almost surely, which contradicts the definition of $\widetilde{Z}$. Hence this distribution is no valid initial distribution.

Analogously to Theorems 2.5 and 2.6, the solution of the stationary GLE (2.11) can be represented by an Ornstein-Uhlenbeck process where $\widetilde{Z}(t) := \int_{-\infty}^{t} e^{(t-s)A_0} CV(s)\,\mathrm{d}s$. Since $\widetilde{Z}(0) = 0$ is no longer necessary in this case, the joint stationary distribution of V, $\widetilde{Z}$, and $\widetilde{F}$ is a valid distribution in this case.

In the remaining thesis, we denote by C_V the velocity auto-covariance function of the solution of the generalized Langevin equation while C_V^A denotes the velocity auto-covariance function of an Ornstein-Uhlenbeck process (1.7) with drift matrix A used to approximate C_V (i.e. the auto-covariance function of the velocity components V).

2.2.3. Functions of Positive Type and the Lur'e Equations

The question whether a solution (G, L, S) of (2.19) or (2.21) (with S symmetric and positive definite) exists is not trivial. A necessary condition is that all eigenvalues of A have non-positive real part: Otherwise let x be an eigenvector of A^T corresponding to an eigenvalue with positive real part. Then multiplying both sides of (1.11) by x^* from the left and x from the right results in a positive left-hand side and non-positive right-hand side. We will thus require A to be stable in this section. (If A has pure imaginary eigenvalues, the corresponding Lur'e equations may be solvable, but since the resulting Ornstein-Uhlenbeck process has no stationary solution, this case is not of interest to us.) We will now formulate a necessary and sufficient condition under which a solution of (2.19) or (2.21) exists.

In this thesis we will usually consider the Lyapunov equation where both $K \in \mathbb{R}^{(d+n)\times d}$ and $\Sigma \in \mathbb{R}^{(d+n)\times(d+n)}$ are unknown and Σ has the form (2.16). There are also applications where the Lyapunov equation with known K (which may have more than d columns) is of interest. While the latter version can be solved as a system of $(n+d)^2$ linear equations or using the method from [17, Appendix C.2], this is not possible if K is unknown.

Definition 2.8. *A function $f\colon \mathbb{R} \to \mathbb{C}^{d\times d}$ is called* of positive type *if $f(-t) = f(t)^*$ for all $t \in \mathbb{R}$ and*

$$\sum_{k,l=1}^{n} x_k^* f(t_k - t_l) x_l \geq 0$$

for every $n \in \mathbb{N}$ and all $t_1, \ldots, t_n \in \mathbb{R}$ and $x_1, \ldots, x_n \in \mathbb{C}^d$.

Such functions are also called *positive definite* but we will not use this name to avoid confusion with positive definite matrices. By approximating integrals of continuous functions with compact support by sums and vice versa and taking into account that such functions are dense in $L^2(\mathbb{R}, \mathbb{C}^d)$, one can prove that a continuous function $f \in L^1_{\mathrm{loc}}(\mathbb{R}, \mathbb{C}^{d\times d})$ is of positive type if and only if $f(-t) = f(t)^*$ for all $t \in \mathbb{R}$ and

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \varphi(t)^* f(t-s)\varphi(s)\,\mathrm{d}s\,\mathrm{d}t \geq 0$$

for every $\varphi \in L^2(\mathbb{R}, \mathbb{C}^d)$, as is done in [46, Section 1.4.3] (cf. also [9, Section 8.1]).

Lemma 2.9. *The auto-covariance function of a stationary stochastic process X with $X(t) \in L^2(\mathbf{P})$ for each t is always a function of positive type.*

Proof. Let X be a stochastic process. We assume that $\mathbf{E}[X] = \mathbf{0}$; otherwise we consider $X - \mathbf{E}[X]$ instead of X. For $t_1, \dots, t_n \in \mathbb{R}$, $x_1, \dots, x_n \in \mathbb{C}^d$ we have (cf. [40, Lemma 1.2])

$$\sum_{k,l=1}^{n} x_k^* C_X(t_k - t_l) x_l = \sum_{k,l=1}^{n} x_k^* \mathbf{E}[X_{t_k} X_{t_l}^T] x_l = \mathbf{E}\left[\left|\sum_{k=1}^{n} X_{t_k}^T x_k\right|^2\right] \geq 0. \qquad \square$$

Another characterization of functions of positive type is given by Bochner's theorem.

Theorem 2.10 (Bochner)**.** *Let $f \in L^1(\mathbb{R}, \mathbb{R}^{d\times d})$ be continuous with Fourier transform $\mathcal{F}f$. Then f is of positive type if and only if $\mathcal{F}f(\xi)$ is a hermitian and positive semi-definite matrix for each $\xi \in \mathbb{R}$.*

The proof of the necessity of f being of positive type is rather simple, cf. [9, Section 8.2] and [46, Section 1.4.2(b)]. (The generalization to matrix-valued functions f is straightforward.) The proof of sufficiency can be found in [19, p. 498]. In the case $d = 1$, Theorem 2.10 is often stated in a measure theoretic context, which states that a given function g is the characteristic function of a finite measure μ, i.e. $g(t) = \int_{\mathbb{R}} e^{\mathrm{i}tx} \mu(\mathrm{d}x)$, if and only if g is of positive type. In our formulation above, $\mathcal{F}f$ is the measure's non-negative (in the case $d = 1$) probability density function while f is its characteristic function.

Remark 2.11. If $f\colon \mathbb{R} \to \mathbb{C}^{d\times d}$ is a function of positive type, setting $n = 2$, $t_1 = 0$, $t_2 = t$, and $x_1 = -x_2 = x$ in Definition 2.8 with $t \in \mathbb{R}$ and $x \in \mathbb{C}^{d\times d}$ arbitrary implies

$$x^*(f(0) - f(t) - f(-t) + f(0))x = x^*(f(0) - f(t) - f(t)^* + f(0)^*)x \geq 0,$$

i.e.

$$x^*(f(t) + f(t)^*)x \leq x^*(f(0) + f(0)^*)x$$

for every $t \in \mathbb{R}$ and $x \in \mathbb{C}^d$. Consequently $x^*(f(t) + f(t)^*)x$ is maximal at $t = 0$. If the right-hand derivative of f at 0, denoted by $f'(0+)$, exists, this implies that $f'(0+) + f'(0+)^*$ is symmetric and negative semi-definite.

Theorem 2.10 implies the following inverse of Lemma 2.9, whose proof we omit here:

Lemma 2.12. *If $f \in L^1(\mathbb{R}, \mathbb{R}^{d\times d}) \cap L^2(\mathbb{R}, \mathbb{R}^{d\times d})$ is a continuous function of positive type with $\mathcal{F}f \in L^1(\mathbb{R}, \mathbb{C}^{d\times d})$, there is a centered, stationary stochastic process $X\colon [0, \infty) \to \mathbb{R}^d$ with $C_X = f$.*

Theorem 2.10 motivates the following name:

Definition 2.13. *A function $f \in L^1(\mathbb{R}, \mathbb{R}^{d\times d})$ is called* of strictly positive type *if $\mathcal{F}f(\xi)$ is hermitian and positive definite for each $\xi \in \mathbb{R}$.*

If C_V^A has the form $C_V^A(t) = P^T e^{tA}\Sigma P$ for $t \geq 0$ with A from (2.20) stable, its Fourier transform is

$$\begin{aligned}
\mathcal{F}C_V^A(\xi) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\mathrm{i}t\xi} C_V^A(t)\,\mathrm{d}t \\
&= \frac{1}{\sqrt{2\pi}} \int_{0}^{\infty} e^{-\mathrm{i}t\xi} C_V^A(t)\,\mathrm{d}t + \int_{0}^{\infty} e^{\mathrm{i}t\xi} C_V^A(t)^T\,\mathrm{d}t \\
&= \frac{1}{\sqrt{2\pi}} P^T \left(\int_0^\infty e^{-\mathrm{i}t\xi\mathbf{I}+tA}\Sigma\,\mathrm{d}t + \int_0^\infty \Sigma e^{\mathrm{i}t\xi\mathbf{I}+tA^T}\,\mathrm{d}t \right) P \qquad (2.22) \\
&= \frac{1}{\sqrt{2\pi}} P^T \left(-(-\mathrm{i}\xi\mathbf{I}+A)^{-1}\Sigma - \Sigma(\mathrm{i}\xi\mathbf{I}+A^T)^{-1} \right) P \\
&= \frac{1}{\sqrt{2\pi}} P^T \left((\mathrm{i}\xi\mathbf{I}-A)^{-1}\Sigma + \Sigma(\mathrm{i}\xi\mathbf{I}-A)^{-*} \right) P
\end{aligned}$$

where the inverse matrices of $-\mathrm{i}\xi\mathbf{I} + A$ and $\mathrm{i}\xi\mathbf{I} + A^T$ exist since A is stable. Using $\Sigma P = \frac{1}{\beta m} P$ and the Schur complement identity (1.1), we obtain

$$P^T(\mathrm{i}\xi\mathbf{I}_{n+d} - A)^{-1}\Sigma P = \frac{1}{\beta m}\big(\mathrm{i}\xi\mathbf{I}_d - D + B^T(\mathrm{i}\xi\mathbf{I}_n - A_0)^{-1}C\big)^{-1}$$

if $\mathrm{i}\xi\mathbf{I}_n - A_0$ is non-singular (in particular if A_0 is stable, which is necessarily the case if C_V^A is of positive type, as we will see in Theorem 2.17) and therefore

$$\begin{aligned}
\mathcal{F}C_V^A(\xi) = \frac{1}{\beta m\sqrt{2\pi}} \Big(&\big(\mathrm{i}\xi\mathbf{I}_d - D + B^T(\mathrm{i}\xi\mathbf{I}_n - A_0)^{-1}C\big)^{-1} + \\
&\big(\mathrm{i}\xi\mathbf{I}_d - D + B^T(\mathrm{i}\xi\mathbf{I}_n - A_0)^{-1}C\big)^{-*} \Big),
\end{aligned}$$

cf. [10], provided that the inverse matrices exist. Consequently if A is stable, C_V^A is of positive type if and only if

$$\big(\mathrm{i}\xi\mathbf{I}_d - D + B^T(\mathrm{i}\xi\mathbf{I}_n - A_0)^{-1}C\big)^{-1} + \big(\mathrm{i}\xi\mathbf{I}_d - D + B^T(\mathrm{i}\xi\mathbf{I}_n - A_0)^{-1}C\big)^{-*} \qquad (2.23)$$

is positive semi-definite for all $\xi \in \mathbb{R}$. For a non-singular matrix H, $H + H^*$ is positive semi-definite if and only if $H^{-1} + H^{-*}$ is positive semi-definite since setting $y := Hx$ in

$$x^*(H + H^*)x \geq 0 \qquad \text{for every } x \in \mathbb{C}^d$$

yields

$$y^*(H^{-1} + H^{-*})y \geq 0 \qquad \text{for every } y \in \mathbb{C}^d.$$

Consequently (2.23) is positive semi-definite for all $\xi \in \mathbb{R}$ if and only if

$$\kappa(\mathrm{i}\xi) + \kappa(\mathrm{i}\xi)^* \quad \text{is positive semi-definite for all } \xi \in \mathbb{R} \qquad (2.24)$$

with

$$\kappa(z) := -D + B^T(z\mathbf{I}_n - A_0)^{-1}C. \tag{2.25}$$

In model reduction theory, κ is called the *transfer function*, cf. [3, 6]. The *Positive Real Lemma*, which we will formulate now, states a necessary and sufficient condition for when a solution of (2.21) exists.

Definition 2.14. *Let* $\mathbb{H} := \{z \in \mathbb{C} : \operatorname{Re} z > 0\}$ *with closure* $\overline{\mathbb{H}} = \{z \in \mathbb{C} : \operatorname{Re} z \geq 0\}$ *and let* $\mathcal{D} \subseteq \mathbb{C}$ *be a domain with* $\overline{\mathbb{H}} \subseteq \mathcal{D}$.

(i) A holomorphic function $f\colon \mathcal{D} \to \mathbb{C}^{d\times d}$ *is called* positive real *if*

- $\overline{f(z)} = f(\overline{z})$ *for each* $z \in \mathbb{H}$,
- $f(z) + f(z)^*$ *is positive semi-definite for each* $z \in \mathbb{H}$.

(ii) A positive real function $f\colon \mathcal{D} \to \mathbb{C}^{d\times d}$ *is called* strictly positive real *if* $f(\mathrm{i}\xi) + f(\mathrm{i}\xi)^*$ *is positive definite for each* $\xi \in \mathbb{R}$.

In Definition 2.14 we restrict ourselves to functions without singularities on the imaginary axis. By suitable adaptations, this definition can be extended to functions with singularities on the imaginary axis.

Remark 2.15. Let be a domain with $\overline{\mathbb{H}} \subseteq \mathcal{D}$. A holomorphic function $f\colon \mathcal{D} \to \mathbb{C}^{d\times d}$ with $\overline{f(z)} = f(\overline{z})$ for each $z \in \mathbb{H}$ is positive real if $f(\mathrm{i}\xi) + f(\mathrm{i}\xi)^*$ is positive semi-definite for each $\xi \in \mathbb{R}$ and $f(z) \xrightarrow{z\to\infty} M$ (with $z \in \mathcal{D}$) for a matrix M: In this case $M + M^T$ is positive semi-definite, consequently each $x \in \mathbb{C}^d$ satisfies $x^*\big(f(\mathrm{i}\xi) + f(\mathrm{i}\xi)^*\big)x \geq 0$ for each $\xi \in \mathbb{R}$ and $x^*\big(f(z) + f(z)^*\big)x \geq -\varepsilon$ and for each $\varepsilon > 0$ and $|z| \geq R$ with R sufficiently large (depending on ε). Since $x^*\big(f(z) + f(z)^*\big)x$ is harmonic (as a linear combination of the real and imaginary parts of the components of f), the maximum principle implies $x^*\big(f(z) + f(z)^*\big)x \geq -\varepsilon$ for each $x \in \mathbb{C}$ and $z \in \mathbb{H}$ with $|z| \leq R$. By letting $\varepsilon \to \infty$ and consequently $R \to 0$, the statement follows.

Moreover, if $x^*\big(f(z) + f(z)^*\big)x = 0$ for a $z \in \mathbb{H}$, then $x^*\big(f(z) + f(z)^*\big)x = 0$ for each $z \in \mathbb{H}$. This implies that if f is strictly positive real, $f(z) + f(z)^*$ is positive definite for each $z \in \mathbb{H}$.

Lemma 2.16 (Positive Real Lemma)**.** *Let* κ *have the form* (2.25) *where* $A_0 \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times d}$, $C \in \mathbb{R}^{n\times d}$, *and* $D \in \mathbb{R}^{d\times d}$ *are chosen such that* n *is minimal. Then* (2.21) *has a solution* $S \in \mathbb{R}^{n\times n}$, $L \in \mathbb{R}^{d\times d}$, $G \in \mathbb{R}^{n\times d}$ *with* S *symmetric and positive definite if and only if* κ *from* (2.25) *is positive real.*

Proofs of Lemma 2.16 can be found in [3, Theorem 3], [4, Chapter 5], and [6, Theorem 5.31]. From [3] we have also borrowed Definition 2.14(i). We will present an algorithm for computing the solution of (2.21) in Appendix A.

The following theorem and its proof are partially based on [15, Theorem 13].

Theorem 2.17. *Let A from (2.20) be stable where A_0, B, C, and D are chosen such that n is minimal, and let $C_V^A(t) = \frac{1}{\beta m} P^T e^{tA} P$ $(t \geq 0)$ be of strictly positive type. Then A_0 is stable, too, and the Lyapunov equation (1.11) has a unique solution (Σ, K) where Σ is symmetric and positive definite and has the form (2.16).*

Proof. For $\lambda \notin \sigma(A)$ we define

$$\xi(\lambda) := P^T(\lambda \mathbf{I} - A)^{-1} P.$$

If $\lambda \notin \sigma(A_0)$, (1.1) implies

$$\xi(\lambda) = \left(\lambda \mathbf{I}_d - D + B^T(\lambda \mathbf{I}_n - A_0)^{-1} C\right)^{-1} = \left(\lambda \mathbf{I}_d + \kappa(\lambda)\right)^{-1}$$

with κ from (2.25). $\xi(\lambda)$ and $x_0(\lambda) := -(\lambda \mathbf{I} - A_0)^{-1} C \xi(\lambda)$ solve the linear system

$$(\lambda \mathbf{I} - A) \begin{bmatrix} \xi(\lambda) \\ x_0(\lambda) \end{bmatrix} = \begin{bmatrix} \lambda \mathbf{I}_d - D & -B^T \\ C & \lambda \mathbf{I}_n - A_0 \end{bmatrix} \begin{bmatrix} \xi(\lambda) \\ x_0(\lambda) \end{bmatrix} = \begin{bmatrix} \mathbf{I}_d \\ \mathbf{0} \end{bmatrix} = P. \tag{2.26}$$

Since C_V^A is a function of strictly positive type, (2.22) shows that

$$P^T(\lambda \mathbf{I} - A)^{-1} P + P^T(\lambda \mathbf{I} - A)^{-*} P$$

is positive definite for all pure imaginary λ. This implies

$$\operatorname{Re}\left(x^* P^T(\lambda \mathbf{I} - A)^{-1} P x\right) > 0 \quad \text{for all} \quad x \in \mathbb{R}^d$$

as otherwise

$$x^* P^T(\lambda \mathbf{I} - A)^{-1} P x + x^* P^T(\lambda \mathbf{I} - A)^{-*} P x \leq 0.$$

Since A is stable, $\lambda \mathbf{I} - A$ is non-singular for all λ in the closed right half plane $\overline{\mathbb{H}}$, i.e. $\operatorname{Re}(x^* \xi(\mathrm{i}t) x)$ has no singularities in $\overline{\mathbb{H}}$. Consequently $\operatorname{Re}(x^* \xi(\mathrm{i}t) x)$ is a harmonic function in $\mathbb{H}$ for every $x \in \mathbb{R}^d$ with $\operatorname{Re}(x^* \xi(\mathrm{i}t) x) > 0$ for every $t \in \mathbb{R}$. $\xi(z) \xrightarrow{z \to \infty} 0$ (with $z \in \mathbb{C}$) implies that, according to the maximum principle, $\operatorname{Re}(x^* \xi(t) x)$ has no zeros in $\overline{\mathbb{H}}$ (for either x).

We now prove that A_0, too, is stable. Assume that A_0 has an eigenvalue μ with $\operatorname{Re} \mu \geq 0$. If $\kappa(\lambda)$ and thus $\lambda \mathbf{I} + \kappa(\lambda)$ is unbounded for $\lambda \to \mu$, then $\xi(\lambda)$ converges to a singular matrix, thus μ is a zero of $x^* \xi(\cdot) x$ in $\overline{\mathbb{H}}$ for a certain x, which contradicts our conclusion from above. If $\kappa(\lambda) = -D + B^T(\lambda \mathbf{I} - A_0)^{-1} C$ is bounded for $\lambda \to \mu$, then each column of B has to be orthogonal to every eigenvector of A_0 corresponding to the eigenvalue μ whose coefficient in the decomposition of any column of C into a linear combination of eigenvectors of A_0 is non-zero. If x_r is such an eigenvector, one directly verifies via (2.26) that $[\mathbf{0}_d^T, x_r^T]^T$ is an eigenvector of A corresponding to the eigenvalue μ, which contradicts the stability of A. Therefore no such eigenvector exists, i.e. the decomposition of any column of C into a linear combination of eigenvectors of A_0 contains no eigenvectors corresponding to the eigenvalue μ. Consequently every column of C is contained in the image $\operatorname{im}(A_0 - \mu \mathbf{I})$ of $A_0 - \mu \mathbf{I}$. Since $\operatorname{im}(A_0 - \mu \mathbf{I})$ is the orthogonal complement of the kernel $\ker((A_0 - \mu \mathbf{I})^*) = \ker(A_0^* - \overline{\mu} \mathbf{I})$, every column of C is orthogonal to every left

eigenvector of A_0 corresponding to the eigenvalue μ. (Here we denote as *left eigenvector* of A a vector x with $A^*x = \overline{\mu}x$.) However, if x_l is such a left eigenvector, (2.26) directly implies that $[\mathbf{0}_d^T, x_l^T]^T$ is a left eigenvector of A corresponding to the eigenvalue μ, which again contradicts the stability of A. Therefore A_0 must be stable.

Consequently κ has no singularities in $\mathbb{H}$ and is therefore holomorphic in $\mathbb{H}$. According to Remark 2.11, $(C_V^A)'(0+) + (C_V^A)'(0+)^T = \frac{1}{\beta m}(D + D^T)$ is negative semi-definite. Since A_0 is stable, κ has no singularities on the imaginary axis and thus (2.24) holds. Hence κ is positive real according to Remark 2.15 and Lemma 2.16 states that (2.21) and thus (1.11) has a unique solution. □

In other words, the Lyapunov equation (1.11) has a solution if C_V^A is a function of strictly positive type, i.e. if A belongs to a valid velocity auto-covariance function. On the other hand, if the Lyapunov equation has a solution (Σ, K), then (A, K) is an Ornstein-Uhlenbeck system with auto-covariance function C_V^A, i.e. C_V^A is of positive type (although not necessarily of strictly positive type) according to Lemma 2.9. We will present an algorithm for obtaining the solution in Appendix A. Since that algorithm fails if no solution exists, it is not necessary to explicitly test if C_V^A is a function of positive type or if κ is positive real. Note that $D + D^T$ has to be positive semi-definite for (2.21) to be solvable as one immediately sees from the third equation.

As we will see in our numerical experiments, omitting the requirement $D = \mathbf{0}$ and allowing the more general equation (2.5) is sometimes necessary to obtain a valid Ornstein-Uhlenbeck system at all. Whether such a representation makes sense physically depends on the physical background of the considered application, but there are applications where $D = \mathbf{0}$ is not necessary.

3. The Lanczos Method for the Generalized Langevin Equation

In this chapter we present our algorithm for the one-dimensional case. We will call it the *Lanczos method* although this name is usually given to Lanczos' algorithm for computing eigenvalues since both algorithms are based on the same underlying ideas. Our goal is the construction of an Ornstein-Uhlenbeck system (1.7) whose velocity auto-covariance function approximates the given velocity auto-covariance function of a generalized Langevin equation. We achieve this by interpolating the given velocity auto-covariance function C_V in an equidistant grid $0, \tau, 2\tau, \ldots, (n_0-1)\tau$ with $\tau > 0$ fixed, i.e. $(r_k)_{k=0,\ldots,n_0-1}$, $r_k := C_V(k\tau) \in \mathbb{R}^{d\times d}$ is the required input of the algorithm.

In this chapter, as well as in Chapter 4, we will restrict ourselves to the case $d = 1$. We assume that $r_0 = \frac{1}{\beta m} = 1$ holds; otherwise we first scale the data accordingly by replacing each r_k by $\frac{r_k}{r_0}$.

3.1. The Idea of the Lanczos Method

We use the following variant of the Lanczos method, which we borrow from [2]: For a given finite sequence $(r_k)_{k=0,\ldots,n_0-1}$ we define the linear *moments functional*

$$\Phi(\phi) := \sum_{j=0}^{n_0-1} a_j r_j$$

for polynomials $\phi(x) = \sum_{j=0}^{\deg\phi} a_j x^j$ with degree $\deg\phi \le n_0 - 1$ and the symmetric bilinear form

$$[\psi, \phi]_r := \Phi(\psi\phi)$$

for polynomials ψ, ϕ with $\deg\psi + \deg\phi \le n_0 - 1$. Here we identify polynomials mainly with their coefficient vectors rather than regarding them as functions since we will neither use any of their analytical properties nor evaluate them at any point. Given $(r_k)_{k=0,\ldots,n_0-1}$, the Lanczos method yields a (finite) sequence $(\phi_k)_{k=0,\ldots,N-1}$ of formally orthogonal polynomials w.r.t. $[\cdot,\cdot]_r$ with increasing degrees $\deg\phi_k = k$, i.e. $[\phi_j, \phi_k]_r = 0$ for $j \ne k$. The algorithm works similarly to the Gram-Schmidt algorithm. Note that $[\cdot,\cdot]_r$ is in general not positive definite and consequently $[\phi_k, \phi_k]_r = 0$ is possible, in which case no such sequence of orthogonal polynomials exists. We will ignore this case here and will always assume that $[\phi_k, \phi_k]_r \ne 0$. (In Section 5.2.1 we will give some basic information about this case without going into details.)

Let $\phi = (\phi_k)_{k=0,\dots,N-1}$ be formally orthogonal polynomials with $\deg \phi_k = k$. Then for $k > 0$, ϕ_k can be written as

$$\phi_k(x) = \frac{1}{t_{k-1,k}} \left(x \cdot \phi_{k-1}(x) - \sum_{j=0}^{k-1} t_{k-1,j} \phi_j(x) \right) \tag{3.1}$$

with $t_{k-1,j} = \frac{[x \cdot \phi_{k-1}, \phi_j]_r}{[\phi_j, \phi_j]_r} \in \mathbb{R}$ for $j < k$ and an arbitrary $t_{k-1,k} \neq 0$. (This is analogous to a step of the Gram-Schmidt algorithm, where the base polynomial $x \cdot \phi_{k-1}(x)$ is orthogonalized against the polynomials $\phi_0, \dots, \phi_{k-1}$.) It is known (cf. [18, p. 214]) that the recursion (3.1) actually is a "short" (three-step) recursion: If $j < i - 1$, then

$$t_{i,j} = \frac{[x \cdot \phi_i, \phi_j]_r}{[\phi_j, \phi_j]_r} = \frac{[\phi_i, x \cdot \phi_j]_r}{[\phi_j, \phi_j]_r} = 0$$

(since ϕ_i is orthogonal to all polynomials of lower degree) and thus (3.1) is equivalent to

$$\phi_k(x) = \frac{1}{t_{k-1,k}} (x \cdot \phi_{k-1}(x) - t_{k-1,k-1}\phi_{k-1}(x) - t_{k-1,k-2}\phi_{k-2}(x)) \tag{3.2}$$

(where $t_{0,-1} := 0$). We set $t_{k-1,j} := 0$ for $j > k$ and combine the recursion coefficients $t_{i,j}$ with $i \geq 0$ in the matrix

$$J := [t_{i,j}]_{i,j=0,\dots,N-1} \in \mathbb{R}^{N \times N},$$

which is a tridiagonal matrix due to (3.2). Here N is chosen as the largest possible value such that all ϕ_k with $k = 0, \dots, N-1$ can be computed from the moments $(r_k)_{k=0,\dots,n_0-1}$; this results in $N = \lceil \frac{n_0}{2} \rceil$. If n_0 is even, the moments required to compute $\phi_{\lceil n_0/2 \rceil}$ via (3.1) are known, too, and one could thus think that $N = \lceil \frac{n_0}{2} \rceil + 1$ is possible, but this choice would result in $[\phi_{N-1}, x^k]_r = 0$ for all $k < N - 1$ or in $[\phi_{N-1}, \phi_{N-1}]_r = 0$. We will mention both of these special cases in Section 5.2.1 but ignore them here. Since neither of them would allow to actually execute this extra step, $N = \lceil \frac{n_0}{2} \rceil$ holds in the case of an even n_0, too.

Starting with the polynomial $\phi_0(x) := \frac{1}{\sqrt{r_0}} = 1$, the Lanczos method computes the sequence ϕ and the matrix J via (3.1). The usefulness of the algorithm results from the following property:

Let a_k be the (infinitely long) coefficient vector of ϕ_k. Then (3.1) can be rewritten as the matrix-vector product

$$\begin{bmatrix} 0 & 0 & \dots & 0 \\ a_0 & a_1 & \dots & a_{N-1} \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & \dots & a_{N-1} \end{bmatrix} J^T + \begin{bmatrix} \mathbf{0} & \dots & \mathbf{0} & \xi \end{bmatrix}$$

where ξ in the final column represents the coefficients of $t_{N-1,N}\phi_N$, which have not yet been computed. By induction this yields

$$\begin{bmatrix} \mathbf{0}_k & \mathbf{0}_k & \dots & \mathbf{0}_k \\ a_0 & a_1 & \dots & a_{N-1} \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & \dots & a_{N-1} \end{bmatrix} (J^T)^k + \sum_{j=0}^{k-1} \begin{bmatrix} \mathbf{0}_j & \dots & \mathbf{0}_j & \mathbf{0}_j \\ \mathbf{0} & \dots & \mathbf{0} & \xi \end{bmatrix} (J^T)^{k-j-1}.$$

Due to the tridiagonal form of J, only the last $k-j$ columns of

$$\begin{bmatrix} \mathbf{0}_j & \dots & \mathbf{0}_j & \mathbf{0}_j \\ \mathbf{0} & \dots & \mathbf{0} & \xi \end{bmatrix} (J^T)^{k-j-1}$$

are non-zero, i.e. for $k \leq N-1$ the first column of the sum is zero. Consequently for $k \leq N-1$ in polynomial notation the first column of this matrix equation reads

$$x^k \phi_0(x) = \sum_{j=0}^{k} (J^k)_{0,j} \phi_j(x).$$

With $\delta_0 := [\phi_0, \phi_0]_r$ this implies

$$r_k = [x^k, 1]_r = \frac{1}{\delta_0}[x^k \phi_0, \phi_0]_r = \frac{1}{\delta_0}\left[\sum_{j=0}^{k} (J^k)_{0,j}\phi_j, \phi_0\right]_r = \frac{1}{\delta_0}\sum_{j=0}^{k}(J^k)_{0,j}[\phi_j, \phi_0]_r. \tag{3.3}$$

Due to the orthogonality of the polynomials, all summands except for the one with $j=0$ vanish and we obtain

$$\frac{1}{\delta_0}\sum_{j=0}^{k}(J^k)_{0,j}[\phi_j, \phi_0]_r = \frac{1}{\delta_0}(J^k)_{0,0}[\phi_0, \phi_0]_r = (J^k)_{0,0} = e_1^T J^k e_1. \tag{3.4}$$

Consequently $A := \frac{1}{\tau}\log J$ satisfies

$$e_1^T e^{k\tau A} e_1 = r_k \quad \text{for} \quad k = 0, \dots, n_0 - 1.$$

Therefore if $X \in \mathbb{R}^N$ is a stationary Ornstein-Uhlenbeck process (1.7) with drift matrix A and a covariance matrix of the form $\Sigma = \frac{1}{\beta m}\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & S \end{bmatrix}$ from (2.16), the first component's auto-covariance function interpolates the values $r_0, \dots, r_{n_0-1}$ in the time grid $0, \tau, \dots, (n_0-1)\tau$. In our application, this means that choosing $r_k := C_V(k\tau)$ and $X = \begin{bmatrix} V \\ Z \end{bmatrix}$ yields an Ornstein-Uhlenbeck process X such that the auto-covariance function of V interpolates the given VACF in $0, \tau, \dots, (n_0-1)\tau$.

When applying the algorithm to construct the polynomials, $t_{k-1,k}$ from (3.1) are scaling factors which can be chosen arbitrarily.

3.2. Further Steps in the Algorithm

3.2.1. Conditions on the Velocity Auto-Covariance Function

As mentioned in Lemma 2.9, C_V is a function of *positive type*. Consequently, according to Remark 2.11, C_V is maximal at 0, which implies $C_V'(0+) \leq 0$ where $C_V'(0+)$ denotes

the right-hand derivative. Depending on the underlying physical system it may also be necessary that C_V satisfies

$$C_V'(0) = 0. \tag{3.5}$$

In an Ornstein-Uhlenbeck system, (3.5) is equivalent to

$$0 = (C_V^A)'(0) = \frac{1}{\beta m} e_1^T A \cdot e^{0 \cdot A} e_1 = \frac{1}{\beta m} D$$

with D from (2.20). $D = 0$ means that the corresponding GLE has no instantaneous friction and is actually of the type (2.3). We ensure this by adjusting the second data point $r_1 = C_V(\tau)$ since modifying this value should have a significant and somehow predictable impact on the values of C_V^A in $(0, \tau)$ and thus on $(C_V^A)'(0)$.

To adjust r_1 correctly, we use a Newton iteration including algorithmic differentiation of the Lanczos method: For each variable which is computed we also compute its derivative w.r.t. r_1. (Every single calculation in Section 3.1 can be differentiated in a straightforward way.) We thus obtain the derivative of J, which we denote as ∂J. To compute the derivative of A w.r.t. to r_1, denoted as ∂A, we employ a formula which we will present later in Lemma 3.1. In accordance with Newton's method we then iteratively adjust r_1 and reapply the Lanczos algorithm from Section 3.1, obtaining new matrices A and ∂A until $D = A_{1,1} \approx 0$.

Note that the physical importance of condition (3.5) depends on the actual physical system considered. In some use cases there is no physical need for C_V to be differentiable in 0. In these cases a failure of the Newton iteration is acceptable or the iteration may be skipped completely as long as $(C_V^A)'(0+) \leq 0$. However, our numerical experiments in Section 3.4 show that applying the Newton iteration may still make sense in order to obtain a function C_V^A of positive type.

3.2.2. Spurious Eigenvalues

Eigenvalues of J outside the interior unit disk mean that the corresponding eigenvalues of A have non-negative real part, hence A is no longer stable and $e_1^T e^{tA} e_1 \xrightarrow{t \to \infty} 0$ no longer holds, i.e. the corresponding Prony series contains exponentials with non-negative real part. This does not make sense physically since it means that the system is not dissipative. We call these eigenvalues "spurious" eigenvalues. However, assuming that our data are based on a physical model, the "contribution" of these spurious eigenvalues to the auto-covariance function, i.e. their coefficients in the Prony series, should be very small and therefore "removing" them from A (i.e. replacing A by a smaller matrix containing only the non-spurious eigenvalues) should hardly influence $C_V^A(t)$ for $t \leq (n_0 - 1)\tau$.

In the following we describe this procedure in detail. We require J to be diagonalizable with pairwise different eigenvalues μ_j, $|\mu_j| \leq |\mu_{j+1}|$ for all j. Then A has the eigenvalues $\lambda_j := \frac{1}{\tau} \operatorname{Log} \mu_j$ with the same eigenvectors, i.e.

$$J = U \Lambda_J U^{-1}, \qquad A = U \Lambda_A U^{-1}$$

where $\Lambda_J = \operatorname{diag}(\mu_1, \ldots, \mu_N)$, $\Lambda_A = \operatorname{diag}(\lambda_1, \ldots, \lambda_N)$, $U \in \mathbb{C}^{N\times N}$ is non-singular, and

$$C_V^A(t) = e_1^T U e^{t\Lambda_A} U^{-1} e_1 = \sum_{j=1}^{N} U_{1,j}(U^{-1})_{j,1} e^{t\lambda_j} = \sum_{j=1}^{N} w_j e^{t\lambda_j} \qquad (t \geq 0)$$

with $w_j := U_{1,j}(U^{-1})_{j,1}$. Let $\operatorname{Re}\lambda_{j_0} < 0 \leq \operatorname{Re}\lambda_{j_0+1}$ for some j_0. (The case $\operatorname{Re}\lambda_N < 0$ is trivial as nothing needs to be done; the case $\operatorname{Re}\lambda_1 \geq 0$ does not make sense as this would mean that all eigenvalues are spurious, so we ignore these cases.) Since the original auto-covariance function interpolated by C_V^A belongs to a physical system, we assume that the coefficients w_j with $j > j_0$ are small compared to $\max_{1\leq j\leq j_0} |w_j|$ and thus

$$\left| C_V^A(t) - \sum_{j=1}^{j_0} w_j e^{t\lambda_j} \right| = \left| \sum_{j=j_0+1}^{N} w_j e^{t\lambda_j} \right|,$$

too, is small for $t \in [0, (n_0 - 1)\tau]$. Hence we replace A by a $j_0 \times j_0$ matrix $\widetilde{A} = \widetilde{U}\Lambda_A^- \widetilde{U}^{-1}$ with

$$C_V^{\widetilde{A}}(t) = \sum_{j=1}^{j_0} \widetilde{w}_j e^{t\lambda_j} \approx \sum_{j=1}^{j_0} w_j e^{t\lambda_j}, \qquad \widetilde{w}_j := \widetilde{U}_{1,j}(\widetilde{U}^{-1})_{j,1}$$

where $\Lambda_A^- = \operatorname{diag}(\lambda_1, \ldots, \lambda_{j_0})$ and $\widetilde{U} \in \mathbb{C}^{j_0\times j_0}$ contains only the first j_0 columns of U (i.e. the columns belonging to eigenvalues of A with negative real part) and only j_0 rows from U. The first row of $\widetilde{U}$ must equal the first row of U except for the removed entries in order to obtain coefficients $\widetilde{w}_j$ in the Prony series representation of $C_V^{\widetilde{A}}$ with

$$\widetilde{w}_j = \widetilde{U}_{1,j}(\widetilde{U}^{-1})_{j,1} = U_{1,j}(\widetilde{U}^{-1})_{j,1} \approx U_{1,j}(U^{-1})_{j,1} = w_j.$$

Here the assumption $(\widetilde{U}^{-1})_{j,1} \approx (U^{-1})_{j,1}$ is motivated as follows: Since $U_{1,j}(U^{-1})_{j,1} = w_j \ll 1$ for $j > j_0$, we assume that $(U^{-1})_{j,1} \ll 1$ for $j > j_0$. Using scalars, the matrix multiplications $\widetilde{U}\widetilde{U}^{-1} = \mathbf{I}$ and $UU^{-1} = \mathbf{I}$ for the ith entry of the first column can be rewritten as

$$\sum_{j=1}^{j_0} \widetilde{U}_{i,j}(\widetilde{U}^{-1})_{j,1} = \delta_{i,1} = \sum_{j=1}^{N} U_{i,j}(U^{-1})_{j,1} = \sum_{j=1}^{j_0} \widetilde{U}_{i,j}(U^{-1})_{j,1} + \sum_{j=j_0+1}^{N} U_{i,j}(U^{-1})_{j,1},$$

where the second sum is small. Consequently $\sum_{j=1}^{j_0} \widetilde{U}_{i,j}(\widetilde{U}^{-1})_{j,1} \approx \sum_{j=1}^{j_0} \widetilde{U}_{i,j}(U^{-1})_{j,1}$ for each i and therefore $(\widetilde{U}^{-1})_{j,1} \approx (U^{-1})_{j,1}$ for each j.

Apart from the first row, $\widetilde{U}$ can omit arbitrary rows from U as long as it is non-singular. In order to ensure that $\widetilde{U}$ is non-singular, we build $\widetilde{U}$ row by row: We first take the first row of U as first row of $\widetilde{U}$ (except for the entries of columns which were removed). After that, we successively append the row of U to $\widetilde{U}$ whose difference to its orthogonal projection onto the linear hull of the present rows of $\widetilde{U}$ has the largest norm (after scaling all rows to have norm 1). I.e., as ith row $\widetilde{U}_{i,\cdot}$ of $\widetilde{U}$ we take the row x of U which

minimizes

$$\frac{1}{\|x\|}\left(x - \sum_{j=1}^{i-1} \frac{\langle x, \widetilde{U}_{j,\cdot}\rangle}{\|\widetilde{U}_{j,\cdot}\|^2}\widetilde{U}_{j,\cdot}\right).$$

We continue appending rows to $\widetilde{U}$ in this way until $\widetilde{U}$ is a square matrix. If the assumption that $(U^{-1})_{j,1}$ is small for $j > j_0$ holds, then

$$C_V^{\widetilde{A}}(t) = \sum_{j=1}^{j_0} \widetilde{U}_{1,j}(\widetilde{U}^{-1})_{j,1} e^{t\lambda_j} \approx \sum_{j=1}^{j_0} w_j e^{t\lambda_j} = C_V^A(t).$$

We continue the algorithm with $\widetilde{J}$, $\widetilde{A}$ and $\widetilde{U}$ but omit the tilde in the following sections; we also write N instead of j_0 for the size of these new matrices.

Sometimes it is necessary to remove some additional eigenvalues with negative real part: According to (2.22) the poles of the Fourier transform of C_V^A are exactly the values $\pm\mathrm{i}\lambda$ for all $\lambda \in \sigma(A)$. While a pure imaginary eigenvalue $\lambda \neq 0$ thus means that $\mathcal{F}C_V^A$ has two poles on the real axis, an eigenvalue with real part close to 0 can manifest itself in (positive or negative) "peaks" in $\mathcal{F}C_V^A(\xi)$ for $\xi \approx \pm\operatorname{Im}\lambda$. A negative peak can result in $\mathcal{F}C_V^A(\xi)$ being negative for $\xi \approx \pm\operatorname{Im}\lambda$, which implies that C_V is no longer of positive type. If the corresponding Prony series coefficient is small, removing this eigenvalue can yield a new matrix A such that C_V^A is of positive type while hardly having any impact on the approximation. This is done similarly to removing spurious eigenvalues above, except that $\lambda_1, \ldots, \lambda_N$ are arranged in such a way that exactly all λ_j with $j > j_0$ need to be removed for a certain j_0.

While this procedure usually hardly modifies $C_V^A(t)$ for $t \in [0, (n_0 - 1)\tau]$, we cannot guarantee this. If the removal of eigenvalues results in significant changes in $C_V^A(t)$ for $t < (n_0 - 1)\tau$, one should not continue the algorithm but rather restart it with a different interpolation grid.

3.2.3. Negative Real Eigenvalues

On top of spurious eigenvalues, J can have negative real eigenvalues. Let μ_j be such an eigenvalue. Simply setting $A = \frac{1}{\tau}\log J$ would result in a matrix A with a complex eigenvalue $\lambda_j = \frac{1}{\tau}\operatorname{Log}\mu_j = \frac{1}{\tau}(\log|\mu_j| + \pi\mathrm{i})$ whose complex conjugate is not an eigenvalue of A, and thus in A being non-real (regardless of which matrix logarithm is used). We therefore "duplicate" the eigenvalue μ_j of J in such a way that both $\frac{1}{\tau}(\log|\mu_j| + \pi\mathrm{i})$ and $\frac{1}{\tau}(\log|\mu_j| - \pi\mathrm{i})$ are eigenvalues of A. We do this by inserting an additional row and column into Λ_A and U (where $A = U\Lambda_A U^{-1}$ denotes again the eigenvalue decomposition of A) for each such eigenvalue in J, obtaining two new matrices $\widetilde{\Lambda}_A$ and $\widetilde{U}$.

Define $\Lambda_J = \operatorname{diag}(\mu_1, \ldots, \mu_N)$, $\Lambda_A = \operatorname{diag}(\lambda_1, \ldots, \lambda_N)$, and U as in Section 3.2.2 (assuming that spurious eigenvalues have already been removed). Let μ_j be a negative real eigenvalue of J. Then we set $\widetilde{\Lambda}_A = \operatorname{diag}(\lambda_1, \ldots, \lambda_{j-1}, \lambda_j^+, \lambda_j^-, \lambda_{j+1}, \ldots \lambda_N)$ with $\lambda_j^\pm := \frac{1}{\tau}(\log|\mu_j| \pm \pi\mathrm{i})$. $\widetilde{U}$ is obtained as follows: If $x = U_{\cdot,j}$ is the eigenvector corresponding to μ_j in the jth column of U, we replace this column in $\widetilde{U}$ by two consecutive columns

containing the new eigenvectors

$$\begin{bmatrix} x \\ \mathrm{i} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} x \\ -\mathrm{i} \end{bmatrix}$$

for λ_+ and λ_- (at column index j and $j+1$), respectively, where the entries i and $-$i belong to the new row, and extend the other columns of $\widetilde{U}$ by one zero entry in this row, i.e. we replace $U = \begin{bmatrix} U_{\cdot,1} & \dots & U_{\cdot,j} & \dots & U_{\cdot,N} \end{bmatrix} \in \mathbb{R}^{N \times N}$ by

$$\widetilde{U} = \begin{bmatrix} U_{\cdot,1} & \dots & U_{\cdot,j-1} & x & x & U_{\cdot,j+1} & \dots & U_{\cdot,N} \\ 0 & \dots & 0 & \mathrm{i} & -\mathrm{i} & 0 & \dots & 0 \end{bmatrix} \in \mathbb{R}^{(N+1)\times(N+1)}.$$

Recall that

$$C_V^A(t) = e_1^T U \Lambda_J U^{-1} e_1 = \sum_{j=1}^{N} w_j e^{t\lambda_j} \quad \text{with} \quad w_j = U_{1,j} U_{j,1}^{-1}$$

for $t \geq 0$. It is easy to verify that $\widetilde{U}$ is non-singular and that

$$\begin{aligned} e_1^T \widetilde{U} &= \begin{bmatrix} U_{1,1} & \dots & U_{1,j-1} & U_{1,j} & U_{1,j} & U_{1,j+1} & \dots & U_{1,N} \end{bmatrix}, \\ \widetilde{U}^{-1} e_1 &= \begin{bmatrix} U_{1,1}^{-1} & \dots & U_{j-1,1}^{-1} & \frac{1}{2} U_{j,1}^{-1} & \frac{1}{2} U_{j,1}^{-1} & U_{j+1,1}^{-1} & \dots & U_{N,1}^{-1} \end{bmatrix}^T, \end{aligned}$$

which implies

$$e_1^T \widetilde{U} e^{t\widetilde{\Lambda}_A} \widetilde{U}^{-1} e_1 = e_1^T \widetilde{U} e^{t\widetilde{\Lambda}_A} \widetilde{U}^{-1} e_1 = \sum_{j=1}^{N} w_j e^{t\lambda_j}.$$

Therefore duplicating any $\mu \in \sigma(J)$ in this way has no impact on the auto-covariance function of the Ornstein-Uhlenbeck system but allows us to compute a real matrix logarithm of J in the case where μ is negative real. We continue the algorithm using $\widetilde{A} = \widetilde{U} \widetilde{\Lambda}_A \widetilde{U}^{-1}$ instead of $A = U \Lambda_A U^{-1}$.

We apply these modifications to U and Λ_A for each negative real eigenvalue of J. We again call the resulting matrix A and omit the tilde in the following. This matrix A is real in exact arithmetic. Imaginary parts in entries of A are due to numerical errors and can be set to 0.

Note that the adjustments in Sections 3.2.2 and 3.2.3 modify the size of A. While the matrix J generated by the Lanczos method has size $\lceil \frac{n_0}{2} \rceil \times \lceil \frac{n_0}{2} \rceil$, the final matrix will usually be smaller due to removed spurious eigenvalues (as we will see in Section 3.4) although it can be larger due to duplicated negative real eigenvalues. Therefore we cannot specify the final size of A exactly but only a "desired" size.

3.2.4. The Lur'e Equations

In order to determine $K \in \mathbb{R}^{N \times d}$ from (1.7), we finally have to solve the Lyapunov equation (1.11) or the equivalent Lur'e equations. Through this we also obtain the

covariance matrix Σ of the stationary distribution of this Ornstein-Uhlenbeck system and verify that A generates a velocity auto-covariance function of positive type. If Newton's method succeeded and the removal of spurious eigenvalues did not modify the first matrix entry too much, we have $D \approx 0$, resulting in the singular equations (2.19). Otherwise we have to solve the regular equations (2.21). We present algorithms for solving both cases in Appendix A.

3.3. Summary of the Complete Algorithm

The algorithm starts with the polynomial $\phi_0(x) := \frac{1}{\sqrt{r_0}} = 1$. In each step, a new polynomial ϕ_k with $\deg \phi_k = k$ is computed via (3.2) where we choose $t_{k-1,k}$ in (3.1) in such a way that $|[\phi_k, \phi_k]_r| = |\Phi(\phi_k^2)| = 1$. Instead of computing the coefficients $t_{k-1,k-1}$ and $t_{k-1,k-2}$ via directly evaluating $[\cdot, \cdot]_r$, we employ the recursive formula presented in [18, Theorem 2] to compute them. We form the tridiagonal matrix J from the coefficients $t_{i,k}$, appending a new column and a new row in each step. The Lanczos algorithm stops when the polynomials' degrees get too large, i.e. when $[\phi_k, \phi_k]_r$ is no longer defined by r. For each variable we also compute its derivative w.r.t. r_1 along with it. Differentiating the Lanczos algorithm is straightforward as it only includes easily differentiable calculations.

If n_0 is odd, we obtain an $\frac{n_0+1}{2} \times \frac{n_0+1}{2}$ matrix J; however, we cannot compute its last (bottom right) entry with index $(\frac{n_0-1}{2}, \frac{n_0-1}{2})$. Due to the tridiagonal form of J, no value $e_1^T J^k e_1$ for $k < n_0$ depends on this entry. Consequently J satisfies $e_1^T J^k e_1 = r_k$ for $k < n_0$, regardless of its value. Nonetheless A and thus C_V depend on this entry. We simply set it to 0.

Next we compute $A = \frac{1}{\tau} \log J$. At this time, we do not yet have to care about negative real eigenvalues of J since we are only interested in the real part of A and can thus take any matrix logarithm. In order to apply the Newton iteration described in Section 3.2.1, we compute the derivative of the matrix logarithm via the following identity, which we borrow from [22, p. 58]:

Lemma 3.1. *Let $\mathcal{A} \subseteq \mathbb{R}$, $\mathcal{B} \subseteq \mathbb{C}$ be open sets and $t_0 \in \mathcal{A}$. Let $H\colon \mathcal{A} \to \mathbb{R}^{n\times n}$ be differentiable in t_0 with a spectrum that lies in $\mathcal{B}$ for all t in a neighborhood of t_0. Let further $f\colon \mathcal{B} \to \mathbb{C}$ be holomorphic. Then*

$$f\left(\begin{bmatrix} H(t_0) & H'(t_0) \\ \mathbf{0} & H(t_0) \end{bmatrix}\right) = \begin{bmatrix} f(H(t_0)) & \frac{\mathrm{d}}{\mathrm{d}t} f(H(t))|_{t=t_0} \\ \mathbf{0} & f(H(t_0)) \end{bmatrix}.$$

Here f also denotes the matrix function induced by the scalar function f.

Defining $H(t)$ as the matrix J from above, regarded as a function of $t = r_1$ (where all other entries in r are fixed to their given values), and taking $f(M) := \log(M)$ as a branch of the complex logarithm (and the corresponding matrix logarithm) such that f

is holomorphic in the spectrum of $H(t)$ for all t in a sufficiently small neighborhood of 0 allows us to differentiate the matrix logarithm, obtaining

$$A'_{1,1} := e_1^T \left(\frac{\mathrm{d}}{\mathrm{d}t} \frac{1}{\tau} \log H(t)|_{t=r_1} \right) e_1,$$

and to apply Newton's method. Depending on the chosen logarithm function, the eigenvalues of A may not be symmetric w.r.t. the real axis (at least if $H(t_0)$ has a negative real eigenvalue and the principal logarithm branch cannot be chosen), which leads to $A_{1,1}$ being non-real. However, since the imaginary part exists only due to this, we can ignore it and simply take the real part of $A_{1,1}$ and its derivative. Therefore we replace r_1 by $r_1 - \frac{\operatorname{Re} A_{1,1}}{\operatorname{Re} A'_{1,1}}$ in each Newton step and reapply the Lanczos algorithm until $A_{1,1} \approx 0$. Of course other methods instead of the Newton iteration are conceivable, especially when the latter does not converge. For simplicity we tried no alternative methods to the Newton iteration in our numerical experiments. When the iteration did not converge, we assumed the algorithm to have failed for this grid and reduced n_0 by 1 in order to obtain a new grid.

After applying the Newton iteration, we treat spurious eigenvalues (i.e. eigenvalues outside the interior unit disk) and negative real eigenvalues of J: For this we compute an eigenvalue decomposition $J = U\Lambda_J U^{-1}$ of J. We subsequently modify Λ_J as described in Section 3.2: First we remove all $\mu \in \sigma(J)$ with $|\mu| \geq 1$ where the corresponding eigenvalue $\lambda = \frac{1}{\tau} \operatorname{Log} \mu$ would have positive real part, replacing Λ_J by a smaller diagonal matrix which contains only the remaining diagonal entries of Λ_J. For each such eigenvalue we also remove the corresponding column of U (which contains the corresponding eigenvector) and one row of U.

We may also have to remove other eigenvalues with negative real part close to 0 and small coefficient in the Prony series if they prevent the resulting VACF from being of positive type. To determine these eigenvalues, we first compute the Fourier transform

$$\mathcal{F}C_V^A(\operatorname{Im} \lambda) = \frac{1}{\beta m \sqrt{2\pi}} P^T \left((\mathrm{i} \operatorname{Im} \lambda \mathbf{I} - A)^{-1} + (\mathrm{i} \operatorname{Im} \lambda \mathbf{I} - A)^{-*} \right) P$$

for all $\lambda \in \sigma(A)$. Then from all λ with $\mathcal{F}C_V^A(\operatorname{Im} \lambda) < 0$ we choose the one with largest (i.e. closest to 0) real part and remove it (and its complex conjugate if it is not real) from A, similarly to removing spurious eigenvalues. We iterate this procedure until $\mathcal{F}C_V^A(\operatorname{Im} \lambda)$ is non-negative for all remaining λ or until an eigenvalue with significant coefficient in the Prony series is encountered, in which case the algorithm is assumed to have failed. Before removing another eigenvalue (or pair of eigenvalues), the values $\mathcal{F}C_V^A(\operatorname{Im} \lambda)$ are re-calculated using the new matrix A as our numerical experiments demonstrate that it is often not necessary to remove all λ for which $\mathcal{F}C_V^A(\operatorname{Im} \lambda)$ is initially negative. (Variations of this procedure are conceivable, e.g. always removing the eigenvalue with the smallest coefficient in the Prony series instead of the largest real part.) On top of this, we also remove all eigenvalues with coefficient smaller than 10^{-6} in the Prony series.

After removing eigenvalues we construct Λ_A and thereby duplicate negative real eigenvalues of J as described in Section 3.2.3. After computing Λ_A and modifying U accordingly,

$A := U\Lambda_A U^{-1}$ is the desired matrix.

Ideally, spurious eigenvalues having a significant impact on $C_V^A(t)$ only for large t, their removal should hardly influence $A_{1,1}$, so (3.5) should still hold approximately after removing them. Still, it is possible that $A_{1,1} > 0$, which implies that the corresponding auto-covariance function is no longer of positive type. A second Newton iteration including all steps performed so far is impossible as it would require computing the derivative of the eigenvalue decomposition of A. Instead we simply set $A_{1,1}$ to 0. Visually, modifying $A_{1,1}$ results in "shifting" the complete function C_V^A up (if increasing $A_{1,1}$) or down (if decreasing $A_{1,1}$) as we will see in Section 3.4. If $A_{1,1} \approx 0$ held before, the impact on C_V^A is minor. Otherwise this can significantly change C_V^A and it might be better to retry the complete algorithm with another grid. (3.5) also implies that we have to solve the singular Lur'e equations. Since $C_V'(0) > 0$ implies that C_V is not of positive type, the resulting function C_V is on the "boundary" of the set of all functions of positive type and due to rounding errors it is possible that the corresponding system does no longer have a VACF of positive type. Thus we instead set $A_{1,1} = -\varepsilon$ with a small $\varepsilon > 0$. For our experiments in Section 3.4 we chose $\varepsilon = 10^{-10}$. As with Newton's method, this step can be skipped if $A_{1,1} \leq 0$ already holds and if (3.5) is not physically necessary. (Note that $A_{1,1} \leq 0$ is not sufficient for C_V^A being of positive type but only necessary.)

Additionally we can replace A by an equivalent tridiagonal matrix by again using the Lanczos algorithm. In theory one could apply the algorithm from Section 3.3 again, this time using $e_1^T A^k e_1$ $(k = 0, \ldots, N-1)$ as moments, thereby obtaining a tridiagonal matrix $\widetilde{A}$ such that $e_1^T \widetilde{A}^k e_1 = e_1^T A^k e_1$ for $k = 0, \ldots, N-1$ by virtue of (3.3) and (3.4). The obtained tridiagonal matrix $\widetilde{A}$ is equivalent to A (as we will explain in Section 6.1.2) and satisfies $C_V^A = C_V^{\widetilde{A}}$. In practice, however, due to the large condition of A^k this can lead to huge numerical errors rendering this method unusable. Hence in our experiments in Section 3.4 we used a different implementation described in [27, Algorithm 2.3.1], which employs the more typical approach via Lanczos vectors instead of formally orthogonal polynomials but yields the same results in exact arithmetic. Another implementation can be found in [16].

The advantage of a tridiagonal A lies in significantly reduced computational costs for simulating sample paths of the Ornstein-Uhlenbeck process, e.g. when using the Euler-Maruyama method: Multiplying A by a vector only takes time $O(N)$ instead of $O(N^2)$. In theory, the distribution of $V(t)$ of the Ornstein-Uhlenbeck process is not changed by this final step. It can, however, be another source of numerical problems due to the Lanczos algorithm's numerical instability.

Finally we solve the Lur'e equations – regular or singular – to verify that the obtained velocity auto-covariance function is a function of positive type and to calculate K. If no solution of (2.21) exists, C_V^A is not of positive type and it only remains to use another time grid, i.e. another τ or n_0, in order to retry the algorithm with another input. The nonexistence of a solution can manifest itself in several ways when applying the algorithm from Appendix A: First, A_0 can be unstable, which implies that C_V^A is not of positive type, see Theorem 2.17. Second, if $D \approx 0$, the algorithm for solving the Lur'e equations reduces the system of equations to a (regular) system of smaller dimension; however, the value D of the reduced system may be positive. Third, if $D \not\approx 0$, Theorem A.5

states conditions for when the Lur'e equations have a solution, which may or may not be fulfilled.

We see that it is in no way guaranteed that this algorithm will always succeed, even with reasonable input, and it may be necessary to try several interpolation grids. In Section 3.4, however, we will see that it is usually possible to find a reasonable grid where the algorithm generates a valid velocity auto-covariance function.

3.3.1. A Note on Numerical Stability

The Lanczos algorithm, just as the Gram-Schmidt algorithm, suffers from stability issues, cf. [27, Chapter 3]. However, they typically emerge only for matrices larger than those which appear here and they posed no problem in our numerical experiments. For matrices of the size that we consider it is also very unlikely that the assumption that J is diagonalizable does not to hold.

To detect possible stability problems, one could compute the Gram matrix of the polynomials $\phi_0, \ldots, \phi_{N-1}$ to test whether they are actually orthogonal. If they are not, a typical remedy proposed e.g. in [27, Section 3.3] is explicitly orthogonalizing the polynomials. In order to reduce the computational cost of a full application of the Gram-Schmidt algorithm, several "partial" orthogonalization methods are mentioned there. Just as stability issues in general, however, the computational cost of orthogonalization is an issue only with matrices which are much larger than ours.

3.4. Numerical Results

While we have already presented some of the following results in [10], the improved algorithm here also accepts odd numbers n_0 (whereas we formerly required n_0 to be even). On top of that, the removal of eigenvalues of A with negligible coefficient which cause C_V^A not to be of positive type results in fewer cases where the algorithm fails. These are reasons for the partially different results obtained here.

3.4.1. Molecular Dynamics Data

We use the molecular dynamics data from [25], from where we borrow the following description. They describe a single colloid in a Lennard-Jones potential. The system was created by placing Lennard-Jones particles with diameter σ on a face-centered cubic lattice with lattice constant 1.71σ. To calculate the interactions, the Lennard-Jones potential was cut off at 2.5σ. The colloid was carved out off the lattice with a radius 3σ and was then defined as rigid body (i.e. the interparticle distances were fixed), which resulted in a colloid mass of $80m_0$, where m_0 is the mass of a Langevin particle. The cubic simulation box has a size of $L = 41.04\sigma$ with periodic boundary conditions in all three dimensions. To sample at the correct temperature, the system was equilibrated using

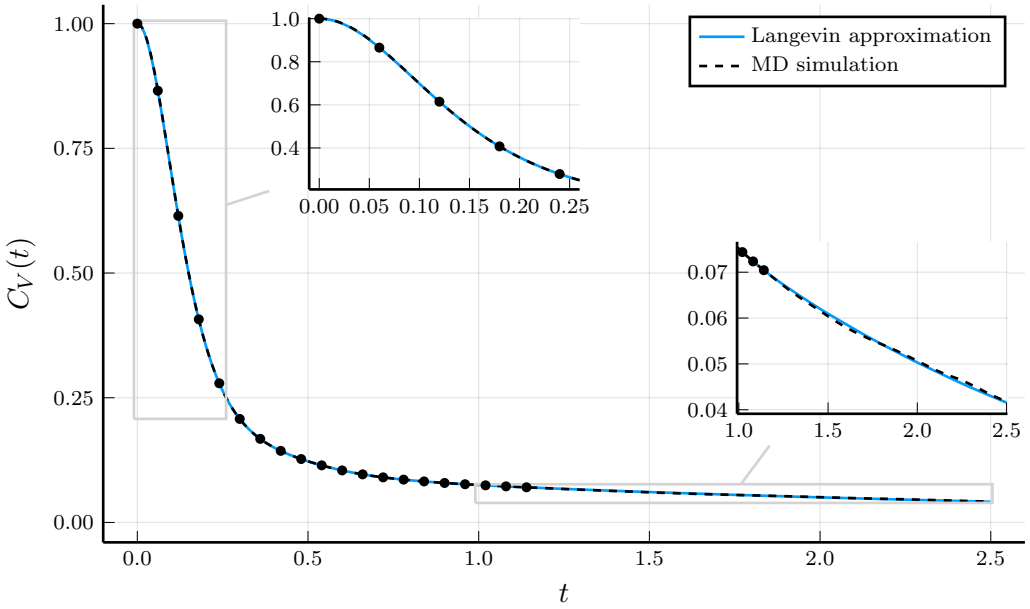


Figure 3.1.: VACF approximations for molecular dynamics data

a Langevin thermostat. The simulations were performed with the simulation package LAMMPS [41] using a time step size of $\Delta_t = 0.001$.

For the Lanczos method we use $n_0 = 20$ data points with a grid size of $\tau = 0.06$. No spurious or negative real eigenvalues were encountered, therefore the obtained matrix A has $N = 10$ rows and columns, i.e. 9 auxiliary variables. Figure 3.1 compares the input VACF and the obtained approximation. The two inset plots display enlarged parts of the large plot: one for small t and one for the long-time tail. Figure 3.2 shows the difference between the input VACF and the VACF of the obtained Ornstein-Uhlenbeck system. We clearly see a comparably large difference for $t \approx \tau$ (which is nonetheless small) since r_1 was adjusted during the Newton iteration. We also see that the approximation is very good up to the final grid point $(n_0 - 1)\tau = 1.14$ but gets significantly worse after the final grid point. Figure 3.3 displays the memory kernel of the obtained Ornstein-Uhlenbeck process compared to kernels obtained via solving the second order Volterra equation and via the iterative method presented in [25] (called IMRV).

We see that the VACF approximation is very good in the whole time interval. On top of that, the corresponding memory kernel is very close to the approximations obtained by other methods.

3.4.2. Noisy Data

We now test if the method is also applicable to noisy data. We use data generated by simulating a colloid in a fluid of 31627 Lennard-Jones particles with a diameter σ and mass m_0. The Lennard-Jones potentials are truncated at $2^{1/6}\sigma$. The cubic simulation box has periodic boundary conditions in all three dimensions (we use only one dimension for

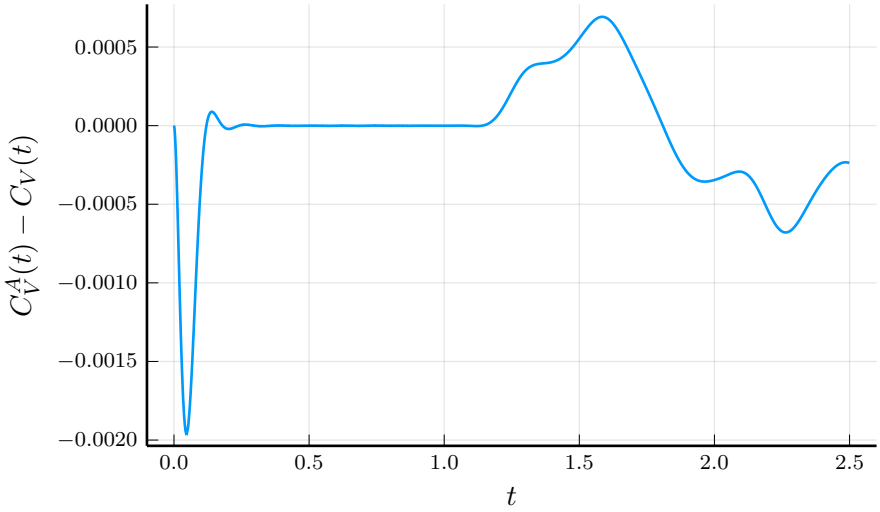


Figure 3.2.: Difference between VACF approximations and input VACF for noisy data

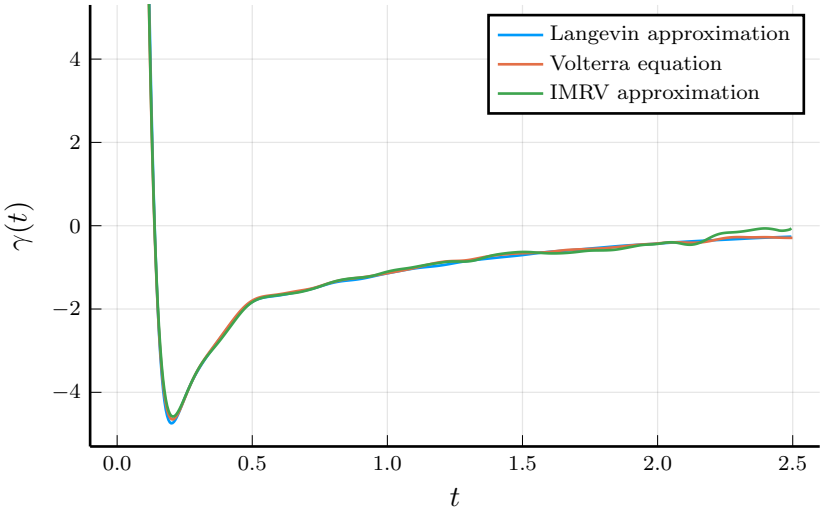


Figure 3.3.: Memory kernel approximations for molecular dynamics data

Noise setting	Grid	n_0	Newton steps	N_{sp}	N_{pt}	N	Relative error
Low	Δ_1	50	3	12	0	14	0.0639
	Δ_2	30	4	9	0	6	0.0428
	Δ_3	14	6	3	0	5	0.0582
Medium	Δ_1	49	3	9	0	17	0.0609
	Δ_2	30	4	5	0	11	0.0688
	Δ_3	–	–	–	–	–	–
High	Δ_1	50	3	8	0	17	0.1339
	Δ_2	29	4	7	0	9	0.1152
	Δ_3	14	4	2	0	5	0.2005

Table 3.1.: Results using the data from Section 4.3.2: Number of interpolation points n_0, number of Newton steps, number N_{sp} of spurious eigenvalues, number N_{pt} of other removed eigenvalues, size N of the final matrix A, and relative approximation error

applying our algorithm) and a side length of 35.76σ. The colloid has a mass of $80m_0$ and is defined as a rigid body with a radius 3σ. In order to sample at the desired temperature, the system was equilibrated with a Langevin thermostat. Again the simulations were performed using LAMMPS [41] with a time step of $\Delta_t = 0.001$.

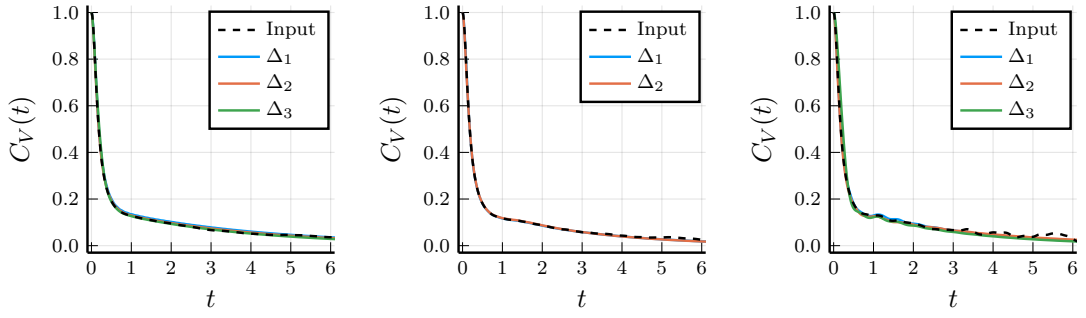
We compare three different noise levels, which differ by the length of the simulation run used to generate the data and by the equilibration time, i.e. the number of simulation steps performed before the actual simulation in order to ensure that the system is close to its equilibrium:

- Low noise: 5 000 000 simulation steps, 400 000 equilibration steps
- Medium noise: 1 000 000 simulation steps, 60 000 equilibration steps
- High noise: 100 000 simulation steps, 6 000 equilibration steps

Simulations for different noise levels were performed independently of each other. We used the following three grids for each noise level:

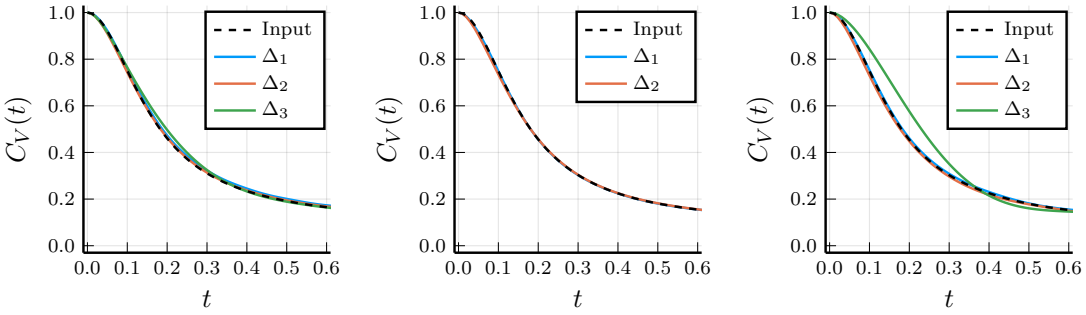
- Δ_1: $\tau = 0.06$, $n_0 = 50$
- Δ_2: $\tau = 0.1$, $n_0 = 30$
- Δ_3: $\tau = 0.2$, $n_0 = 15$

As mentioned before, it is not guaranteed that the algorithm succeeds. In case of failure, we reduced n_0 by 1 and reapplied the algorithm to the shortened grid. Table 3.1 shows for each noise level and each grid size the largest value of n_0 for which the algorithm succeeded. It also displays the necessary number of Newton steps, the number N_{sp} of spurious eigenvalues ($\lambda \in \sigma(A)$ with $\operatorname{Re}\lambda \geq 0$), the number N_{pt} of additional eigenvalues removed in order to obtain a VACF of positive type (i.e. eigenvalues $\lambda \in \sigma(A)$ which were removed although $\operatorname{Re}\lambda < 0$), the size N of the final matrix A (which depends on



(i) low noise (ii) medium noise (iii) high noise

Figure 3.4.: VACF approximations for noisy data with three noise levels



(i) low noise (ii) medium noise (iii) high noise

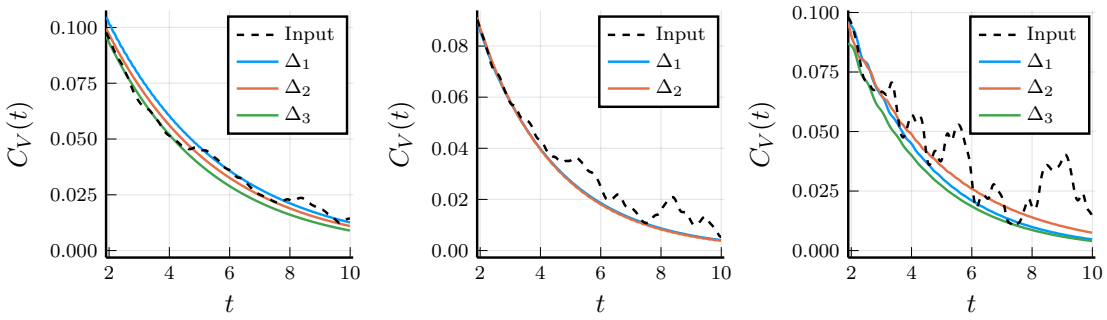
Figure 3.5.: VACF approximations for small t for all noise settings

the number of removed eigenvalues and the number of negative real eigenvalues), and the relative error

$$\frac{\int_0^{10} |C_V^A(t) - C_V(t)| \, \mathrm{d}t}{\int_0^{10} |C_V(t)| \, \mathrm{d}t},$$

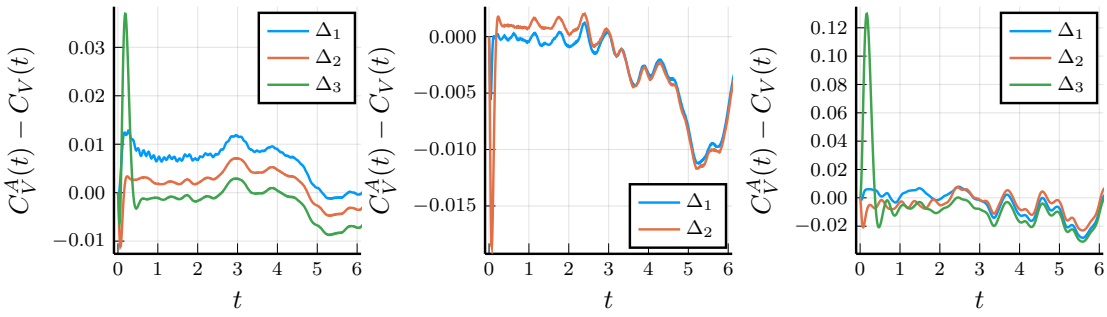
where the integrals are approximated by sums on the grid $\{0, 0.01, \ldots, 10\}$, on which the input VACF was given. We see that the algorithm immediately succeeded in four out of nine cases. In another four cases the second attempt was successful. In the medium noise setting with Δ_3 none of the values $n_0 = 15, \ldots, 8$ succeeded, thus we consider the algorithm to have failed in this case. In all successful cases only eigenvalues of A with positive real part had to be removed. Note that the approximation error is partially due to the noise in the input which does not exist in C_V^A, which is a desired effect.

Figure 3.4 compares the VACF approximations obtained for each grid in the three noise settings, where every plot shows one noise setting. Overall we obtain good approximations. Figures 3.5 and 3.6 show enlarged versions of Figure 3.4. Figure 3.5 only shows the time interval $t \in [0, 0.6]$, where C_V decays rapidly, while Figure 3.6 shows the tail of the VACF for $t \in [2, 10]$. Δ_3 in the high noise setting leads to a rather significant error for small t, which indicates that the chosen grid is too coarse to capture the rapid decrease of the



(i) low noise (ii) medium noise (iii) high noise

Figure 3.6.: VACF approximations for large t for all noise settings



(i) low noise (ii) medium noise (iii) high noise

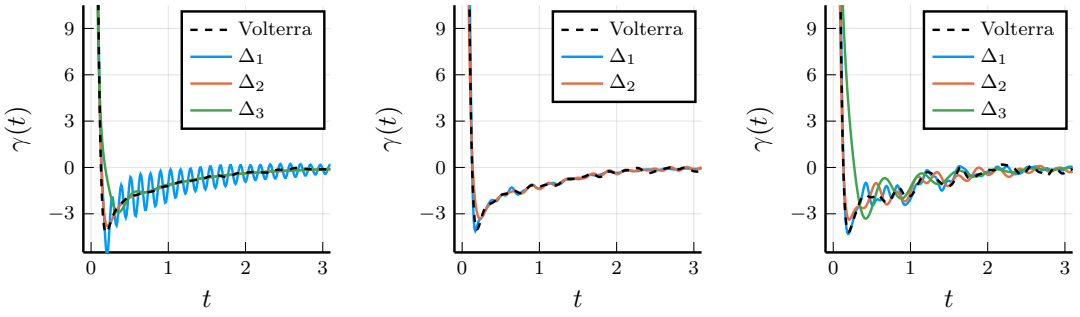
Figure 3.7.: Difference between VACF approximations and input VACF for noisy data

VACF for small t and the inflection point at $t \approx 0.1$. This is also true in the low noise setting although the error is smaller here. Figure 3.6 shows that despite the noise the algorithm manages to overall yield good approximations of the long-time tail of C_V for all three grids.

Figure 3.7 displays the difference between the obtained VACF approximations and the original VACF. It clearly shows the rather large error of Δ_3 for small t, too.

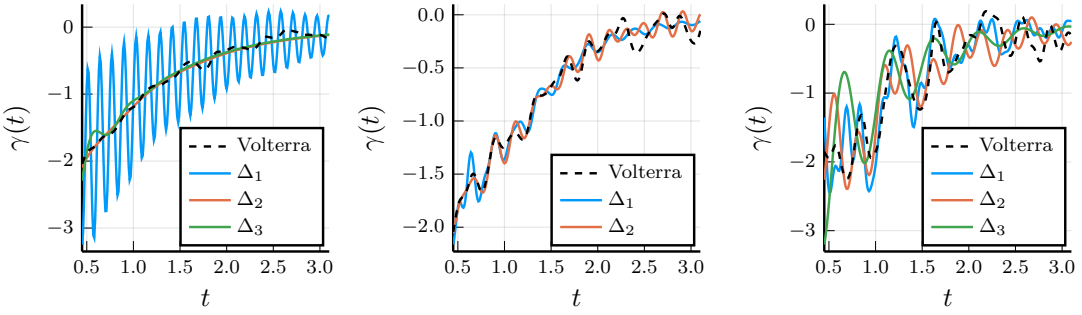
Figure 3.8 compares the obtained memory kernels of all three grids to those obtained by solving the (first kind) Volterra equation. Figure 3.9 shows an enlarged version for $t \geq 0.5$. The kernel corresponding to Δ_1 in the low noise settings exhibits a very strong oscillation, which, albeit weaker, is also visible in the VACF in Figure 3.7. This is due to a pair of complex conjugated eigenvalues $-0.5337 \pm 48.58\mathrm{i}$ of A_0 with significant coefficients $-0.7787 \mp 0.2443\mathrm{i}$ in the corresponding Prony series. Here the sum of the absolute values of all coefficients is 111.5, but the largest coefficients belong to the eigenvalues $-18.29 \pm 13.90\mathrm{i}$ with smallest (i.e. largest absolute) real part, which decay very rapidly in the Prony series. The sum of the absolute values of the remaining coefficients is 10.26.

A actually has two eigenvalues which are very close to those of A_0, $-0.5388 \pm 48.59\mathrm{i}$, and the corresponding oscillations can be seen in Figure 3.7; however, the Prony series



(i) low noise (ii) medium noise (iii) high noise

Figure 3.8.: Memory kernels obtained via the Lanczos method and via solving the Volterra equation for all three noise settings



(i) low noise (ii) medium noise (iii) high noise

Figure 3.9.: Memory kernels obtained via the Lanczos method and via solving the Volterra equation for all three noise settings

coefficients $3.59 \cdot 10^{-4} \pm 1.07 \cdot 10^{-4}$i in the VACF are much smaller than those in the memory kernel. Removing these two eigenvalues from A eliminates the oscillations from the VACF as shown in Figure 3.10. The remaining oscillations in the memory kernel (the red line on the right-hand side of Figure 3.10) are much smaller (comparable to the other obtained memory kernels) and have a smaller frequency, which indicates that they are caused by other eigenvalues.

Comparing the three grids in Figure 3.8, we see that the negative "peak" of the memory kernel is not well approximated by Δ_3, again indicating that this grid is too coarse. Δ_1 performs best in this regard. Δ_3, however, leads to the smallest oscillations for larger t. The difference is particularly well visible in the high noise setting. In general a good approximation of the memory kernel is not strictly necessary; a good approximation of the given VACF is more important to us.

In the low noise setting, Figure 3.7 shows that for $t \geq 0.5$ the three VACFs obtained by the different grids seem to differ mostly by a constant. Indeed this constant difference is due to the value of $A_{1,1}$ as modifying $A_{1,1}$ visually "shifts" the VACF up or down. Table 3.2 displays $A_{1,1}$ for each obtained Ornstein-Uhlenbeck system after applying the

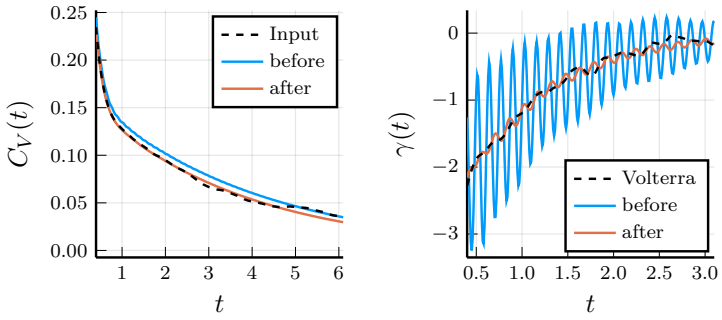


Figure 3.10.: VACF and memory kernel obtained using Δ_1 in the small noise setting before and after removing the pair of eigenvalues responsible for the oscillations

Grid	Low noise	Medium noise	High noise
Δ_1	-0.1157	-0.0004	-0.0576
Δ_2	-0.0284	-0.0115	0.0609
Δ_3	0.0163	–	0.0881

Table 3.2.: Value of $A_{1,1}$ after applying the Newton iteration and removing spurious eigenvalues

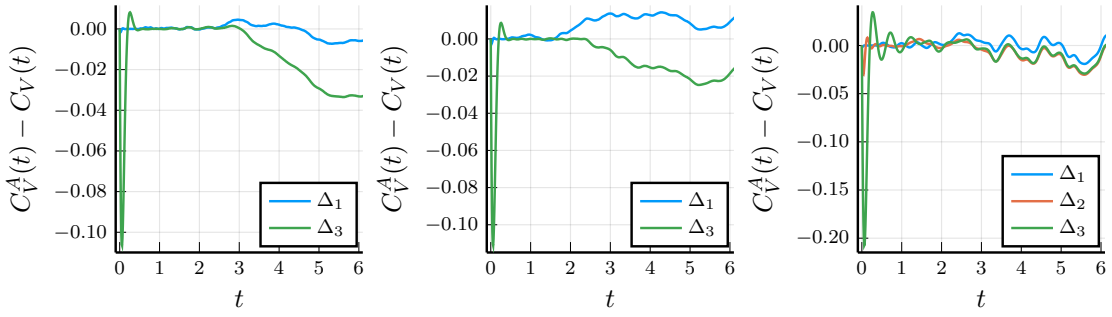
Newton iteration and removing spurious eigenvalues but before setting $A_{1,1}$ to $-\varepsilon$. For Δ_1 and Δ_2, this value is negative and setting it to $-\varepsilon$ "shifts" C_V^A up while for Δ_3 it is positive and C_V^A is "shifted" down by setting $A_{1,1}$ to $-\varepsilon$. A similar behavior can be observed in the medium noise case for $t \in [0.3, 3]$. If the condition $C_V'(0) = 0$ is not physically required, one can thus improve the approximation by not setting $A_{1,1}$ to $-\varepsilon$ provided it is already non-positive. Applying the Newton iteration still makes sense in this case as we will see immediately.

When $C_V'(0) = 0$ is not strictly required, one might ask if omitting the Newton iteration altogether leads to a better approximation. However, when we tested the algorithm without the iteration, it failed more often and for two settings (Δ_2 in the low and medium noise setting) it did not succeed for any n_0, where we tested all n_0 down to half the original n_0. In most cases where the algorithm failed due to $A_{1,1}$ being positive, $A_{1,1}$ was not close to 0, hence setting it to $-\varepsilon$ would have produced a VACF completely unsuited as approximation for the input VACF. Consequently it seems that although the removal of spurious eigenvalues can modify the value of $A_{1,1}$, the Newton iteration still helps in keeping its (absolute) value closer to 0. Table 3.3 displays the corresponding results, similarly to Table 3.1. We see that the approximation error does not improve either. We might of course be more lucky with other data, especially data where $C_V'(0) < 0$.

Figure 3.11 displays the differences between the VACF approximations and the input VACF for the cases in which the algorithm without Newton iteration succeeded. We see that although the error without Newton iteration is smaller for $\tau < t < 2\tau$ (compared to Figure 3.7) because τ is also interpolated, it is even larger for $t < \tau$ when using Δ_3. This holds for Δ_2 in the high noise setting (the only noise setting where both methods

Noise setting	Grid	Without Newton iteration					With Newton iteration					
		n_0	N_{sp}	N_{pt}	N	Relative error	n_0	Newton steps	N_{sp}	N_{pt}	N	Relative error
Low	Δ_1	39	6	0	14	0.0420	50	3	12	0	14	0.0405
	Δ_2	–	–	–	–	–	29	4	10	0	5	0.0460
	Δ_3	15	2	0	7	0.2380	15	8	2	2	3	0.1269
Medium	Δ_1	35	8	0	10	0.1061	49	3	9	0	17	0.0610
	Δ_2	–	–	–	–	–	30	4	5	0	11	0.0698
	Δ_3	15	2	0	7	0.1830	11	5	3	4	4	0.2137
High	Δ_1	46	9	0	14	0.0971	50	3	10	0	17	0.1429
	Δ_2	30	4	0	11	0.1467	23	4	2	0	11	0.1002
	Δ_3	15	2	0	7	0.1821	13	4	4	0	5	0.1485

Table 3.3.: Results when omitting setting $A_{1,1}$ to $-\varepsilon$ (without and with the Newton iteration): Number n_0 of interpolation points, number N_{sp} of spurious eigenvalues, number N_{pt} of other removed eigenvalues, size N of the final matrix, and relative approximation error



(i) low noise (ii) medium noise (iii) high noise

Figure 3.11.: Difference between VACF approximations and input VACF for noisy data without applying a Newton iteration and without setting $A_{1,1} = -\varepsilon$

succeed with this grid), too, although the difference between the errors of both methods in $[0, \tau]$ (with and without the Newton iteration) is smaller here. For Δ_1 the error in $[0, \tau]$ is not larger than the general approximation error introduced by removing spurious eigenvalues. Therefore the Newton iteration makes sense even if one does not require $A_{1,1} = 0$. Note that in our example the input VACF satisfies $C_V'(0) = 0$; the results may differ for data where this does not hold.

No stability problems were encountered with either data while applying the Lanczos method: The maximum error of an entry in a (normalized, i.e. with diagonal entries ± 1) Gram matrix was $3.79 \cdot 10^{-8}$. Therefore we did not apply any explicit orthogonalization techniques.

4. The Impact of a Constant Force

4.1. Description of the Modified Model

We now consider the situation of a macromolecule which is subject to a constant, known external force $F_V \in \mathbb{R}$. As in Chapter 3, we restrict ourselves to the case $d = 1$. The external force leads to the equation

$$V'(t) = F_V - \int_0^t \gamma(t-s)V(s)\,\mathrm{d}s + F(t)$$

with γ and F as in the generalized Langevin equation (2.3) or, with an additional distributional term in the kernel, to

$$\mathrm{d}V(t) = \left(F_V + DV(t) - \int_0^t \gamma(t-s)V(s)\,\mathrm{d}s + F(t)\right)\mathrm{d}t + \frac{1}{\sqrt{\beta m}} L\,\mathrm{d}W(t) \tag{4.1}$$

with $D < 0$, $L > 0$ as in (2.5). Just as F in (2.3) and (2.5), the external force has been scaled by m for simplicity. In the following we will consider the more general equation (4.1). Analogously to Section 2.1 there exists a version of (4.1) with the integral starting at $-\infty$, where the joint process of V and F has a stationary distribution, but we will not discuss this version here.

Let $\bar{V}(t) := \mathbf{E}[V(t)]$. Integrating (4.1) and taking the expectation yields

$$\begin{aligned}
\bar{V}(t) - \bar{V}(0) &= \mathbf{E}\left[\int_0^t \mathrm{d}V(s)\right] \\
&= tF_V + D\mathbf{E}\left[\int_0^t V(s)\,\mathrm{d}s\right] - \mathbf{E}\left[\int_0^t \int_0^s \gamma(s-u)V(u)\,\mathrm{d}u\,\mathrm{d}s\right] + \\
&\quad \mathbf{E}\left[\int_0^t F(s)\,\mathrm{d}s\right] + \mathbf{E}\left[\int_0^t \frac{1}{\sqrt{\beta m}} L\,\mathrm{d}W(s)\right] \\
&= tF_V + D\int_0^t \bar{V}(s)\,\mathrm{d}s - \int_0^t \int_0^s \gamma(s-u)\bar{V}(u)\,\mathrm{d}u\,\mathrm{d}s.
\end{aligned}$$

All remaining terms are differentiable and differentiating shows that $\bar{V}$ satisfies

$$\bar{V}'(t) = F_V + D\bar{V}(t) - \int_0^t \gamma(t-s)\bar{V}(s)\,\mathrm{d}s \qquad \text{for } t > 0, \qquad \bar{V}(0) = \bar{V}_0. \tag{4.2}$$

By subtracting $\bar{V}(t)$ from (4.1), we see that $\widetilde{V}(t) := V(t) - \bar{V}(t)$ solves the GLE (2.5):

$$\mathrm{d}\widetilde{V}(t) = \left(D\widetilde{V}(t) - \int_0^t \gamma(t-s)\widetilde{V}(s)\,\mathrm{d}s + F(t)\right)\mathrm{d}t + \frac{1}{\sqrt{\beta m}} L\,\mathrm{d}W(t).$$

In the following we will assume that $\widetilde{V}$ is a stationary process. Consequently $C_{\widetilde{V}}$ solves the Volterra equation (2.10) and since V differs from $\widetilde{V}$ only by a deterministic term, $C_V = C_{\widetilde{V}}$. In contrast to (2.5), however, (4.1) has no stationary solution since then $\bar{V}(t) = \bar{V}_0$ would be constant and consequently $\bar{V}'(t) = \mathbf{0}$, in which case (4.2) would imply

$$F_V = \left(\int_0^t \gamma(s)\,\mathrm{d}s - D\right)\bar{V}_0$$

for every $t \geq 0$, where the left-hand side is independent of t while the right-hand side is not.

Since (4.2) is a Volterra equation similar to (2.8), its solution can be written explicitly using the differential resolvent, analogously to (2.9), cf. [19, Theorem 3.3.3]:

$$\bar{V}(t) = r(t)\bar{V}_0 + \int_0^t r(t-s)F_V\,\mathrm{d}s$$

where r is the solution of (2.8). If $r(t) \xrightarrow{t\to\infty} 0$, this implies

$$\bar{V}(t) \xrightarrow{t\to\infty} \int_0^\infty r(s)\,\mathrm{d}s \cdot F_V =: \bar{V}_\infty.$$

Taking the limit in (4.2) now shows that the right-hand side converges and since $\bar{V}'(t)$ cannot converge to another value than $\mathbf{0}$ if $\bar{V}(t)$ converges, (4.2) implies $\bar{V}'(t) \xrightarrow{t\to\infty} \mathbf{0}$ and therefore

$$F_V = \left(\int_0^\infty \gamma(s)\,\mathrm{d}s - D\right)\bar{V}_\infty$$

i.e.

$$\bar{V}_\infty = \mu F_V \quad \text{with} \quad \mu := \left(\int_0^\infty \gamma(t)\,\mathrm{d}t - D\right)^{-1} = \beta m \int_0^\infty C_V(t)\,\mathrm{d}t \tag{4.3}$$

(recall that $C_V(t) = \frac{1}{\beta m} r(t)$ for $t \geq 0$.) μ is denoted as the system's *mobility*, cf. [28, Equation (4.9)]. Consequently we will later assume that t is large enough and $\bar{V}(t) \approx \bar{V}_\infty$. (When using the stationary equation with the integral in (4.1) starting at $-\infty$, setting $\bar{V}(0) = \bar{V}_\infty$ implies $\bar{V}(t) = \bar{V}_\infty$ for all t. In this case if $\widetilde{V}$ is stationary, so is V.)

In the case of an Ornstein-Uhlenbeck system

$$\gamma(t) = B^T e^{tA_0} C, \qquad C_V(t) = \frac{1}{\beta m} e_1^T e^{tA} e_1 \quad \text{with} \quad A = \begin{bmatrix} D & B^T \\ -C & A_0 \end{bmatrix}$$

μ satisfies

$$\mu = \beta m \int_0^\infty C_V(t)\,\mathrm{d}t = \int_0^\infty e_1^T e^{tA} e_1\,\mathrm{d}t = -e_1^T A^{-1} e^{0\cdot A} e_1 = -(A^{-1})_{1,1} \tag{4.4}$$

and

$$\mu^{-1} = B^T\Big(\int_0^\infty e^{tA_0}\,\mathrm{d}t\Big)C - D = -B^T A_0^{-1}C - D. \tag{4.5}$$

((4.5) also follows from (4.4) and Definition and Lemma 1.1.)

Since $\widetilde{V}$ solves the generalized Langevin equation without external force, the algorithm from Chapter 3 allows for the computation of an Ornstein-Uhlenbeck system (A, K) for $\widetilde{V}$. Provided this Ornstein-Uhlenbeck system has the same mobility as (4.1), $V + \bar{V}_\infty$ can be represented by a generalized Ornstein-Uhlenbeck process, as the following lemma shows. For $t \to \infty$ this allows an asymptotic representation of V by a generalized Ornstein-Uhlenbeck process.

Lemma 4.1. *Let V be a solution of (4.1) where $\gamma(t) = B^T e^{tA_0}C$ and let (A, K) be an Ornstein-Uhlenbeck system such that $\widetilde{X} = \begin{bmatrix}\widetilde{V}\\ Z\end{bmatrix}$ solves*

$$\mathrm{d}\widetilde{X}(t) = A\widetilde{X}(t)\,\mathrm{d}t + K\,\mathrm{d}W(t), \qquad \widetilde{X}(0) = \widetilde{X}_0 \sim \mathcal{N}(0, \Sigma)$$

with Z from Theorem 2.6 and $\widetilde{V}(t) := V(t) - \bar{V}(t)$ and such that A, K, and Σ solve the Lyapunov equation (1.11). Let μ be given by (4.5). Then $X := \begin{bmatrix}\widetilde{V} + \bar{V}_\infty\\ Z\end{bmatrix}$ solves the inhomogeneous Ornstein-Uhlenbeck equation

$$\mathrm{d}X(t) = F_X\,\mathrm{d}t + AX(t)\,\mathrm{d}t + K\,\mathrm{d}W(t), \qquad X(0) = \widetilde{X}_0 + \bar{X}_\infty. \tag{4.6}$$

with

$$F_X = \begin{bmatrix}F_V\\ \mathbf{0}\end{bmatrix}, \quad \bar{X}_\infty = \begin{bmatrix}\bar{V}_\infty\\ A_0^{-1}C\bar{V}_\infty\end{bmatrix}, \quad \textit{and} \quad \bar{V}_\infty = \mu F_V.$$

Proof. A Gaussian process Y is a solution of (4.6) if

$$\begin{aligned} Y(t) &= Y(0) + \int_0^t F_X\,\mathrm{d}s + \int_0^t AY(s)\,\mathrm{d}s + \int_0^t K\,\mathrm{d}W(s)\\ &= Y(0) + tF_X + \int_0^t AY(s)\,\mathrm{d}s + KW(t) \end{aligned} \tag{4.7}$$

with $Y(0) = \widetilde{X}_0 + \bar{X}_\infty$. $\widetilde{X}$ is an Ornstein-Uhlenbeck process and therefore (see (1.9))

$$X(t) = \widetilde{X}(t) + \bar{X}_\infty = e^{tA}\widetilde{X}_0 + \bar{X}_\infty + \int_0^t e^{(t-s)A}K\,\mathrm{d}W(s).$$

Analogously to the calculation of the solution of an Ornstein-Uhlenbeck process (see Theorem 1.18), integration leads to

$$\begin{aligned} \int_0^t X(s)\,\mathrm{d}s &= \int_0^t e^{sA}\widetilde{X}_0\,\mathrm{d}s + t\bar{X}_\infty + \int_0^t\int_0^s e^{(s-u)A}K\,\mathrm{d}W(u)\,\mathrm{d}s\\ &= A^{-1}e^{sA}\Big|_0^t\widetilde{X}_0 + t\bar{X}_\infty + \int_0^t KW(s)\,\mathrm{d}s + \int_0^t\int_0^s Ae^{(s-u)A}KW(u)\,\mathrm{d}u\,\mathrm{d}s \end{aligned}$$

$$
\begin{aligned}
&= A^{-1}(e^{tA} - \mathbf{I})\widetilde{X}_0 + t\bar{X}_\infty + \int_0^t KW(s)\,\mathrm{d}s + \int_0^t \int_u^t Ae^{sA}\,\mathrm{d}s\, e^{-uA} KW(u)\,\mathrm{d}u \\
&= A^{-1}(e^{tA} - \mathbf{I})\widetilde{X}_0 + t\bar{X}_\infty + \int_0^t KW(s)\,\mathrm{d}s + \int_0^t e^{(t-u)A} KW(u)\,\mathrm{d}u - \\
&\quad \int_0^t KW(u)\,\mathrm{d}u.
\end{aligned}
$$

By inserting into the right-hand side of (4.7) with $Y := X$, we obtain

$$
\begin{aligned}
&X(0) + tF_X + \int_0^t AX(s)\,\mathrm{d}s + KW(t) \\
&\quad = \widetilde{X}_0 + \bar{X}_\infty + tF_X + A\Big(A^{-1}(e^{tA} - \mathbf{I})\widetilde{X}_0 + t\bar{X}_\infty + \int_0^t e^{(t-u)A} KW(u)\,\mathrm{d}u\Big) + \\
&\quad\quad KW(t) \\
&\quad = e^{tA}\widetilde{X}_0 + \bar{X}_\infty + tF_X + tA\bar{X}_\infty + \int_0^t Ae^{(t-u)A} KW(u)\,\mathrm{d}u + KW(t) \\
&\quad = e^{tA}\widetilde{X}_0 + \bar{X}_\infty + tF_X + tA\bar{X}_\infty + \int_0^t e^{(t-u)A} K\,\mathrm{d}W(u) \\
&\quad = X(t) + tF_X + tA\bar{X}_\infty \\
&\quad = X(t) + t\begin{bmatrix} F_V + (D + B^T A_0^{-1} C)\bar{V}_\infty \\ \mathbf{0} - C\bar{V}_\infty + A_0 A_0^{-1} C\bar{V}_\infty \end{bmatrix} \\
&\quad = X(t) + t\begin{bmatrix} F_V - \mu^{-1}\bar{V}_\infty \\ \mathbf{0} \end{bmatrix} \\
&\quad = X(t)
\end{aligned}
$$

where we have used (4.5) in the penultimate step. Consequently $Y := X$ solves (4.7), i.e., X is a solution of (4.6). □

On the other hand if $X = \begin{bmatrix} V \\ Z \end{bmatrix}$ is a solution of (4.6), then $V(t) - \bar{V}_\infty + \bar{V}(t)$ solves (4.1).

In order to find an inhomogeneous Ornstein-Uhlenbeck process which approximates the solution of (4.1), we have to ensure that the Ornstein-Uhlenbeck process solving the generalized Langevin equation for $\widetilde{V}$ has the correct mobility. Thus we obtain from (4.4) another condition

$$
-(A^{-1})_{1,1} = \int_0^\infty (e^{tA})_{1,1}\,\mathrm{d}t \stackrel{!}{=} \mu \tag{4.8}
$$

to A apart from (3.5). Our goal is to modify the method from Chapter 3 in a way that it yields an Ornstein-Uhlenbeck system which fulfills both conditions. As in Section 3.2.1 we will do so by slightly modifying the input data r. However, (4.8) leads to the assumption that modifying only individual data points r_j does not make sense here as these influence the integral only marginally. Instead we will scale all data points except r_0 (which is given by $C_V(0) = \frac{1}{\beta m}$) and r_1 (which we already modify individually in order to satisfy (3.5)) by a common factor c_μ.

4.2. Adaptations to the Algorithm

In this chapter we will present two adaptations of the algorithm from Section 3.3 which compute an Ornstein-Uhlenbeck system whose mobility is close to the mobility of the input VACF. For simplicity we assume $r_0 = \frac{1}{\beta m} = 1$ again.

We first estimate $\bar{V}_\infty$ and μ and thus obtain F_V (or we determine μ from $\bar{V}_\infty$ and F_V, depending on what is more feasible). Subsequently we only consider the centered velocity $\widetilde{V}(t) := V(t) - \bar{V}_\infty$, which has the same auto-covariance function $C_{\widetilde{V}}(t) = C_V(t)$. To $\widetilde{V}$ we apply the algorithm from Chapter 3 to determine an Ornstein-Uhlenbeck system. We now want this system to satisfy (4.8) on top of (3.5).

One possibility is to extend the Newton iteration from Section 3.2.1 to a two-dimensional Newton iteration which satisfies the constraints (3.5) and (4.8): In each Newton step, $(r_0, r_1, r_2, \ldots, r_{n_0-1})$ is replaced by $(r_0, \widetilde{r}_1, c_\mu r_2, \ldots, c_\mu r_{n_0-1})$ with suitable $\widetilde{r}_1$ and c_μ, using the derivative with respect to r_1 as well as the directional derivative with respect to the directional vector $(0, 0, r_2, \ldots, r_{n_0-1})$. If the actual mobility of the corresponding physical system is known, it can be directly compared with the mobility of the generated Ornstein-Uhlenbeck system. The derivative of $-A^{-1}$ is $A^{-1}(\partial A)A^{-1}$ where ∂A denotes the derivative of A, cf. [37]. Instead of Newton's method, one can also think at other methods to solve both equations, such as the Levenberg-Marquardt method or gradient descent methods.

In the case that μ is unknown, we have to approximate it via approximating the right-hand integral in (4.3). In this case, we should approximate the mobility of the obtained Ornstein-Uhlenbeck system in the same way to ensure that the impact of approximation errors, e.g. due to a significant long-time tail in the VACF, is similar for both values. When we apply Newton's method, however, A can still contain spurious eigenvalues (with non-negative real part), in which case the integral in (4.8) does not exist. Since removing spurious eigenvalues requires an eigenvalue decomposition of J, it cannot be done inside the two-dimensional Newton iteration as that would require differentiation of the eigenvalue decomposition. Therefore we have to compare the mobility approximated from the input data with the exact mobility of the Ornstein-Uhlenbeck system. This, however, can entail a large error if the contribution of the tail, which is not accounted for by the quadrature formula (and is, possibly, not known at all), to the integral is large. In the following we will present two ways to solve this problem.

4.2.1. A Nested Iterative Algorithm

In the first adaptation to our algorithm we extend the Newton iteration by a second iteration procedure, obtaining two nested iterations which each satisfy one of the two conditions (3.5) and (4.8). The inner iteration is the Newton iteration from Section 3.2.1 while each step of the the outer iteration determines a factor c_μ by which $r_2, \ldots, r_{n-1}$ are multiplied. The combination of both iterations hopefully satisfies both conditions, which we will assess in Section 3.4.

More precisely, we adapt the algorithm as follows: After applying the Lanczos method, we adjust r_1 just as in Section 3.2.1, using a one-dimensional Newton iteration, until $A_{1,1} \approx 0$. Next we remove spurious eigenvalues and duplicate negative real eigenvalues as described in Sections 3.2.2 and 3.2.3. We use the resulting matrix A to calculate the VACF of the corresponding Ornstein-Uhlenbeck system and we approximate its mobility μ_A as well as the mobility μ of the true system. Next we estimate the "contribution" of $r_2, \dots, r_{n_0-1}$ to μ_A (described below in detail) in order to determine the scaling factor c_μ for $r_2, \dots, r_{n_0-1}$. We subsequently replace $r_2, \dots, r_{n_0-1}$ by $c_\mu r_2, \dots, c_\mu r_{n_0-1}$.

Since (3.5) does no longer need to hold after this modification, we iterate this procedure, i.e. we continue by again carrying out several Newton steps until $A_{1,1} \approx 0$, removing spurious eigenvalues of the new matrix A and scaling $r_2, \dots, r_{n_0-1}$ with a new constant c_μ. In this way we obtain the nested iteration scheme illustrated in Algorithm 1.

For the approximation of μ and μ_A from (4.3) and (4.8), respectively, we employ the trapezoidal rule on a finer time grid $\widetilde{\Delta} = \{0, \widetilde{\tau}, 2\widetilde{\tau}, \dots, \widetilde{n}\widetilde{\tau}\}$ with $\widetilde{\tau} = \frac{\tau}{k}$, $k \in \mathbb{N}$ (if VACF data on such a grid is available; otherwise we have to use the coarse grid $\widetilde{\tau} = \tau$):

$$\mu \approx \frac{\widetilde{\tau}}{2} C_V(0) + \sum_{j=1}^{\widetilde{n}} \widetilde{\tau} C_V(j\widetilde{\tau}), \qquad \mu_A \approx \frac{\widetilde{\tau}}{2} C_V^A(0) + \sum_{j=1}^{\widetilde{n}} \widetilde{\tau} C_V^A(j\widetilde{\tau}). \tag{4.9}$$

(We use the final grid point $\widetilde{n}\widetilde{\tau}$ with full weight 1 instead of weight $\frac{1}{2}$, as one would do when approximating a finite interval.) We also estimate the "contribution" μ_A^* of $r_2, \dots, r_{n-1}$ to the mobility with the trapezoidal rule on the same grid. To obtain an estimate of this contribution, only grid points $j\widetilde{\tau} \geq 2\tau$ are considered with full weight 1 while grid points $j\widetilde{\tau} \leq \tau$ are omitted. For grid points $j\widetilde{\tau} \in (\tau, 2\tau)$ we interpolate their weight in the sum linearly from 0 to 1. This results in the estimate

$$\mu_A^* := \sum_{j=k+1}^{2k-1} \left(\frac{j}{k} - 1 \right) \widetilde{\tau} C_V(j\widetilde{\tau}) + \sum_{j=2k}^{\widetilde{n}} \widetilde{\tau} C_V(j\widetilde{\tau}). \tag{4.10}$$

The scaling factor c_μ is then chosen in such a way that $\mu = \mu_A + (c_\mu - 1)\mu_A^*$ holds, i.e.

$$c_\mu = 1 + \frac{\mu - \mu_A}{\mu_A^*}. \tag{4.11}$$

During the iteration process we will denote by $c_\mu^{(k)}$ the scaling factor in the kth iteration step, by which the values $r_2, \dots, r_{n-1}$ obtained in the $(k-1)$th step are multiplied, i.e. c_μ is the product of all $c_\mu^{(k)}$.

Remark 4.2. One can also think of scaling r_1 together with $r_2, \dots, r_{n_0-1}$ (and adjusting the calculation of μ_A^* and c_μ accordingly). This means that r_1 is modified both by the Newton iteration and by the multiplication of all values with c_μ. In our numerical experiments presented in Section 4.3, this modification did not result in any noticeable qualitative differences in the results, which is why we will not consider it there.

Algorithm 1 Nested iteration scheme for satisfying (3.5) and (4.8)

Input: $r = (r_0, \dots, r_{n_0-1})$ with $r_0 = 1$

Approximate $\mu \leftarrow \displaystyle\int_0^\infty C_V(t)\,\mathrm{d}t \approx \frac{\widetilde{\tau}}{2} C_V(0) + \widetilde{\tau} \sum_{j=1}^{\widetilde{n}} C_V(j\widetilde{\tau})$.

while $\frac{|\mu - \mu_A|}{\mu} \geq \delta_\mu$ **do**

Compute A and its derivative ∂A w.r.t. r_1 using the Lanczos algorithm.

while $|A_{1,1}| \geq \delta_A$ **do**

Replace r_1 by $r_1 - \frac{\operatorname{Re} A_{1,1}}{\operatorname{Re} \partial A_{1,1}}$.

Recompute A and ∂A using the Lanczos algorithm.

end while

Remove spurious eigenvalues and duplicate negative real eigenvalues.

Approximate $\mu_A \leftarrow \displaystyle\int_0^\infty C_V^A(t)\,\mathrm{d}t \approx \frac{\widetilde{\tau}}{2} C_V^A(0) + \widetilde{\tau} \sum_{j=1}^{\widetilde{n}} C_V^A(j\widetilde{\tau})$.

Set $\mu_A^* \leftarrow \displaystyle\sum_{j=k+1}^{2k-1} \Big(\frac{j}{k} - 1\Big) \widetilde{\tau} C_V^A(j\widetilde{\tau}) + \widetilde{\tau} \sum_{j=2k}^{\widetilde{n}} C_V^A(j\widetilde{\tau})$.

Set $c_\mu^{(k)} \leftarrow 1 + \frac{\mu - \mu_A}{\mu_A^*}$ and replace $r_2, \dots, r_{n-1}$ by $c_\mu r_2, \dots, c_\mu r_{n_0-1}$.

end while

Remove spurious eigenvalues and duplicate negative real eigenvalues.

if (3.5) is required **then**

Set $A_{1,1} \leftarrow -\varepsilon$.

end if

$\delta_\mu > 0$ and $\delta_A > 0$ in Algorithm 1 are tolerances for testing whether a certain value is close to 0. In both loops we also specify a maximum number of steps after which the loop is canceled. (These are omitted in Algorithm 1 for simplicity.) If the outer iteration is canceled for this reason, the algorithm fails as we assume that it does not converge. When the inner loop (the Newton iteration) reaches the maximum number of steps, the algorithm continues normally after the loop (and will start another Newton iteration in the next outer iteration step).

As in Chapter 3 we cannot guarantee that after applying Algorithm 1 and removing spurious eigenvalues (3.5) still holds. Hence at the end we set $A_{1,1}$ to $-\varepsilon$ with a small $\varepsilon > 0$.

4.2.2. A Modified Newton Iteration

Computing μ_A in (4.9) requires removing spurious eigenvalues from A first. Therefore Algorithm 1 can only depend on μ_A but not on its derivative since the latter cannot be computed. Consequently it is not possible to employ a Newton iteration to adjust r accordingly.

Alternatively we can restrict ourselves to the interpolation grid $0, \tau, \dots, (n_0 - 1)\tau$ for

estimating μ and μ_A, i.e.

$$\mu \approx \widetilde{\mu} := \frac{\tau}{2} r_0 + \sum_{j=1}^{n_0-1} \tau r_j, \qquad \mu_A \approx \widetilde{\mu}_A := \frac{\tau}{2} C_V^A(0) + \sum_{j=1}^{n_0-1} \tau C_V^A(j\tau).$$

As described above, we modify r by replacing r_1 by a value $\widetilde{r}_1$ and multiplying $r_2, \ldots, r_{n_0-1}$ by a factor c_μ in order to satisfy the two conditions $A_{1,1} = 0$ and $\mu = \widetilde{\mu}_A$. Consequently $A_{1,1}$ and $\widetilde{\mu}_A$ can be regarded as functions of $\widetilde{r}_1$ and c_μ, where

$$\widetilde{\mu}_A(\widetilde{r}_1, c_\mu) = \frac{\tau}{2} r_0 + \tau \widetilde{r}_1 + c_\mu \sum_{j=2}^{n_0-1} \tau r_j$$

and $A_{1,1}$ is given by the result of the Lanczos method. The value of c_μ for which $\widetilde{\mu}_A(\widetilde{r}_1, c_\mu) = \widetilde{\mu}$ holds can be easily computed from $\widetilde{r}_1$ (assuming that the above approximations of μ and μ_A are exact):

$$c_\mu = 1 + (r_1 - \widetilde{r}_1) \left(\sum_{j=2}^{n_0-1} r_j \right)^{-1}. \tag{4.12}$$

Hence we can write c_μ as a function of $\widetilde{r}_1$; then $A_{1,1}$ depends on $\widetilde{r}_1$ only. Our second algorithm employs a Newton iteration in order to find an $\widetilde{r}_1$ such that $A_{1,1}(\widetilde{r}_1, c_\mu(\widetilde{r}_1)) \stackrel{!}{=} 0$. When applying the Lanczos algorithm, we compute the derivatives of $A_{1,1}(\widetilde{r}_1, c_\mu)$ w.r.t. $\widetilde{r}_1$ and to c_μ. The derivative $c'_\mu(\widetilde{r}_1)$ is given via (4.12) as

$$c'_\mu(\widetilde{r}_1) = -\left(\sum_{j=2}^{n_0-1} r_j \right)^{-1}$$

and the chain rule yields the derivative of $A_{1,1}(\widetilde{r}_1, c_\mu(\widetilde{r}_1))$ w.r.t. $\widetilde{r}_1$. Compared to Section 4.2.1, this should lead to a better rate of convergence (quadratic rather than linear) at the cost of lower accuracy due to the coarser grid. (Depending on the available data, however, choosing $\widetilde{\tau} < \tau$ might be impossible anyway.)

Spurious eigenvalues have no impact on the Newton iteration due to the chosen grid. Therefore they are dealt with only after the iteration converged, see Section 3.2. Algorithm 2 describes the complete procedure. Again $\delta_A > 0$ is a tolerance for testing if $|A_{1,1}| \approx \delta_A$.

4.3. Numerical results

In the following we study how well the mobility of the data from Section 3.4 can be approximated by Algorithms 1 and 2. We use the same data as in Section 3.4. Note that these data were generated without external force, so we will assume that they already represent the centered data $\widetilde{V} := V - \bar{V}_\infty$ from Section 4.2 and will only try to find an Ornstein-Uhlenbeck system approximating the given velocity auto-covariance function

Algorithm 2 Newton iteration scheme for satisfying (3.5) and (4.8)

Input: $r = (r_0, \ldots, r_{n_0-1})$

Set $\mu \leftarrow \tau \sum_{j=0}^{n} r_j$.
Set $\widetilde{r}_1 \leftarrow r_1$ and $c_\mu \leftarrow 1$.
while $|A_{1,1}| > \delta_A$ **do**
  Set $c_\mu \leftarrow 1 + (r_1 - \widetilde{r}_1)\left(\sum_{j=2}^{n_0-1} r_j\right)^{-1}$ and $c'_\mu \leftarrow -\left(\sum_{j=2}^{n_0-1} r_j\right)^{-1}$.
  Compute $A(\widetilde{r}_1, c_\mu)$ and its derivatives $\partial_{\widetilde{r}_1} A(\widetilde{r}_1, c_\mu)$ and $\partial_{c_\mu} A(\widetilde{r}_1, c_\mu)$ w.r.t. $\widetilde{r}_1$ and c_μ using the Lanczos algorithm.
  Set $\partial A \leftarrow \partial_{r_1} A + \partial_{c_\mu} A \cdot c'_\mu$.
  Replace r_1 by $r_1 - \frac{\operatorname{Re} A_{1,1}}{\operatorname{Re} \partial A_{1,1}}$.
end while
Remove spurious eigenvalues and duplicate negative real eigenvalues.
if (3.5) is required **then**
  Set $A_{1,1} \leftarrow -\varepsilon$.
end if

which produces the "correct" mobility. Since we do not know the actual mobility, we will estimate it in a rather simple way using the trapezoidal rule as in (4.9). Again we restrict our attention to the normalized data ($r_0 = \frac{1}{\beta m} = 1$) for approximating the mobility.

4.3.1. Molecular Dynamics Data

First we study the data from Section 3.4.1. Approximating the mobility of the input VACF yields an estimate $\mu = 0.32772017$, where we use the grid $\widetilde{\Delta} = \{0, \widetilde{\tau}, 2\widetilde{\tau}, \ldots, \widetilde{n}\widetilde{\tau}\}$ with $\widetilde{\tau} = 0.001$ and $\widetilde{n} = 2499$ for the approximation. Note that this velocity autocovariance function has a long, significant tail (see Figure 3.1) beyond the observed time interval, which seriously impedes approximating the mobility this way. This estimate is therefore very inaccurate; for serious attempts of estimating μ one should employ other methods. Since we are not interested in determining the exact value of μ, we will ignore this inaccuracy and proceed as if this value was the correct mobility. As long as we estimate μ and μ_A via the same method, we should still obtain good VACF approximations.

In Algorithm 1 we choose $\delta_A = \delta_\mu = 10^{-8}$ and perform at most five Newton steps in each outer iteration step. In Algorithm 2 we choose $\delta_A = 10^{-8}$ and perform at most ten Newton steps. Besides the interpolation grid from Section 3.4.1, we try three other interpolation grids, which leads to the following four grids:

- Δ_1: $\tau = 0.03$, $n_0 = 40$
- Δ_2: $\tau = 0.04$, $n_0 = 30$
- Δ_3: $\tau = 0.06$, $n_0 = 20$
- Δ_4: $\tau = 0.12$, $n_0 = 10$

Grid	n_0	N	Outer steps	Newton steps	$A_{1,1}$	c_μ	$\widetilde{r}_1$	$\frac{\widetilde{r}_1}{r_1}$
Δ_1	39	14	6	27	$1.33 \cdot 10^{-2}$	0.9889	0.9545	0.9909
Δ_2	28	10	8	25	$2.04 \cdot 10^{-3}$	0.9978	0.9351	0.9985
Δ_2	23	10	6	28	$-8.89 \cdot 10^{-3}$	1.0031	0.9326	0.9958
Δ_3	20	10	6	10	$1.14 \cdot 10^{-7}$	1.0005	0.8658	0.9983
Δ_4	10	6	6	7	$-4.45 \cdot 10^{-8}$	0.9999	0.6145	0.9997

Table 4.1.: Summarized results of Algorithm 1

		without setting $A_{1,1} = -\varepsilon$		after setting $A_{1,1} = -\varepsilon$	
Grid	n_0	μ_A	$\left\lvert\frac{\mu-\mu_A}{\mu}\right\rvert$	μ_A	$\left\lvert\frac{\mu-\mu_A}{\mu}\right\rvert$
Δ_1	39	(0.327 718)	$7.00 \cdot 10^{-6}$	0.326 469	$3.82 \cdot 10^{-3}$
Δ_2	28	(0.327 720)	$4.80 \cdot 10^{-7}$	0.327 527	$5.91 \cdot 10^{-4}$
Δ_2	23	0.327 720	$2.31 \cdot 10^{-10}$	0.328 568	$2.59 \cdot 10^{-3}$
Δ_3	20	(0.327 720)	$1.56 \cdot 10^{-10}$	0.327 720	$3.28 \cdot 10^{-8}$
Δ_4	10	0.327 720	$1.45 \cdot 10^{-10}$	0.327 720	$1.27 \cdot 10^{-8}$

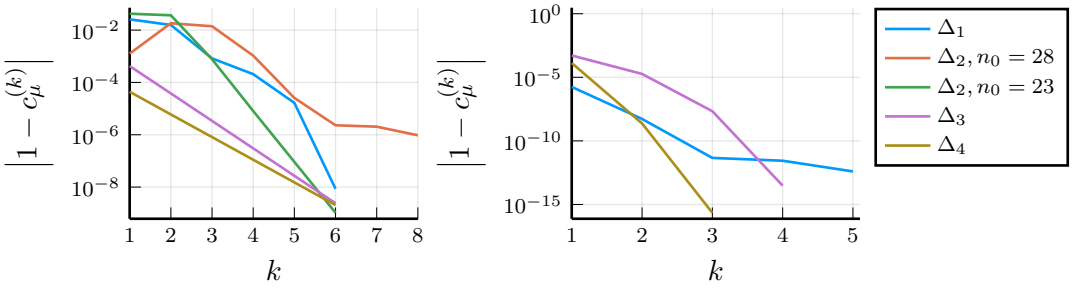
Table 4.2.: Mobility μ_A obtained via Algorithm 1

As in Section 3.4, we successively reduce n_0 by 1 until the algorithm succeeds. We note that one of the algorithms being successful or not for a certain n_0 does not imply that the other one will be successful or not.

Tables 4.1 and 4.2 summarize the results of Algorithm 1 for all four grids. Table 4.1 displays the largest n_0 for which the algorithm succeeded, the (final) size N of A for this n_0, the number of iteration steps (outer iteration steps and the total number of Newton steps), the value of $A_{1,1}$ before it is set to $-\varepsilon = -10^{-10}$, the final scaling factor c_μ (i.e. the product of all individual scaling factors $c_\mu^{(k)}$), the final value $\widetilde{r}_1$, and the quotient $\frac{\widetilde{r}_1}{r_1}$. For Δ_2 the tables display two results for different values n_0: For $n_0 > 23$ we never

		without setting $A_{1,1} = -\varepsilon$		after setting $A_{1,1} = -\varepsilon$	
Grid	n_0	μ_A	$\left\lvert\frac{\mu-\mu_A}{\mu}\right\rvert$	μ_A	$\left\lvert\frac{\mu-\mu_A}{\mu}\right\rvert$
Δ_1	39	0.335 993	$2.52 \cdot 10^{-2}$	0.336 275	$2.61 \cdot 10^{-2}$
Δ_2	28	(0.328 055)	$1.02 \cdot 10^{-3}$	(0.328 069)	$1.07 \cdot 10^{-3}$
Δ_2	23	0.315 226	$3.81 \cdot 10^{-2}$	0.315 257	$3.80 \cdot 10^{-2}$
Δ_3	20	0.327 594	$3.84 \cdot 10^{-4}$	0.327 636	$2.57 \cdot 10^{-4}$
Δ_4	10	(0.327 732)	$3.65 \cdot 10^{-5}$	0.327 774	$1.65 \cdot 10^{-4}$

Table 4.3.: Mobility μ_A of the Ornstein-Uhlenbeck systems obtained using the algorithm from Chapter 3



Algorithm 1 Algorithm 2

Figure 4.1.: Scaling factors $c_\mu^{(k)}$ in each step k of Algorithms 1 and 2 for each grid for which the iteration converged as in Tables 4.1 and 4.4

achieved $\left|\frac{\mu-\mu_A}{\mu}\right| \leq \delta_\mu = 10^{-8}$ but for $n_0 = 28$ we achieved $\left|\frac{\mu-\mu_A}{\mu}\right| < 10^{-6}$, which seems to be small enough. Table 4.2 displays the (approximated) mobility of the obtained Ornstein-Uhlenbeck system before and after setting $A_{1,1}$ to $-\varepsilon$ as well as the relative error $\left|\frac{\mu-\mu_A}{\mu}\right|$ for both cases. For comparison, Table 4.3 displays the same values for the systems obtained by applying the simple Lanczos method from Chapter 3 (without any modifications for obtaining the right mobility). For all values μ_A we used the same n_0 in all three tables, which is the largest n_0 for which the resulting VACF after setting $A_{1,1}$ in Algorithm 1 (i.e. the case in the right-most columns of Table 4.2) to $-\varepsilon$ was of positive type. As a result the VACFs obtained in the other cases were sometimes not of positive type, which is indicated by parentheses around the corresponding value μ_A in the tables.

We see that the algorithm yields an Ornstein-Uhlenbeck system whose mobility is very close to the desired one. However, the results of Δ_1 and Δ_2 in Table 4.1 show that enforcing $A_{1,1} \approx 0$ by setting $A_{1,1}$ to $-\varepsilon$ can have a significant impact on the mobility of the obtained Ornstein-Uhlenbeck system. This makes sense if we recall that modifying $A_{1,1}$ visually "shifts" the complete velocity auto-covariance function up or down, as we observed in Section 3.4.2 (Figures 3.4 to 3.7 especially). Note that the Newton iteration converged in these cases but the removal of spurious eigenvalues significantly modified $A_{1,1}$ afterwards. Using Δ_3 and Δ_4, no spurious eigenvalues occurred, consequently setting $A_{1,1} = -\varepsilon$ hardly changed $A_{1,1}$. However, we already know that spurious eigenvalues usually cannot be avoided.

For both algorithms Figure 4.1 displays the scaling factors $c_\mu^{(k)}$ for each grid and each iteration step k. $c_\mu^{(k)}$ seems to converge linearly towards 1 for Algorithm 1. Clearly Algorithm 2 clearly requires fewer iteration steps. For Δ_2 and $n_0 = 28$ we stopped Algorithm 1 after eight iterations since $\left|\frac{\mu-\mu_A}{\mu}\right|$ did not get much smaller for larger k than those displayed.

Tables 4.4 to 4.6 summarize the results of Algorithm 2, similarly to Tables 4.1 to 4.3. Again values $\widetilde{\mu}_A$ in parentheses indicate that the corresponding matrix A does not generate a VACF of positive type for this n_0. With Δ_2 this algorithm yielded no

Grid	n_0	N	Newton steps	$A_{1,1}$	$c_\mu - 1$	$\widetilde{r}_1$	$\frac{\widetilde{r}_1}{r_1} - 1$
Δ_1	39	14	5	$-1.63 \cdot 10^{-3}$	$1.75 \cdot 10^{-6}$	0.9633	$-2.89 \cdot 10^{-12}$
Δ_3	20	10	4	$4.93 \cdot 10^{-2}$	$5.01 \cdot 10^{-4}$	0.8659	$-1.02 \cdot 10^{-13}$
Δ_4	10	6	3	$3.54 \cdot 10^{-3}$	$1.31 \cdot 10^{-4}$	0.6145	$-3.33 \cdot 10^{-16}$

Table 4.4.: Summarized results of Algorithm 2

		without setting $A_{1,1} = -\varepsilon$		after setting $A_{1,1} = -\varepsilon$	
Grid	$\widetilde{\mu}$	$\widetilde{\mu}_A$	$\left\lvert\frac{\tilde{\mu}-\tilde{\mu}_A}{\tilde{\mu}}\right\rvert$	$\widetilde{\mu}_A$	$\left\lvert\frac{\tilde{\mu}-\tilde{\mu}_A}{\tilde{\mu}}\right\rvert$
Δ_1	0.254 578	0.254 970	$1.54 \cdot 10^{-3}$	0.255 073	$1.94 \cdot 10^{-3}$
Δ_3	0.255 628	(0.255 541)	$3.40 \cdot 10^{-4}$	0.252 743	$1.13 \cdot 10^{-2}$
Δ_4	0.253 703	(0.253 688)	$6.19 \cdot 10^{-5}$	0.253 492	$8.32 \cdot 10^{-4}$

Table 4.5.: Mobility $\widetilde{\mu}_A$ obtained via Algorithm 2

VACF of positive type for $n = 30, \ldots, 20$. For the other grids Table 4.4 displays the largest n_0 for which the algorithm succeeded, the size N of the final matrix A, the number of Newton steps performed, the value of $A_{1,1}$ before it is set to $-\varepsilon$, c_μ, and $\widetilde{r}_1$. Table 4.5 displays the approximation $\widetilde{\mu}_A$ for each grid along with the relative error of the obtained approximation without and with setting $A_{1,1}$ to $-\varepsilon$ while Table 4.6 displays the same values corresponding to the Ornstein-Uhlenbeck systems obtained via the Lanczos algorithm from Chapter 3. Note that $\widetilde{\mu}$ differs significantly from the estimated μ from above. While the algorithm indeed requires fewer steps than Algorithm 1, the relative error is not smaller than the one from Section 4.3.1 even without setting $A_{1,1}$ to ε. Therefore the algorithm does not offer any advantages in this case.

4.3.2. Noisy Data

In order to study the impact of noise, we use the data from Section 3.4.2. Note that here it is advisable to approximate the integral

$$\mu = \beta m \int_0^\infty C_V(t)\,\mathrm{d}t$$

only up to some time T (instead of ∞) in Algorithm 1 since for $t > T$ the impact of noise on C_V is too large. We choose $T = 5$ and use the time grid size $\widetilde{\tau} = 0.001$ for the quadrature. We will estimate the mobility for each noise level independently as we consider them individually.

We use the same data and the same time grids as in Section 3.4:

- Δ_1: $\tau = 0.06$, $n_0 = 50$
- Δ_2: $\tau = 0.1$, $n_0 = 30$

		without setting $A_{1,1} = -\varepsilon$		after setting $A_{1,1} = -\varepsilon$	
Grid	$\widetilde{\mu}$	$\widetilde{\mu}_A$	$\left\lvert\frac{\tilde{\mu}-\tilde{\mu}_A}{\tilde{\mu}}\right\rvert$	$\widetilde{\mu}_A$	$\left\lvert\frac{\tilde{\mu}-\tilde{\mu}_A}{\tilde{\mu}}\right\rvert$
Δ_1	0.254 578	0.255 091	$2.01 \cdot 10^{-3}$	0.255 091	$2.01 \cdot 10^{-3}$
Δ_3	0.255 628	0.255 524	$4.09 \cdot 10^{-4}$	0.255 524	$4.09 \cdot 10^{-4}$
Δ_4	0.253 703	(0.253 682)	$8.30 \cdot 10^{-5}$	(0.253 682)	$8.30 \cdot 10^{-5}$

Table 4.6.: Mobility $\widetilde{\mu}_A$ of the Ornstein-Uhlenbeck systems obtained using the algorithm from Chapter 3

Noise setting	Grid	n_0	N	Outer steps	Newton steps	$A_{1,1}$	c_μ	$\widetilde{r}_1$	$\frac{\widetilde{r}_1}{r_1}$
Low	Δ_1	47	15	5	18	$-1.60 \cdot 10^{-2}$	1.0207	0.8478	0.9527
	Δ_2	30	6	7	35	$-2.76 \cdot 10^{-2}$	0.9897	0.7356	0.9796
	Δ_3	14	5	5	13	$1.68 \cdot 10^{-2}$	0.9848	0.4924	1.0694
Medium	Δ_1	49	17	13	22	$2.81 \cdot 10^{-2}$	1.0128	0.8928	1.0042
	Δ_2	28	9	7	15	$-9.75 \cdot 10^{-3}$	1.0181	0.7424	0.9912
High	Δ_1	43	13	8	15	$-3.49 \cdot 10^{-2}$	1.0080	0.8867	0.9970
	Δ_2	29	9	8	15	$6.19 \cdot 10^{-2}$	0.9976	0.7338	0.9795
	Δ_3	15	7	10	29	$1.17 \cdot 10^{-1}$	1.0215	0.5307	1.1677

Table 4.7.: Results of Algorithm 1 for the different noise levels

- Δ_3: $\tau = 0.2$, $n_0 = 15$

We now compute Ornstein-Uhlenbeck systems using Algorithm 1. Table 4.7 summarizes the main results, displaying the same information as Table 4.1 for these data. It displays the largest number n_0 of grid points for which the algorithm succeeded. (Again we successively reduced n_0 by 1 until the algorithm succeeded.) For this n_0 it shows the size N of the final matrix A, the number of outer iteration steps and total number of Newton steps, the value of $A_{1,1}$ before it is set to $-\varepsilon$, the final scaling factor, the final value $\widetilde{r}_1$, and the quotient $\frac{\widetilde{r}_1}{r_1}$. Table 4.8 displays the mobility of the obtained approximating Ornstein-Uhlenbeck system if we set $A_{1,1}$ to $-\varepsilon = -10^{-10}$ and if we omit doing so along with the relative errors, similarly to Table 4.2. Values in parentheses again indicate that the corresponding matrix A does not produce a valid velocity auto-covariance function before setting $A_{1,1}$ to $-\varepsilon$. Table 4.9 displays the mobility of the Ornstein-Uhlenbeck systems obtained using the original algorithm from Chapter 3, analogously to Table 4.3.

The results in Tables 4.7 and 4.8 confirm our observations from Section 4.3.1: Algorithm 1 often allows to approximate the mobility very well even with noisy data although trying several values n_0 may be necessary. Setting $A_{1,1} = -\varepsilon$, however, has a significant impact on the mobility if $A_{1,1} \not\approx 0$. Compared to Section 4.3.1, the increased number of spurious eigenvalues especially with higher noise now leads to far worse results, which are no longer useful at all for approximating μ; indeed comparing Tables 4.8 and 4.9 shows that

Noise setting	μ	Grid	without setting $A_{1,1} = -\varepsilon$		after setting $A_{1,1} = -\varepsilon$	
			μ_A	$\left\lvert\frac{\mu-\mu_A}{\mu}\right\rvert$	μ_A	$\left\lvert\frac{\mu-\mu_A}{\mu}\right\rvert$
Low	0.606 699	Δ_1	0.606 699	$1.74 \cdot 10^{-9}$	0.611 801	$8.41 \cdot 10^{-3}$
		Δ_2	0.606 699	$1.20 \cdot 10^{-8}$	0.615 376	$1.43 \cdot 10^{-2}$
		Δ_3	(0.606 699)	$7.76 \cdot 10^{-10}$	0.601 460	$8.64 \cdot 10^{-3}$
Medium	0.563 293	Δ_1	(0.563 293)	$1.01 \cdot 10^{-9}$	0.555 536	$1.38 \cdot 10^{-2}$
		Δ_2	0.563 241	$9.33 \cdot 10^{-5}$	0.565 975	$4.76 \cdot 10^{-3}$
High	0.595 552	Δ_1	0.595 552	$3.47 \cdot 10^{-10}$	0.606 247	$1.80 \cdot 10^{-2}$
		Δ_2	(0.595 552)	$2.91 \cdot 10^{-10}$	0.577 450	$3.04 \cdot 10^{-2}$
		Δ_3	(0.595 552)	$7.77 \cdot 10^{-10}$	0.560 856	$5.83 \cdot 10^{-2}$

Table 4.8.: Mobility obtained via Algorithm 1

Noise setting	μ	Grid	without setting $A_{1,1} = -\varepsilon$		after setting $A_{1,1} = -\varepsilon$	
			μ_A	$\left\lvert\frac{\mu-\mu_A}{\mu}\right\rvert$	μ_A	$\left\lvert\frac{\mu-\mu_A}{\mu}\right\rvert$
Low	0.606 699	Δ_1	0.608 374	$2.76 \cdot 10^{-3}$	0.646 555	$6.57 \cdot 10^{-2}$
		Δ_2	0.612 075	$8.86 \cdot 10^{-3}$	0.621 151	$2.38 \cdot 10^{-2}$
		Δ_3	(0.614 495)	$1.28 \cdot 10^{-2}$	0.609 288	$4.27 \cdot 10^{-3}$
Medium	0.563 293	Δ_1	0.555 783	$1.33 \cdot 10^{-2}$	0.555 891	$1.31 \cdot 10^{-2}$
		Δ_2	0.553 907	$1.67 \cdot 10^{-2}$	0.557 022	$1.11 \cdot 10^{-2}$
High	0.595 552	Δ_1	0.572 775	$3.82 \cdot 10^{-2}$	0.589 725	$9.78 \cdot 10^{-3}$
		Δ_2	(0.596 660)	$1.86 \cdot 10^{-3}$	0.578 754	$2.82 \cdot 10^{-2}$
		Δ_3	(0.608 044)	$2.10 \cdot 10^{-2}$	0.580 530	$2.52 \cdot 10^{-2}$

Table 4.9.: Mobility of the Ornstein-Uhlenbeck systems obtained in Section 3.4.2

the obtained mobility is not closer to the "real" mobility than the one of the systems obtained in Section 3.4.2. Table 4.8 shows that the algorithm yields good results only when one omits setting $A_{1,1}$ to $-\varepsilon$ and is thus only applicable when $C_V'(0) = 0$ is not required. Even in this case, one sometimes needs more retries than in Section 3.4 in order for the algorithm to succeed, but it is possible to obtain Ornstein-Uhlenbeck systems which approximate the "real" mobility very well.

In all cases c_μ in Table 4.7 differs from 1 by less than 0.03, so the obtained VACF is still close to the original VACF.

Note that detecting a failure of Algorithm 1 is more complicated than in Chapter 3, where it was sufficient to check if the Newton iteration converged and if the obtained auto-covariance function was of positive type, i.e. if the Lur'e equations were solvable. Now we also have to check if the outer iteration converges. Even if the iteration converges

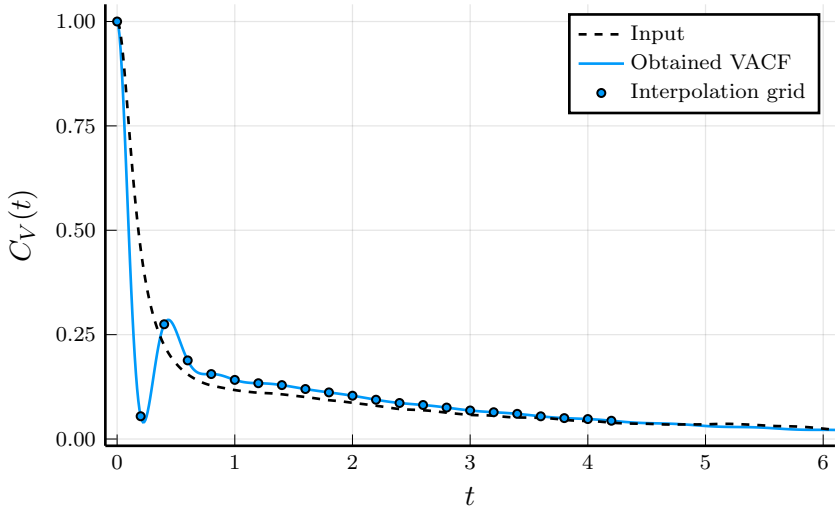
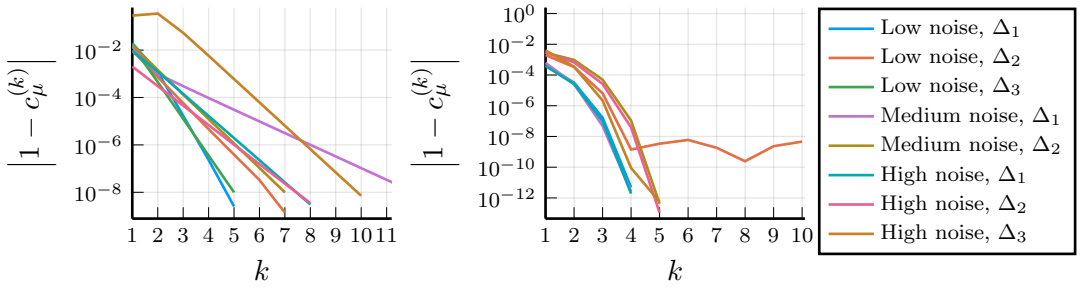


Figure 4.2.: Example of a setting where the Algorithm 1 converges and yields a VACF of positive type which is nevertheless a bad approximation



Algorithm 1 Algorithm 2

Figure 4.3.: Scaling factors $c_\mu^{(k)}$ in each step k of Algorithms 1 and 2 for each grid and noise setting (showing the settings for which the iteration converged as in Tables 4.7 and 4.10)

and yields a auto-covariance function of positive type, the final approximation may be bad if the final values of $\widetilde{r}_1$ and c_μ modify r too much. This happens quite rarely but Figure 4.2 displays the VACF of a different example obtained using the medium noise setting and the grid $\tau = 0.2$, $n_0 = 22$ where the resulting VACF, although of positive type, obviously does not approximate the given input well.

Figure 4.3 displays the factors $c_\mu^{(k)}$ for each iteration step for every grid, similarly to Figure 4.1. Again they seem to converge linearly in the case of convergence for Algorithm 1, where higher noise seems to result in slower convergence in most cases. For Algorithm 2 the convergence seems to be quadratic as one would expect from the Newton iteration.

Tables 4.10 to 4.12 display the results of Algorithm 2, similarly to Tables 4.4 to 4.6. We performed at most 10 Newton iterations; this maximum was only reached in the low noise

Noise setting	Grid	n_0	N	Newton steps	$A_{1,1}$	c_μ	$\widetilde{r}_1$	$\frac{\widetilde{r}_1}{r_1}$
Low	Δ_1	50	14	4	−0.0922	1.0005	0.8866	0.9963
	Δ_2	30	6	10	0.1703	1.0023	0.7423	0.9886
	Δ_3	–	–	–	–	–	–	–
Medium	Δ_1	49	17	4	0.0695	1.0006	0.8853	0.9957
	Δ_2	30	11	5	0.3238	1.0038	0.7357	0.9821
High	Δ_1	50	17	4	−0.0277	1.0004	0.8865	0.9968
	Δ_2	29	9	5	0.2595	1.0036	0.7365	0.9831
	Δ_3	12	7	5	0.1826	1.0043	0.4490	0.9879

Table 4.10.: Summarized results of Algorithm 2

			without setting $A_{1,1} = -\varepsilon$		after setting $A_{1,1} = -\varepsilon$	
Noise setting	Grid	$\widetilde{\mu}$	$\widetilde{\mu}_A$	$\left\lvert\frac{\tilde{\mu}-\tilde{\mu}_A}{\tilde{\mu}}\right\rvert$	$\widetilde{\mu}_A$	$\left\lvert\frac{\tilde{\mu}-\tilde{\mu}_A}{\tilde{\mu}}\right\rvert$
Low	Δ_1	0.498 058	0.497 945	$2.28 \cdot 10^{-4}$	0.517 317	$3.87 \cdot 10^{-2}$
	Δ_2	0.496 771	(0.498 572)	$3.63 \cdot 10^{-3}$	0.466 055	$6.18 \cdot 10^{-2}$
Medium	Δ_1	0.470 270	(0.469 199)	$2.28 \cdot 10^{-3}$	0.456 697	$2.89 \cdot 10^{-2}$
	Δ_2	0.472 696	(0.471 580)	$2.36 \cdot 10^{-3}$	0.417 702	$1.16 \cdot 10^{-1}$
High	Δ_1	0.488 238	0.484 970	$6.69 \cdot 10^{-3}$	0.490 416	$4.46 \cdot 10^{-3}$
	Δ_2	0.480 130	(0.482 414)	$4.76 \cdot 10^{-3}$	0.437 333	$8.91 \cdot 10^{-2}$
	Δ_3	0.444 407	(0.443 308)	$2.47 \cdot 10^{-3}$	0.416 752	$6.22 \cdot 10^{-2}$

Table 4.11.: Mobility $\widetilde{\mu}_A$ obtained via Algorithm 2

setting with Δ_2. When using the low noise data with Δ_3, no value $n_0 = 15, \ldots, 8$ yielded satisfying results, i.e. when the iteration did not fail, $|A_{1,1}|$ was so large that setting $A_{1,1}$ to $-\varepsilon$ had a significant impact on the VACF. Compared to Table 4.7 the scaling coefficients in Table 4.10 are often much closer to 1 and again the algorithm requires fewer iteration steps. However, we see that just as with Algorithm 1, the mobility of the obtained Ornstein-Uhlenbeck system is no longer close to the "true" mobility if we enforce $A_{1,1} \approx 0$. Comparing Table 4.11 to Table 4.12, which displays the results when using the algorithm from Chapter 3 (using the n_0 from Table 4.10), we see that at least for these data Algorithm 2 offers no advantage over the base algorithm from Chapter 3.

The results from Section 4.3 show that while Algorithm 1 can yield good results, it is far from being a reliable method for obtaining a good Ornstein-Uhlenbeck representation of a generalized Langevin equation with external force. On top of that, it is only applicable if the obtained system does not need to satisfy $C_V'(t) = 0$, which depends on the physical background of the considered problem. Otherwise the mobility of the obtained Ornstein-Uhlenbeck systems is not closer to the real mobility than the one of the systems from

Noise setting	Grid	$\widetilde{\mu}$	without setting $A_{1,1} = -\varepsilon$		after setting $A_{1,1} = -\varepsilon$	
			$\widetilde{\mu}_A$	$\left\lvert\frac{\tilde{\mu}-\tilde{\mu}_A}{\tilde{\mu}}\right\rvert$	$\widetilde{\mu}_A$	$\left\lvert\frac{\tilde{\mu}-\tilde{\mu}_A}{\tilde{\mu}}\right\rvert$
Low	Δ_1	0.498 058	0.497 839	$4.40 \cdot 10^{-4}$	0.497 839	$4.40 \cdot 10^{-4}$
	Δ_2	0.496 771	0.498 381	$3.24 \cdot 10^{-3}$	0.498 381	$3.24 \cdot 10^{-3}$
	Δ_3	0.482 056	(0.489 737)	$1.59 \cdot 10^{-2}$	0.489 737	$1.59 \cdot 10^{-2}$
Medium	Δ_1	0.470 270	0.469 198	$2.28 \cdot 10^{-3}$	0.469 198	$2.28 \cdot 10^{-3}$
	Δ_2	0.472 696	0.471 397	$2.75 \cdot 10^{-3}$	0.471 397	$2.75 \cdot 10^{-3}$
High	Δ_1	0.488 238	0.485 282	$6.05 \cdot 10^{-3}$	0.485 282	$6.05 \cdot 10^{-3}$
	Δ_2	0.480 130	(0.482 070)	$4.04 \cdot 10^{-3}$	0.482 070	$4.04 \cdot 10^{-3}$
	Δ_3	0.444 407	(0.443 048)	$3.06 \cdot 10^{-3}$	0.443 048	$3.06 \cdot 10^{-3}$

Table 4.12.: Mobility $\widetilde{\mu}_A$ corresponding to the Ornstein-Uhlenbeck systems obtained in Section 3.4.2

Chapter 3. The mobility of the Ornstein-Uhlenbeck systems obtained by Algorithm 2 was not closer to the "real" mobility (i.e. the estimate we used) than the one from Section 3.4 and we thus see no advantage in this algorithm.

5. The Multidimensional Lanczos Algorithm

We now want to extend the algorithm from Chapter 3 to the multidimensional generalized Langevin equation with a matrix-valued velocity auto-covariance function. The key part of our approach will again be the interpolation of the given auto-covariance function in a given time grid $0, \tau, \ldots, (n_0 - 1)\tau$ by a function

$$C_V^A(t) := P^T e^{tA} \Sigma P = \frac{1}{\beta m} P^T e^{tA} P \quad (t \geq 0), \qquad P = \begin{bmatrix} \mathbf{I}_d \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(d|N-d)\times d},$$

where

$$A \in \mathbb{R}^{N\times N} \quad \text{and} \quad \Sigma = \frac{1}{\beta m} \begin{bmatrix} \mathbf{I}_d & \mathbf{0} \\ \mathbf{0} & S \end{bmatrix} \in \mathbb{R}^{(d|N-d)\times(d|N-d)}$$

have to be determined. If A is diagonalizable with eigenvalues $\lambda_1, \ldots, \lambda_N$ and eigenvectors $u_1, \ldots, u_N$, then

$$C_V^A(t) = \frac{1}{\beta m} P^T U e^{t\Lambda_A} U^{-1} P \quad \text{with} \quad \Lambda_A = \operatorname{diag}(\lambda_1, \ldots, \lambda_N), \quad U = [u_1, \ldots, u_N]$$

and thus

$$C_V^A(t) = \frac{1}{\beta m} \sum_{k=1}^{N} a_k b_k^* e^{\lambda_k t} \in \mathbb{R}^{d\times d}$$

where $a_k = P^* u_k$, $b_k = P^*(U^{-*})_{\cdot,k}$. (The eigenvalues do not need to be pairwise different; for multiple eigenvalues the corresponding rank 1 coefficient matrices $a_k b_k^*$ sum up to matrices of higher rank as described in Section 2.2.1 for the memory kernel.)

In order to interpolate a multidimensional (matrix-valued) auto-covariance function by such a Prony series, we first have to generalize the Lanczos algorithm from Section 3.1 to the matrix-valued problem. The most notable change is that the bilinear form $[\cdot, \cdot]_r$ introduced in Section 3.1 is (in general) non-symmetric in the multidimensional case. Therefore we will obtain two sequences of biorthogonal polynomials instead of a single sequence.

To extend the algorithm, we will first ignore our original goal and our application to the generalized Langevin equation for this chapter and introduce the Lanczos algorithm for solving the *realization problem*. Most of the presentation of Sections 5.1 and 5.2 follows [14]. In Chapter 6 we will show how to apply this method to extend the algorithm from Chapter 3 to the multidimensional generalized Langevin equation, including the generalization of the additional steps from Section 3.2.

5.1. Introduction

The algorithm from [14] constructs from a given sequence $H_0, \dots, H_{n_0-1}$ of so-called *moments* $H_k \in \mathbb{R}^{d\times d}$ two sequences $\psi = (\psi_k)_{k=0,\dots,N-1}$, $\phi = (\phi_k)_{k=0,\dots,N-1}$ ($N \in \mathbb{N}\cup\{\infty\}$ will be specified later) of polynomials $\psi_k(x) := \sum_{i=0}^{k} a_{k,i}x^i$ and $\phi_k(x) := \sum_{i=0}^{k} b_{k,i}x^i$. Here $a_{k,i}$ and $b_{k,i}$ are the (scalar) coefficients of the polynomials ψ_k and ϕ_k with $a_{k,i} = b_{k,i} := 0$ for $i > k$. We denote by

$$a_k := \begin{bmatrix} a_{k,0} \\ a_{k,1} \\ \vdots \end{bmatrix} \quad \text{and} \quad b_k := \begin{bmatrix} b_{k,0} \\ b_{k,1} \\ \vdots \end{bmatrix}$$

the infinitely large coefficient vectors of ψ_k and ϕ_k (which both contain only a finite number of non-zero entries). The polynomial sequences ψ and ϕ are constructed with increasing degrees $\deg\psi_k = \deg\phi_k = k$.

A central property of the polynomials $(\psi_k)_{k\in\mathbb{N}_0}$ and $(\phi_k)_{k\in\mathbb{N}_0}$ is their biorthogonality,

$$[\psi_k, \phi_l]_H \neq 0 \quad \text{if and only if} \quad k = l,$$

w.r.t. the (not necessarily symmetric or positive definite) bilinear form

$$[\psi_k, \phi_l]_H := b_k^T \mathcal{H} a_l \tag{5.1}$$

with the infinitely large (see below) block Hankel matrix

$$\mathcal{H} := \begin{bmatrix} H_0 & H_1 & H_2 & \dots \\ H_1 & H_2 & H_3 & \dots \\ H_2 & H_3 & H_4 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{5.2}$$

In the following, we will always assume $[\psi_k, \phi_k]_H \neq 0$ for all $k < N$. Otherwise there are no polynomials $\psi_{k'}$ and $\phi_{k'}$ strictly satisfying the biorthogonality conditions for $k' \geq k$. Instead, one can only achieve "blockwise" biorthogonality by combining successive polynomials into blocks. We will mention this case briefly in Section 5.2.1 but will not treat it here in detail.

The moments sequence $H = (H_k)_{k=0,\dots,n_0-1}$ can be finitely or infinitely large, i.e. $n_0 = \infty$ is allowed. We will focus on the finite case as only this case is relevant in applications with experimentally determined or simulated data. In this case, only some entries of $\mathcal{H}$ are defined by H. We still consider $\mathcal{H}$ as a matrix with infinitely many rows and columns but consider all entries not given by H as "undefined": While in principle we could set them arbitrarily by extending the moments sequence H accordingly, we do not want to rely on their values and will thus stop our algorithm as soon as it requires any of these values. The result of the vector-matrix-vector product (5.1) is only defined if all undefined entries in $\mathcal{H}$ are multiplied by a zero entry in a or in b. Consequently $[\psi, \phi]_H$ is only defined for

polynomials with sufficiently small degree such that all undefined entries are multiplied with polynomial coefficients which are zero, more precisely if $\lfloor \frac{\deg \psi}{d} \rfloor + \lfloor \frac{\deg \phi}{d} \rfloor < n_0$.

In general, the algorithm as presented in [14] does not require the matrices H_k to be square. However, we restrict ourselves to the case of square matrices as we are only interested in this case.

5.2. Summary of the Algorithm

As mentioned in Section 5.1, the algorithm from [14] requires as input the moments sequence H and constructs two sequences ψ and ϕ of biorthogonal polynomials with increasing degrees. These sequences are maximal in the sense that they contain a polynomial of each degree up to degree $\lceil \frac{n_0}{2} \rceil \cdot d - 1$, see below.

The algorithm works similarly to the Gram-Schmidt algorithm. However, $[\cdot, \cdot]_H$ not being symmetric, it generates two sequences of polynomials instead of a single one. The algorithm starts with two empty sequences ψ and ϕ, which it then expands successively as follows: Assume that we have already performed k steps ($k \geq 0$) of the algorithm. In these steps the k first polynomials $\psi_0, \ldots, \psi_{k-1}$ as well as the k first polynomials $\phi_0, \ldots, \phi_{k-1}$ with degrees $0, 1, \ldots, k-1$ have already been determined. (For convenience, we start numbering the steps with 0.) In step k we add a new polynomial to ψ via orthogonalizing a base polynomial $\hat{\psi}_k$ against $\phi_0, \ldots, \phi_{k-1}$. As base polynomial we choose $\hat{\psi}_k(x) := x^k$ for $k < d$ and $\hat{\psi}_k(x) := x^d \psi_{k-d}(x)$ otherwise. We can also scale the new polynomial by an arbitrary scalar $t_{k-d,k} \neq 0$. Therefore the next polynomial ψ_k is given by

$$\psi_k := \frac{1}{t_{k-d,k}} \left(\hat{\psi}_k - \sum_{j=0}^{k-1} \frac{[\hat{\psi}_k, \phi_j]_H}{[\psi_j, \phi_j]_H} \psi_j \right). \tag{5.3}$$

Analogously a new polynomial ϕ_k is computed via

$$\phi_k := \frac{1}{u_{k-d,k}} \left(\hat{\phi}_k - \sum_{j=0}^{k-1} \frac{[\psi_j, \hat{\phi}_k]_H}{[\psi_j, \phi_j]_H} \phi_j \right) \tag{5.4}$$

with $\hat{\phi}_k(x) := x^k$ for $k < d$ and $\hat{\phi}_k(x) := x^d \phi_{k-d}(x)$ otherwise and an arbitrary scaling factor $u_{k-d,k} \neq 0$. Recall that we assume $[\psi_j, \phi_j]_H \neq 0$; we briefly describe what happens otherwise in Section 5.2.1. After ψ_k and ϕ_k have been computed and appended to ψ and ϕ, the algorithm continues with the next step.

In the case of a finite moments sequence, the iteration stops when a required value of the bilinear form $[\cdot, \cdot]_H$ is not defined due to the polynomials' degrees being too large. (5.3) and (5.4) show that only entries $\mathcal{H}_{i,j}$ with $i \leq k$ and $j \leq k-1$ are required for computing ψ_k and only entries $\mathcal{H}_{i,j}$ with $i \leq k-1$ and $j \leq k$ are required for computing ϕ_k. These are defined for $k = 0, \ldots, \lceil \frac{n_0}{2} \rceil \cdot d - 1$. If n_0 is even, they are also defined for $k = \lceil \frac{n_0}{2} \rceil \cdot d$ but this step in the algorithm would lead to one of the two special cases mentioned in Section 5.2.1 if neither of them has occurred yet, and is thus not performed. In the case of an infinite moments matrix, the algorithm could continue infinitely (except

in a special case, see Section 5.2.1). For the remaining chapter let $N := \lceil \frac{n_0}{2} \rceil \cdot d$ be the number of steps which are performed.

Let

$$t_{i,j} := \frac{[\hat{\psi}_{i+d}, \phi_j]_H}{[\psi_j, \phi_j]_H} \quad \text{for} \quad -d \leq i < N-d, \quad 0 \leq j < N, \quad i \neq j-d. \tag{5.5}$$

We use the coefficients $t_{i,j}$ with $i \geq 0$ as entries of a matrix J. J contains in each row the coefficients $t_{i,j}$ corresponding to the polynomial ψ_i (where the rows of the first d polynomials are discarded). After N steps we have computed all $t_{i,j}$ with $-d \leq i < N-d$ and $0 \leq j < N$. For our application in Section 5.4 we require J to be a quadratic matrix. Hence we also compute $t_{i,j}$ for $N-d \leq i < N$ and $0 \leq j < N$ via (5.5). (Note that determining the corresponding polynomials $\psi_N, \ldots, \psi_{N+d-1}$ is not possible since this would require the values $t_{i,j}$ with $N-d \leq i < N$ and $j \geq N$, which we cannot compute with the defined entries of $\mathcal{H}$. However, the required values $t_{i,j}$ can be computed without determining these polynomials.) After doing so we have an $N \times N$ matrix $J = [t_{i,j}]_{0 \leq i < N, 0 \leq j < N}$.

Analogously to Section 3.1, reformulating (5.3) with the coefficients $t_{i,j}$ from (5.5) and the arbitrarily chosen $t_{j-d,j}$ from (5.3) yields

$$\hat{\psi}_k = \sum_{j=0}^{k} t_{k-d,j} \psi_j, \qquad k = 0, \ldots, N-1.$$

(This implies that $t_{i,j} = \frac{[\hat{\psi}_{i+d}, \phi_j]_H}{[\psi_j, \phi_j]_H}$ also holds for $i = j-d$.) Collecting this identity for all $k = d, \ldots, d+N-1$ and rewriting as a matrix-vector product using the coefficient vectors a_i of ψ_i yields

$$\begin{bmatrix} \mathbf{0}_d & \mathbf{0}_d & \ldots & \mathbf{0}_d \\ a_0 & a_1 & \ldots & a_{N-1} \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & \ldots & a_{N-1} \end{bmatrix} J^T + \begin{bmatrix} \mathbf{0} & \ldots & \mathbf{0} & \xi_1 & \ldots & \xi_d \end{bmatrix}. \tag{5.6}$$

Here the left-hand side contains the coefficients of $x^d\psi_0, \ldots, x^d\psi_{N-1}$. Similarly to Section 3.1, $\xi_1, \ldots, \xi_d$ represent the contributions of all polynomials ψ_j with $j > N-1$, which are not represented by the matrix product. (In other words, they represent the summands which would be added in the right-hand matrix product if the matrix $[a_0, a_1, \ldots, a_{N-1}]$ were extended by further columns and J^T by further rows.)

For $j \geq 0$, (5.5) and the block Hankel structure of $\mathcal{H}$ imply

$$[\psi_j, \phi_j]_H t_{i,j} = [x^d\psi_i, \phi_j]_H = [\psi_i, x^d\phi_j]_H. \tag{5.7}$$

If $\deg(x^d\phi_j) < \deg \psi_i$, i.e. $j < i-d$, then $[\psi_i, x^d\phi_j]_H = 0$ as ψ_i is orthogonal to all polynomials whose degree is smaller than i, and thus $t_{i,j} = 0$. In addition $t_{i,j} = 0$ for all $j > i+d$ since then $\deg \phi_j > \hat{\psi}_{i+d}$ and thus ϕ_j is orthogonal to $\hat{\psi}_{i+d}$. Consequently J is a banded matrix with bandwidth d. When implementing the algorithm, this knowledge can of course be used to not explicitly compute matrix entries which are known to be 0.

Algorithm 3 Lanczos method for computing formally orthogonal polynomials

Input: Moments $H = (H_0, \dots, H_{n_0-1})$

Set $N \leftarrow \lceil \frac{n_0}{2} \rceil d$.
for $k = 0, 1, \dots, N-1$ **do**
  if $k < d$ **then**
    Set $\hat{\psi}_k(x) \leftarrow x^k, \quad \hat{\phi}_k(x) \leftarrow x^k$.
  else
    Set $\hat{\psi}_k(x) \leftarrow x^d \psi_{k-d}(x), \quad \hat{\phi}_k(x) \leftarrow x^d \phi_{k-d}(x)$.
  end if
  for $j = \max\{0, k-d\}, \dots, k-1$ **do**
    Set $t_{k-d,i} \leftarrow \frac{[\hat{\psi}_k, \phi_j]_H}{[\psi_j, \phi_j]_H}$.
    Set $u_{k-d,i} \leftarrow \frac{[\psi_j, \hat{\phi}_k]_H}{[\psi_j, \phi_j]_H}$.
  end for
  Choose $t_{k-d,k} \neq 0$ and set $\psi_k \leftarrow \dfrac{1}{t_{k-d,k}} \left(\hat{\psi}_k - \displaystyle\sum_{j=\max\{0,k-2d\}}^{k-1} t_{k-d,j} \psi_j \right)$.
  Choose $u_{k-d,k} \neq 0$ and set $\phi_k \leftarrow \dfrac{1}{u_{k-d,k}} \left(\hat{\phi}_k - \displaystyle\sum_{j=\max\{0,k-2d\}}^{k-1} u_{k-d,j} \phi_j \right)$.
end for
for $k = N, \dots, N+d-1$ **do**
  for $j = \max\{0, k-d\}, \dots, \min\{k-1, N-1\}$ **do**
    Set $t_{k-d,j} \leftarrow \frac{[x^d \psi_{k-d}, \phi_j]_H}{[\psi_j, \phi_j]_H}$.
  end for
end for
Set $J \leftarrow [t_{i,j}]_{i,j=0,\dots,N-1}$ (with $t_{i,j} = 0$ if it has not been explicitly computed).
return J

Analogously we can define

$$u_{i,j} := \frac{[\psi_j, \hat{\phi}_{i+d}]_H}{[\psi_j, \phi_j]_H} \quad \text{for} \quad -d \le i < N-d, \quad 0 \le j < N, \quad i \neq j-d.$$

and combine the coefficients $u_{i,j}$ with $i \ge d$ in a banded matrix, which, however, is of no importance to us.

In our application, to be discussed in Chapter 6, H_0 is always symmetric and positive definite. We will therefore assume throughout this chapter and the following one that H_0 is non-singular. (Indeed H_0 is always non-singular if none of the special cases mentioned in Section 5.2.1 occurs.)

Algorithm 3 summarizes the complete algorithm. In our application in Chapter 6, we choose

$$t_{k-d,k} = u_{k-d,k} := \left| \left[\hat{\psi}_k - \sum_{j=\max\{0,k-d\}}^{k-1} u_{k-d,j} \psi_j, \hat{\phi}_k - \sum_{j=\max\{0,k-d\}}^{k-1} t_{k-d,j} \phi_j \right]_H \right|^{1/2} \tag{5.8}$$

to obtain normalized polynomials ψ_k and ϕ_k in the sense that $\left|[\psi_k, \phi_k]_H\right| = 1$.

5.2.1. A Note on Special Cases

We can encounter two special situations when applying the algorithm from Section 5.2 to an arbitrary moments sequence. Since all moments sequences where either of these cases occurs form a zero set (when using exact arithmetic) for every fixed dimension d and sequence length n_0, it is extremely unlikely to encounter these cases with experimental data. Since we never encountered them in our numerical experiments, we will not treat them here but refer to [14] instead, which discusses both cases in detail.

The first special case occurs whenever a polynomial ψ_k is orthogonal to all polynomials $\hat{\phi}$ for which $[\psi_k, \hat{\phi}]_H$ is defined. This is equivalent to the kth row of $\mathcal{H}$ being linearly dependent on all previous rows of $\mathcal{H}$. In the case of a finite moments sequence and a partially defined matrix, this means that the defined part of the kth row depends linearly on the corresponding parts of the previous rows. (Note that one considers only the part of the row defined by H as extending the row by zeros or otherwise can lead to the extended row no longer being linearly dependent on the previous rows. This would implicitly extend the moments sequence, which we want to avoid.)

In this case the polynomial in question is discarded instead of being added to ψ and the next polynomial (whose degree is greater by 1 than the discarded one's) is taken instead. This implies that ψ will not contain a polynomial of degree k. This process is called *deflation*. A discarded polynomial $\widetilde{\psi}$ will not be used in the later steps of the algorithm. Therefore, since $x^d\widetilde{\psi}(x)$ will never be used as base polynomial in (5.3), ψ will contain no polynomials of degree $k + md$, $m \in \mathbb{N}$, as the block Hankel form of $\mathcal{H}$ implies that these polynomials would also be discarded due to deflation.

Analogously, deflation in ϕ occurs whenever a polynomial ϕ_k is orthogonal to all polynomials $\hat{\psi}$ for which $[\hat{\psi}, \phi_k]_H$ is defined. This is equivalent to the defined part of the kth column of $\mathcal{H}$ being linearly dependent on the corresponding parts of all previous columns. Deflation can occur independently in ψ and in ϕ. This implies that after deflation has occurred, ψ_k and ϕ_k no longer have to have the same degrees. When d deflations in ψ or d deflations in ϕ have occurred, the algorithm stops (even with an infinite moments sequence) since all following polynomials would be deflated due to having the form $x^{md}\widetilde{\psi}$, $m \in \mathbb{N}$, with a deflated polynomial $\widetilde{\psi}$ or $x^{md}\widetilde{\phi}$ with a deflated polynomial $\widetilde{\phi}$, respectively. This happens if and only if the infinite matrix $\mathcal{H}$ has only finite rank. Deflation does not occur in the one-dimensional case or, more precisely, in the case of deflation the one-dimensional algorithm immediately stops. (Stopping due to deflation does not mean that the algorithm failed. Instead for Theorem 5.5 below it means that a matrix J which is smaller than $N \times N$ is sufficient to have the realization property mentioned there.)

The second special case, called *look-ahead*, can also occur in the one-dimensional case. Look-ahead is necessary when in step k neither ψ_k nor ϕ_k is discarded due to deflation but $[\psi_k, \phi_k]_H = 0$ still holds. It is consequently impossible to orthogonalize the following polynomials against ψ_k and ϕ_k. Instead one continues the algorithm as normal but skips orthogonalizing the new polynomials against ψ_k and ϕ_k. After the next step (when one

has obtained ψ_{k+1} and ϕ_{k+1}) it is possible to orthogonalize all following polynomials against ψ_k and ψ_{k+1} (or ϕ_k and ϕ_{k+1}, respectively) simultaneously provided that

$$\det \begin{bmatrix} [\psi_k, \phi_k]_H & [\psi_{k+1}, \phi_k]_H \\ [\psi_k, \phi_{k+1}]_H & [\psi_{k+1}, \phi_{k+1}]_H \end{bmatrix} \neq 0.$$

(If the determinant is 0, one has to continue the algorithm by computing further polynomials ψ_i and ϕ_i until $\det [[\psi_i, \phi_j]_H]_{i,j=k,\dots,l} \neq 0$ after a certain step l.) However, $[\psi_{k+1}, \phi_k]_H = 0$ and $[\psi_k, \phi_{k+1}]_H = 0$ will in general not hold in this case. Therefore, when look-ahead is necessary, the algorithm will not yield two sequences of biorthogonal polynomials but will instead yield two "block-biorthogonal" sequences: Both ψ and ϕ will be partitioned into blocks such that polynomials ψ_i and ϕ_j are orthogonal if they are not in corresponding blocks. The partition into blocks is the same for ψ and ϕ, i.e. ψ_i and ψ_j are in the same block of ψ if and only if ϕ_i and ϕ_j are in the same block of ϕ. New blocks are started whenever no look-ahead occurs in a step of the algorithm.

Both special cases can be handled by the Lanczos algorithm, but this requires non-trivial modifications to the algorithm. T is no longer banded in these cases, but can have (single) entries outside its banded area. A more detailed description of the algorithm with deflation and look-ahead can be found in [14]. We will stick to the simple case and assume in this thesis that neither deflation nor look-ahead occurs although we tested for them in our numerical experiments. As mentioned above, this assumption is reasonable in exact arithmetic.

To detect deflation or a necessary look-ahead step, one checks if a certain value is zero. Due to numerical errors, one should not test for exact equality in this case but rather test whether the corresponding value is sufficiently small in absolute value. Handling this "inexact" deflation and look-ahead requires some further modifications to the algorithm, cf. [14]. It also implies that the set of moments sequences where deflation or look-ahead occurs is no longer a zero set. Still we never encountered one of these special cases in our numerical experiments.

5.3. The Realization Problem

Our goal is to find a matrix $J \in \mathbb{R}^{N \times N}$ for a given moments sequence $H = (H_k)_{k=0,\dots,n_0-1}$, $H_k \in \mathbb{R}^{d \times d}$, such that

$$H_k = L^T J^k R \quad \text{for all } k = 0, \dots, n_0 - 1 \tag{5.9}$$

where

$$L = L_N := \begin{bmatrix} L_0 \\ \mathbf{0}_{N-d,d} \end{bmatrix} \in \mathbb{R}^{N \times d} \quad \text{and} \quad R = R_N := \begin{bmatrix} R_0 \\ \mathbf{0}_{N-d,d} \end{bmatrix} \in \mathbb{R}^{N \times d} \tag{5.10}$$

with given L_0 and R_0 such that $L_0^T R_0 = H_0$. Here we first treat the more general case with arbitrary matrices $L, R \in \mathbb{R}^{N \times d}$, $N > d$, as it is treated in [48, Section 6.5].

Definition 5.1. *Let $n_0 \in \mathbb{N} \cup \{\infty\}$ and let $H = (H_k)_{k=0,\dots,n_0-1}$ be a matrix sequence. A triple (J, L, R) with $J \in \mathbb{R}^{N\times N}$ and $L, R \in \mathbb{R}^{N\times d}$, $N \in \mathbb{N}$, satisfying (5.9) is called a* realization *of H. N is the realization's* dimension. *A realization with minimal dimension is called* minimal. *A triple (J, L, R) is called* minimal *if it is a minimal realization of the sequence $(L^T J^k R)_{k\in\mathbb{N}_0}$.*

Remark 5.2. The one-dimensional algorithm from Section 3.1 computes a realization of the given auto-covariance data, see (3.3) and (3.4). This is essential for the interpolation property of that algorithm which we employ in Chapter 3 in order to interpolate the given VACF. In the following we will prove that the extended algorithm from Section 5.2 retains this realization property.

A proof of the following statement about the existence of a realization can be found [48, Theorem 28].

Theorem 5.3. *An infinite sequence $H = (H_k)_{k\in\mathbb{N}_0} \subseteq \mathbb{R}^{d\times d}$ has a realization if and only if the infinite matrix $\mathcal{H}$ from (5.2) has finite rank. Then* $\operatorname{rank}\mathcal{H}$ *is the minimum possible dimension of a realization of H.*

Remark 5.4. According to [48, Theorem 27] and [4, Theorem 3.4.2], the minimal realization is uniquely determined up to changes of basis: If (J, L, R) is a minimal realization, every other minimal realization $(\widetilde{J}, \widetilde{L}, \widetilde{R})$ has the form

$$\widetilde{J} = Q^{-1}JQ, \qquad \widetilde{L} = Q^T L, \qquad \widetilde{R} = Q^{-1}R$$

with a non-singular matrix $Q \in \mathbb{R}^{N\times N}$.

We can always extend a finite sequence $H = (H_k)_{k=0,\dots,n_0-1}$ arbitrarily to an infinite sequence $(H_k)_{k\in\mathbb{N}_0}$ by choosing H_k arbitrarily for $k \geq n_0$. Then a realization of H exists if and only if any infinite extension of H has a realization, and the minimum dimension of the finite sequence is the minimum possible rank of any matrix $\mathcal{H}$ corresponding to an infinite extension. Thus we define the rank of a partially defined block Hankel matrix $\mathcal{H}$ as the minimum possible rank of any block Hankel matrix $\mathcal{H}$ corresponding to an infinite continuation of H.

By extending a finite sequence H by zero matrices to an infinite sequence, one can always obtain a block Hankel matrix with rank at most $n_0 \cdot d$. Hence every finite sequence has a realization. Since the extension by zeros does not necessarily have the minimum possible rank, the dimension of the realization yielded by Algorithm 3 can be smaller: For example, for $d = 1$ and $H_0 = 1$, $H_1 = \frac{1}{2}$ the continuation by zeros yields a rank 2 Hankel matrix while the continuation $H_k = 2^{-k}$ results in a rank 1 Hankel matrix.

Since we require $H_0 = L^T R \in \mathbb{R}^{d\times d}$ to be non-singular, the $N \times d$ matrices L and R need to have rank d for every realization (J, L, R); in particular, $N \geq d$. We now split L and R into two blocks each:

$$L = \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} \in \mathbb{R}^{(d|N-d)\times d} \quad \text{and} \quad R = \begin{bmatrix} R_1 \\ R_2 \end{bmatrix} \in \mathbb{R}^{(d|N-d)\times d}.$$

Then, by replacing L by ΠL with a suitable permutation matrix Π, we can achieve that the upper part L_1 of L is non-singular. In view of Remark 5.4, we will omit Π in the next lines and assume instead that L_1 is already non-singular. Another change of basis, using the change-of-basis matrix

$$K := \begin{bmatrix} L_1^{-T}L_0^T - L_1^{-T}L_2^T R_2 R_0^{-1} & -L_1^{-T}L_2^T \\ R_2 R_0^{-1} & \mathbf{I}_{N-d} \end{bmatrix} \in \mathbb{R}^{(d|N-d)\times(d|N-d)},$$
$$K^{-1} = \begin{bmatrix} L_0^{-T}L_1^T & L_0^{-T}L_2^T \\ -R_2 H_0^{-1}L_1^T & -R_2 H_0^{-1}L_2^T + \mathbf{I}_{N-d} \end{bmatrix} \in \mathbb{R}^{(d|N-d)\times(d|N-d)}$$

yields (exploiting $H_0 = L^T R = L_1^T R_1 + L_2^T R_2$) another realization $(\widetilde{J}, \widetilde{L}, \widetilde{R})$ with

$$\widetilde{L} = K^T L = \begin{bmatrix} L_0 \\ \mathbf{0}_{N-d,d} \end{bmatrix}, \qquad \widetilde{R} = K^{-1} R = \begin{bmatrix} R_0 \\ \mathbf{0}_{N-d,d} \end{bmatrix}, \qquad \widetilde{J} = K^{-1} J K.$$

Consequently the restriction of L and R to the form (5.10) does not restrict the existence of a realization if H_0 is non-singular.

5.4. The Realization Property of the Matrix J

The algorithm from Section 5.2 enables us to compute a realization of a sequence H. Since H_0 will always be positive definite in our application, we choose $L_0 = R_0 := \hat{L}$ in (5.10), where $H_0 = \hat{L}\hat{L}^T$ is the Cholesky decomposition of H_0. For simplicity we restrict ourselves to the situation $H_0 = \mathbf{I}_d$ in the following. Otherwise one can first apply the algorithm to the modified moments sequence $\widetilde{H}_i := \hat{L}^{-1} H_i \hat{L}^{-T}$, obtaining a realization (J, L, R). Then $(J, L\hat{L}^T, R\hat{L}^T)$ is a realization of the original moments sequence. (If H_0 is not positive definite, one can apply the algorithm to the moments sequence $\widetilde{H}_i := H_i H_0^{-1}$ instead. Then (J, L, RH_0) is a realization of the original moments sequence.)

Theorem 5.5. *Let $H = (H_k)_{k=0,\dots,n_0-1}$ be a finite moments sequence with $H_0 = \mathbf{I}_d$ such that the first $N := \lceil \frac{n_0}{2} \rceil \cdot d$ columns of $\mathcal{H}$ from (5.2) are linearly independent. Let $J \in \mathbb{R}^{N\times N}$ be given by the coefficients $t_{i,j}$ from (5.5) and L and R by (5.10) with $L_0 = R_0 = \mathbf{I}_d$, where $t_{i-d,i}$ and $u_{i-d,i}$, $0 \le i < d$ are normalized via (5.8). Then (J, L, R) is a minimal realization of H.*

Proof. $H_0 = \mathbf{I}_d$ implies that the base polynomials $\hat{\psi}_k(x) = x^k$ and $\hat{\phi}_k(x) = x^k$ are already biorthogonal and consequently $\psi_k(x) = x^k$ and $\phi_k(x) = x^k$ for $0 \le k < d$. This implies $[\psi_k, \phi_k]_H = e_k^T H_0 e_k = 1$ for $0 \le k < d$.

Applying (5.6) k times inductively yields

$$\begin{bmatrix} \mathbf{0}_{kd} & \mathbf{0}_{kd} & \dots & \mathbf{0}_{kd} \\ a_0 & a_1 & \dots & a_{N-1} \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & \dots & a_{N-1} \end{bmatrix} (J^T)^k + \sum_{i=0}^{k-1} \begin{bmatrix} \mathbf{0}_{id} & \dots & \mathbf{0}_{id} & \mathbf{0}_{id} & \dots & \mathbf{0}_{id} \\ \mathbf{0} & \dots & \mathbf{0} & \xi_1 & \dots & \xi_d \end{bmatrix} (J^T)^{k-i-1}. \tag{5.11}$$

Since J is banded with bandwidth d, the non-zero entries $\xi_1, \dots, \xi_d$ are propagated by only d columns per multiplication with J^T, therefore only the last $(k-i)d$ columns of

$$\begin{bmatrix} \mathbf{0}_{id} & \dots & \mathbf{0}_{id} & \mathbf{0}_{id} & \dots & \mathbf{0}_{id} \\ \mathbf{0} & \dots & \mathbf{0} & \xi_1 & \dots & \xi_d \end{bmatrix} (J^T)^{k-i-1}$$

are non-zero. Consequently, with $N = \lceil \frac{n_0}{2} \rceil d$, the first d columns of the sum in (5.11) are zero if $k \leq \lceil \frac{n_0}{2} \rceil - 1$. For $0 \leq i < d$, $0 \leq j < d$, $0 \leq k < \lceil \frac{n_0}{2} \rceil$ we have

$$(H_k)_{i,j} = \left[x^{i+kd}, x^j\right]_H = \left[x^{kd}\psi_i, \phi_j\right]_H.$$

We apply the jth column of (5.11) to obtain

$$\left[x^{kd}\psi_i, \phi_j\right]_H = \left[\sum_{l=0}^{kd+i-1} (J^k)_{i,l}\psi_l, \phi_j\right]_H = \sum_{l=0}^{kd+i-1} (J^k)_{i,l}[\psi_l, \phi_j]_H.$$

Since ϕ_j is orthogonal to all ψ_l with $l \neq j$, out of all summands in the sum only the one with $l = j$ remains, resulting in

$$(H_k)_{i,j} = \sum_{l=0}^{kd+i-1} (J^k)_{i,l}[\psi_l, \phi_j]_H = (J^k)_{i,j}[\psi_j, \phi_j]_H = (J^k)_{i,j},$$

where $[\psi_j, \phi_j]_H = 1$ follows from the chosen normalization (5.8). Consequently $H_k = L^T J^k R$.

$J \in \mathbb{R}^{N \times N}$ has as many rows and columns as there are polynomials in ψ and in ϕ. As explained in Section 5.2, the algorithm performs $\lceil \frac{n_0}{2} \rceil d$ steps. Since the first $\lceil \frac{n_0}{2} \rceil d$ columns of $\mathcal{H}$ are linearly independent, $\operatorname{rank} \mathcal{H} \geq \lceil \frac{n_0}{2} \rceil d$ holds. Therefore the realization is minimal according to Theorem 5.3. □

The Lanczos algorithm is often formulated for other problems such as solving systems of linear equations or determining the eigenvalues of a matrix, cf. [27]. The method described here is connected to these applications of the Lanczos algorithm. [14, Section 8] briefly describes this connection.

Remark 5.6. As described in Section 5.2, J is a banded matrix with bandwidth d. Consequently the entries $(J^2)_{i,j}$ depend only on entries $J_{i',j'}$ satisfying $i = i'$ and $|j - j'| \leq d$ or satisfying $j = j'$ and $|i - i'| \leq d$. Inductively, entries $(J^k)_{i,j}$ only depend on the entries $t_{i',j'}$ for which $|j - j'| \leq l_1 d$ and $|i - i'| \leq l_2 d$ hold for certain $l_1, l_2 \in \mathbb{N}_0$ with $l_1 + l_2 \leq k - 1$.

Therefore the entries of $L^T J^k R$ depend only on entries $J_{i,j}$ with $i < d \cdot k$ and $j < d \cdot k$ and an extension of the moments sequence $(H_0, \dots, H_{n_0-1})$ by extra elements and the corresponding extension of J by extra rows and columns would have no effect on the values $L^T J^k R$ for $k < N$.

6. Application of the Multidimensional Lanczos Algorithm to the Generalized Langevin Equation

6.1. The Multidimensional Algorithm

As in Chapter 3, we can use the Lanczos method to compute an Ornstein-Uhlenbeck system such that the velocity auto-covariance function of the corresponding Ornstein-Uhlenbeck process $\begin{bmatrix} V(t) \\ Z(t) \end{bmatrix}$ with auxiliary variables Z interpolates a given velocity auto-covariance function $C_V \colon \mathbb{R} \to \mathbb{R}^{d \times d}$ in an equidistant grid $\Delta = \{0, \tau, \ldots, (n_0 - 1)\tau\}$. Analogously to Chapter 3, we first compute a minimal realization (J, L, R) of the moments sequence $(r_j)_{j=0,\ldots,n_0-1}$, $r_j := C_V(j\tau)$ with L, R from (5.10), using the multidimensional Lanczos algorithm from Chapter 5.

By taking $A := \frac{1}{\tau} \operatorname{Log} J$ (if J has no negative real eigenvalues) we subsequently obtain a matrix such that $L^T e^{j\tau A} R = r_j$, which yields the desired Ornstein-Uhlenbeck system. In the following, we explain the remaining steps in the algorithm.

6.1.1. Restrictions on the Velocity Auto-Covariance Function

In the case that the system from which the input data originates satisfies

$$C_V'(0) = \mathbf{0}, \tag{6.1}$$

the obtained Ornstein-Uhlenbeck system should satisfy this condition, too. In order to achieve this, one could again think at adapting r_1 via a Newton iteration, modifying the d^2 entries of r_1 in order to satisfy the d^2 scalar conditions given by $C_V'(0) = \mathbf{0}$. The corresponding derivatives of A can again be obtained via Lemma 3.1. We then replace r_1 by $r_1 + \Delta r_1$ with a matrix Δr_1 according to Newton's method.

The Newton iteration, however, did not converge in our numerical experiments for $d > 1$. Alternative optimization methods (gradient descent and the Levenberg-Marquardt algorithm) did not find a zero of the mapping $r_1 \mapsto C_V'(0)$ either. On top of that, if one succeeds in fulfilling (6.1), another problem arises when considering the Laurent

expansion of $\mathcal{F}C_V^A$ near ∞. To see this we compute the Laurent expansion of $\mathcal{F}C_V^A$ via a Neumann series:

$$\begin{aligned}
\sqrt{2\pi}\mathcal{F}C_V^A(\xi) &= \int_{-\infty}^{\infty} e^{-\mathrm{i}t\xi} C_V^A(t)\,\mathrm{d}t \\
&= \int_0^{\infty} e^{-\mathrm{i}t\xi} C_V^A(t)\,\mathrm{d}t + \int_{-\infty}^{0} e^{-\mathrm{i}t\xi}\big(C_V^A(-t)\big)^T\,\mathrm{d}t \\
&= \int_0^{\infty} e^{-\mathrm{i}t\xi}\frac{1}{\beta m} P e^{tA} P^T\,\mathrm{d}t + \int_0^{\infty} e^{\mathrm{i}t\xi}\frac{1}{\beta m} P e^{tA^T} P^T\,\mathrm{d}t \\
&= \frac{1}{\beta m} P\bigg(\int_0^{\infty} e^{t(A-\mathrm{i}\xi\mathbf{I})}\,\mathrm{d}t + \int_0^{\infty} e^{t(A^T-\mathrm{i}\xi\mathbf{I})}\,\mathrm{d}t\bigg)P^T \\
&= \frac{1}{\beta m} P\big(-(A-\mathrm{i}\xi\mathbf{I})^{-1} - (A^T+\mathrm{i}\xi\mathbf{I})^{-1}\big)P^T \\
&= \frac{1}{\beta m} P\big(-\mathrm{i}\xi^{-1}(\mathrm{i}\xi^{-1}A+\mathbf{I})^{-1} + \mathrm{i}\xi^{-1}(-\mathrm{i}\xi^{-1}A^T+\mathbf{I})^{-1}\big)P^T \\
&= \frac{1}{\beta m} P\bigg(-\mathrm{i}\xi^{-1}\sum_{k=0}^{\infty}\big(-\mathrm{i}\xi^{-1}A\big)^k + \mathrm{i}\xi^{-1}\sum_{k=0}^{\infty}\big(\mathrm{i}\xi^{-1}A^T\big)^k\bigg)P^T \\
&= \frac{1}{\beta m}\sum_{k=1}^{\infty} P\big((-\mathrm{i})^k A^{k-1} + \mathrm{i}^k (A^T)^{k-1}\big)P^T\xi^{-k}
\end{aligned}$$

if ξ is sufficiently large such that $\|\xi^{-1}A\| < 1$ and $\|\xi^{-1}A^T\| < 1$ for any operator norm.

The first summand in the Laurent expansion always vanishes. If $PAP^T = \mathbf{0}$ (i.e. if (6.1) holds), the second summand vanishes, too, leaving

$$\mathcal{F}C_V^A(\xi) = \frac{1}{\beta m\sqrt{2\pi}}\Big(\mathrm{i}P\big(A^2-(A^2)^T\big)P^T\xi^{-3} + P\big(A^3+(A^3)^T\big)P^T\xi^{-4} + O(\xi^{-5})\Big).$$

For $\xi \to \infty$, this sum is dominated by the first summand. $P\big(A^2-(A^2)^T\big)P^T$ being a real matrix, its eigenvalues are complex conjugated, i.e. the eigenvalues of $\mathrm{i}P\big(A^2-(A^2)^T\big)P^T$ are symmetric w.r.t. the imaginary axis. $\mathrm{i}P\big(A^2-(A^2)^T\big)P^T$, however, is hermitian and thus only has real eigenvalues. Consequently $\mathrm{i}P\big(A^2-(A^2)^T\big)P^T$ has no negative eigenvalues if and only if all eigenvalues are 0, i.e. if $PA^2P^T = P(A^2)^TP^T$. To obtain an auto-covariance function of positive type we would therefore have to ensure that P^TA^2P is symmetric on top of satisfying (6.1). (This is consistent with the Volterra equation of second type, (2.12), which implies $PA^2P^T = (C_V^A)''(0+) = -\frac{1}{\beta m}\gamma(0)$ in the case $D = \mathbf{0}$, where $\frac{1}{\beta m}\gamma(0) = C_F(0)$ must be symmetric.)

As mentioned in Chapter 3, the condition (6.1) only matters for certain physical systems. Due to the problems described above, we will restrict ourselves to systems which do not require (6.1) to hold (and where the given data do not satisfy this condition, either) in Section 6.2. We also note that if Newton's method succeeded, it would be feasible only for small values of d due to a runtime of $O(d^6)$ per iteration step. Dropping the condition (6.1) again leads to an Ornstein-Uhlenbeck system which solves the generalized Langevin equation (2.5) with instantaneous friction.

6.1.2. Further Steps in the Algorithm

Similarly to Sections 3.2.2 to 3.2.4 we have to perform some additional steps. First we replace J by a smaller matrix whose eigenvalues are a subset of the eigenvalues of J; notably we remove all eigenvalues which are not contained in the open unit disk as well as some additional eigenvalues which prevent C_V^A from being of positive type and whose coefficients in the corresponding Prony series are (hopefully) negligible. We also "duplicate" negative real eigenvalues of J so that when computing $A := \frac{1}{\tau} \log J$ it is possible to obtain a real A by taking two different (complex conjugated) logarithms of these eigenvalues. Here we can proceed exactly as in Sections 3.2.2 and 3.2.3 except that when removing spurious eigenvalues, the first d rows of the matrix $\widetilde{U}$ from Section 3.2.2 must equal the first d rows of U except for the entries of removed columns (instead of only the first row as in Section 3.2.2). In Section 3.2.2 we required all eigenvalues to be pairwise different. As mentioned at the beginning of Chapter 5, this is no longer the case since writing a rank k coefficient matrix in the Prony series as a sum of rank 1 matrices requires the corresponding eigenvalue to have multiplicity k. A still has to be diagonalizable, though.

Next we can replace A by a similar banded matrix $\widetilde{A}$, similarly to Section 3.3. In exact arithmetic, it would be possible to reapply the block Lanczos method (Algorithm 3) from Chapter 5 using $\frac{1}{\beta m} P^T A^k P$, $k = 0, \dots, N-1$, as moments. This method yields a matrix $\widetilde{A}$ with

$$\frac{1}{\beta m} P^T \widetilde{A}^k P = \frac{1}{\beta m} P^T A^k P$$

and since (A, P, P) and $(\widetilde{A}, P, P)$ are both minimal realizations of the moments sequence $P^T A^k P$, A and $\widetilde{A}$ are equivalent according to Remark 5.4, i.e. they represent the same Ornstein-Uhlenbeck system except for a change of basis which transforms only the auxiliary variables. As in Section 3.2, however, this method is not feasible in practice due to the large numerical errors caused by the large condition of A^k. Consequently we apply another version of the Lanczos algorithm described in [1]. The connection between the two methods is described in [14, Section 8.1].

Here $\widetilde{A}$ is a banded matrix with bandwidth d instead of a tridiagonal matrix as in Chapter 3. Replacing A by $\widetilde{A}$ reduces the computational cost for simulating the corresponding Ornstein-Uhlenbeck process (using the Euler-Maruyama method) from $O(N^2)$ to $O(d \cdot N)$ per simulation step.

As in the one-dimensional case we obtain $K \in \mathbb{R}^{N \times d}$ and $\Sigma \in \mathbb{R}^{N \times N}$ by solving the Lur'e equations

$$\begin{aligned} A_0 S + S A_0^T &= -G G^T \\ SB - C &= -G L^T \\ D + D^T &= -L L^T \end{aligned}$$

(with unknown G, L, and S) where

$$A = \begin{bmatrix} D & B^T \\ -C & A_0 \end{bmatrix} \in \mathbb{R}^{(d|N-d)\times(d|N-d)}, \qquad K = \frac{1}{\sqrt{\beta m}} \begin{bmatrix} L \\ G \end{bmatrix} \in \mathbb{R}^{(d|N-d)\times d}, \\ \Sigma = \frac{1}{\beta m} \begin{bmatrix} \mathbf{I}_d & \mathbf{0} \\ \mathbf{0} & S \end{bmatrix} \in \mathbb{R}^{(d|N-d)\times(d|N-d)}. \tag{6.2}$$

We present an algorithm for solving the Lur'e equations in Appendix A.

6.2. Numerical Results

As mentioned above, the Newton iteration did not succeed in our multidimensional numerical experiments. Hence we here only present results using data which do not satisfy $C_V'(0) = \mathbf{I}$; consequently we did not apply the Newton iteration or any other method to adapt the entries of D (6.2) and we did not set them to any particular value after removing spurious eigenvalues.

In the plots of this section we write $C_{i,j}(t) := C_{V_i,V_j}(t)$ and $C^A_{i,j}(t) := C^A_{V_i,V_j}(t)$ for better readability.

6.2.1. A Three-Dimensional Active Particle

We consider a three-dimensional system of an active particle immersed in a passive Lennard-Jones fluid. The active particle propels itself with a constant force subject to rotational diffusion. The rotational inertia of the active particle is set to zero for simplicity. In addition to being immersed in the Lennard-Jones fluid, the active particle is coupled to a thermal bath via a Markovian Langevin thermostat. The time evolution of the orientation of the active particle is governed by rotational diffusion.

All simulations were performed using LAMMPS [41], where a time step of $\Delta_t = 0.001$ was used. In order to sample at the desired temperature, the system was equilibrated with a Langevin thermostat.

The individual components of the velocity are actually uncorrelated, therefore the off-diagonal VACF entries are pure noise. We consider the following time grids:

- Δ_1: $\tau = 0.15$, $n_0 = 40$
- Δ_2: $\tau = 0.3$, $n_0 = 20$
- Δ_3: $\tau = 0.5$, $n_0 = 12$
- Δ_4: $\tau = 0.2$, $n_0 = 10$

The first three grids cover roughly the same time interval $[0, 6]$; more precisely 6 would be the next point added to each grid if we increased n_0 by 1. The fourth grid covers a shorter time interval; the next point added to Δ_4 would be 2. Such a grid may be advantageous in cases where the long simulation runs required to accurately determine

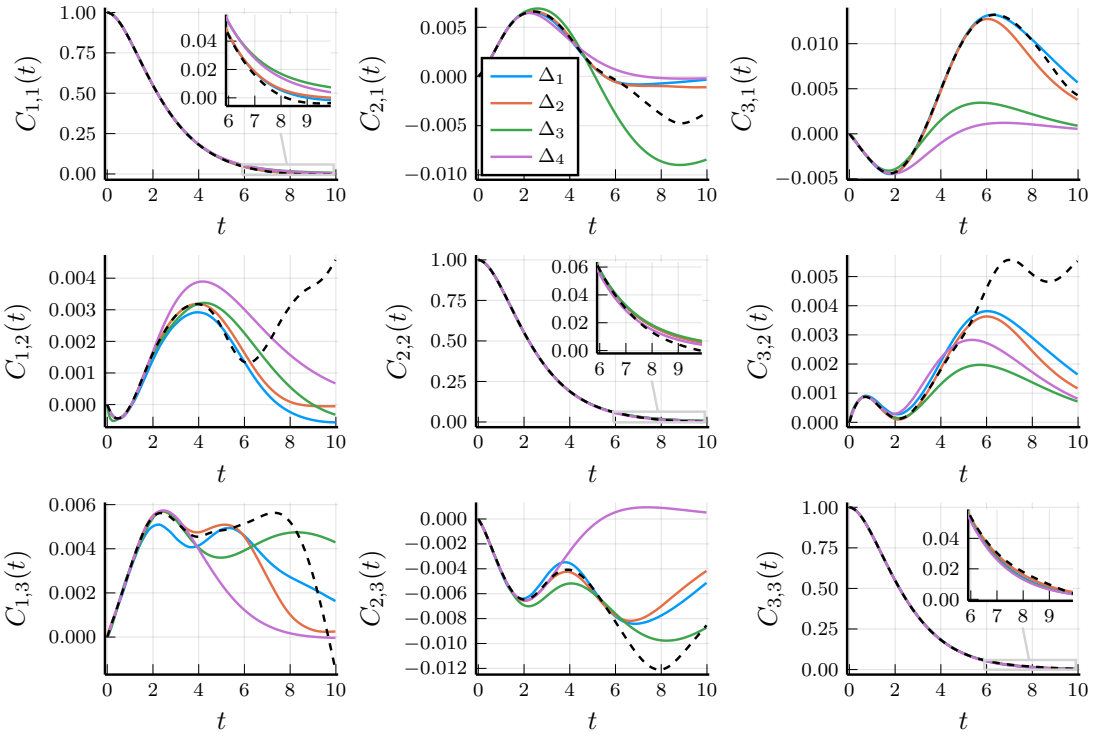


Figure 6.1.: Comparison of the given VACF (black) and the VACFs of the Ornstein-Uhlenbeck systems generated via the Lanczos method (colored)

the velocity auto-covariance function in a longer time interval are too expensive, and it yields smaller Ornstein-Uhlenbeck systems than longer grid with the same step size τ. We consider this grid to assess how the shorter time interval covered impacts the approximation.

Figure 6.1 compares the velocity auto-covariance functions of the three obtained Ornstein-Uhlenbeck systems to the input data. Each of the nine subplots displays one component of the (3×3)-dimensional VACF C_V, which is normalized to obtain $C_V(0) = \mathbf{I}_3$. One can see significant differences between the given VACF data and the VACF of the obtained Ornstein-Uhlenbeck process in several off-diagonal components. However, comparing the scales of the diagonal and off-diagonal components shows that the error is still small compared to the scale of the diagonal entries. The diagonal components are well approximated.

Figure 6.2 displays the difference between the VACF of the Ornstein-Uhlenbeck systems and the original VACF. We see that the scale of the error is comparable for diagonal and off-diagonal entries. Overall, the coarsest grid Δ_3 performs worse than the finer and larger grids Δ_1 and Δ_2. The results of these two grids are comparable. Unsurprisingly, the difference gets much larger after the last interpolation point for Δ_1 and Δ_2, which indicates that the error introduced by the removal of spurious eigenvalues inside the interpolation grid is far smaller than the general approximation error. Conversely the larger error of Δ_3 is mainly not due to a larger error between the interpolation points

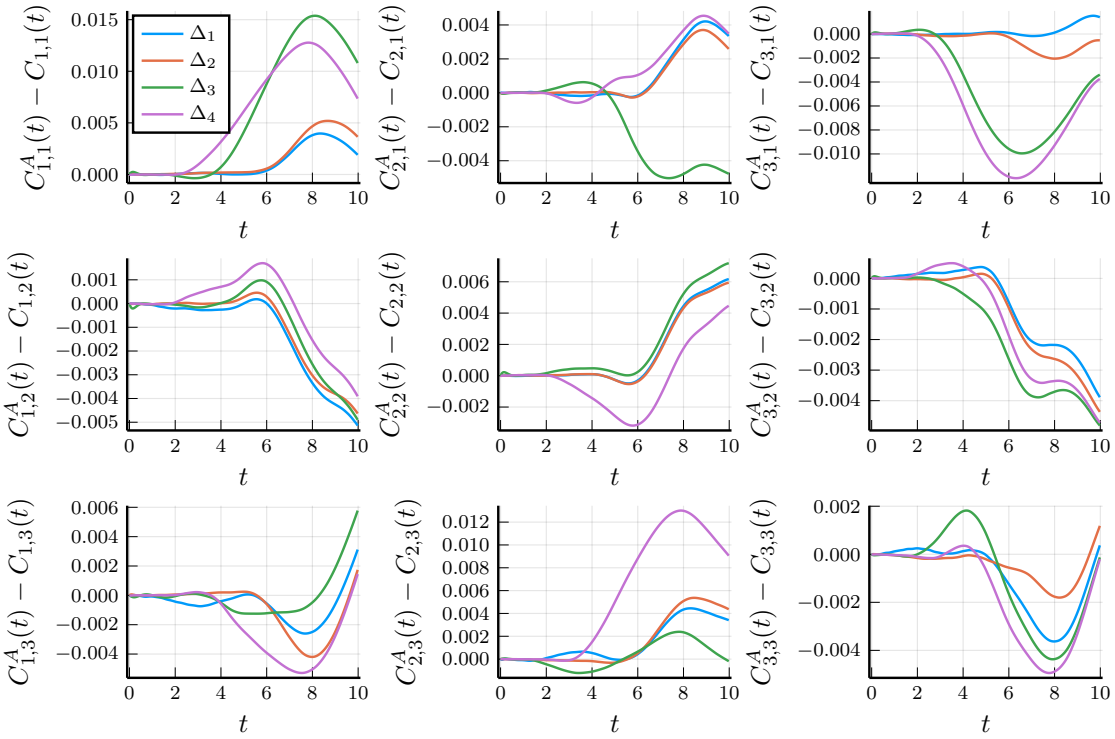


Figure 6.2.: Difference between the input VACF and the VACF of the Ornstein-Uhlenbeck systems generated via the Lanczos method

as one could assume since this is the coarsest grid, but due to fact that the error gets large before the last interpolation point. This indicates a spurious eigenvalue (or several eigenvalues) with a comparably large coefficient in the Prony series representation of C_V^A. Indeed the largest coefficient of a removed eigenvalue is $1.18 \cdot 10^{-4}$ for Δ_3 while it is $1.52 \cdot 10^{-5}$ for Δ_1 and $2.21 \cdot 10^{-6}$ for Δ_2. Figure 6.3 displays the Frobenius norm of the difference between the VACF of the Ornstein-Uhlenbeck systems and the original VACF. Δ_4 performs comparably to Δ_3 with a slightly larger error, as Figure 6.3 shows.

Table 6.1 summarizes the most relevant results: It displays the largest n_0 for which the algorithm succeeded, the number N_{sp} of spurious eigenvalues ($\lambda \in \sigma(A)$ with $\operatorname{Re}\lambda \geq 0$), the number N_{pt} of additional eigenvalues which were removed in order to obtain a VACF of positive type, and the size N of the final matrix A after removing spurious eigenvalues and duplicating negative real eigenvalues. (The size of the initial matrix A before treating these eigenvalues is $d \cdot \lceil \frac{n_0}{2} \rceil$ with $d = 3$.) We see that the initial n_0 was successful for all three grids. The table also displays the relative approximation error,

$$\sum_{t \in \Delta} \|C_V(t) - C_V^A(t)\|_F \,/\, \sum_{t \in \Delta} \|C_V(t)\|_F \tag{6.3}$$

where $\Delta = \{0, 0.001, \ldots, 10\}$ is the grid on which the input VACF was sampled, and the relative error when only considering the diagonal entries (called “diagonal error” in the

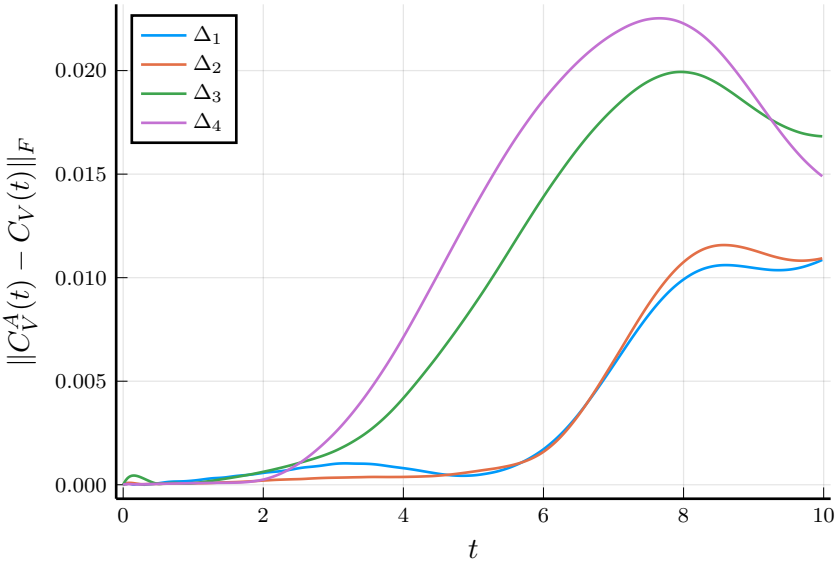


Figure 6.3.: Frobenius norm of the difference between the input VACF and the VACF of the Ornstein-Uhlenbeck systems generated via the Lanczos method

table),

$$\sum_{t\in\Delta} \left\|\operatorname{diag}(C_V(t)) - \operatorname{diag}(C_V^A(t))\right\|_2 / \sum_{t\in\Delta} \left\|\operatorname{diag}(C_V(t))\right\|_2. \tag{6.4}$$

Since the off-diagonal VACF entries are pure noise, one can apply the one-dimensional method from Chapter 3 to each diagonal entry separately instead, obtaining three individual Ornstein-Uhlenbeck systems which can subsequently be combined into a single Ornstein-Uhlenbeck system (where all off-diagonal entries of the VACF are 0). For comparison Table 6.1 also shows the same data for the resulting Ornstein-Uhlenbeck system. (For all three diagonals and all grids the one-dimensional method was successful for the displayed values n_0.) Here the values of N_{sp}, N_{pt}, and N are the added values for all three diagonals. As the one-dimensional algorithm does not approximate the

		Multidimensional algorithm					One-dimensional algorithm				
Grid	n_0	N_{sp}	N_{pt}	N	Error	Diagonal Error	N_{sp}	N_{pt}	N	Error	Diagonal Error
Δ_1	40	30	4	27	0.0080	0.0052	28	8	26	0.0340	0.0180
Δ_2	20	11	0	19	0.0081	0.0049	7	0	24	0.0348	0.0190
Δ_3	12	7	0	11	0.0208	0.0155	4	0	16	0.0362	0.0214
Δ_4	10	3	0	13	0.0249	0.0142	4	0	11	0.0328	0.0150

Table 6.1.: Comparison of the multidimensional and one-dimensional algorithm: Number of interpolation points n_0, number N_{sp} of spurious eigenvalues, number N_{pt} of other removed eigenvalues, size N of the final matrix A, relative approximation error (6.3), and approximation error of the diagonal components (6.4)

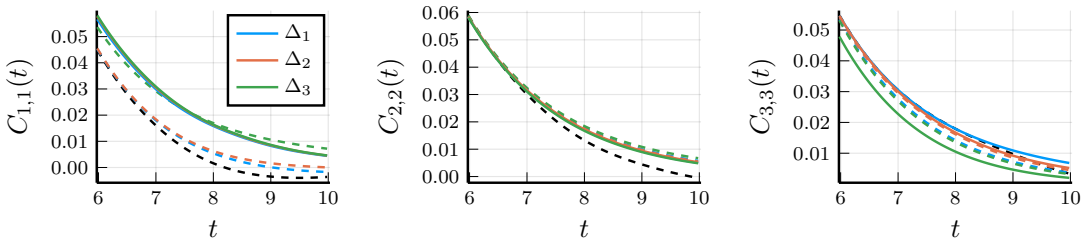


Figure 6.4.: Comparison of the diagonal VACF components using the multidimensional (dashed) and one-dimensional (solid) Lanczos method with the grids Δ_1, Δ_2, and Δ_3. The dashed black line is the input VACF.

off-diagonal entries, the total error is far worse than the error of the multidimensional algorithm. Comparing the diagonal error, however, we see that the multidimensional algorithm still performs comparably to the one-dimensional algorithm and is even much better for Δ_1 and Δ_2.

To investigate why the one-dimensional algorithm performs worse, we compare the resulting VACF approximations in Figures 6.4 and 6.5. Figure 6.4 compares the diagonal entries of the VACF obtained via the multidimensional method using the grids Δ_1, Δ_2, and Δ_3 to the VACFs obtained by applying the one-dimensional method to each diagonal component of the input VACF individually. (It only displays the VACF for $t \geq 6$, where the different lines are better distinguishable.) We used the one-dimensional method without Newton iteration and without setting $A_{1,1}$ to any fixed value. We see that the three-dimensional method yields a better approximation of the first diagonal VACF component for Δ_1, Δ_2, and Δ_3. The reason for this is that for each of these three grids, the one-dimensional algorithm leads to an exponential term in the Prony series with a rather large coefficient whose exponent is just outside the unit disk: The exponent (i.e. the eigenvalue of $J = e^{\tau A}$) in the Prony series is 1.0173 for Δ_1, 1.0323 for Δ_2, and 1.0459 for Δ_3. The corresponding Prony coefficients are -0.0069, -0.0074, and -0.0084, respectively. For the other diagonal components the differences between the two methods are smaller at least until the last grid point with the exception of the third component using Δ_3. Concerning the computational cost, the one-dimensional method is of course preferable although this is no real issue with our examples.

Figure 6.5 displays the approximation errors of the one-dimensional and the multidimensional method for all three grids. They confirm the results from Table 6.1: The multidimensional method seems to yield better results for these input data for Δ_1 and Δ_2 while for Δ_3 the results of both methods are comparable. We see no particular reason why the multidimensional method should actually result in a smaller error than the one-dimensional method, so we assume that the observed behavior is caused by these specific data and that other data will not confirm these results.

Figure 6.6 displays the memory kernels of the obtained Ornstein-Uhlenbeck systems. Again the diagonal components are much larger than the off-diagonal components as expected. The three kernels differ mainly for small t while the tail for large t is similar for all three grids; consequently the obtained memory kernel approximation seems to be

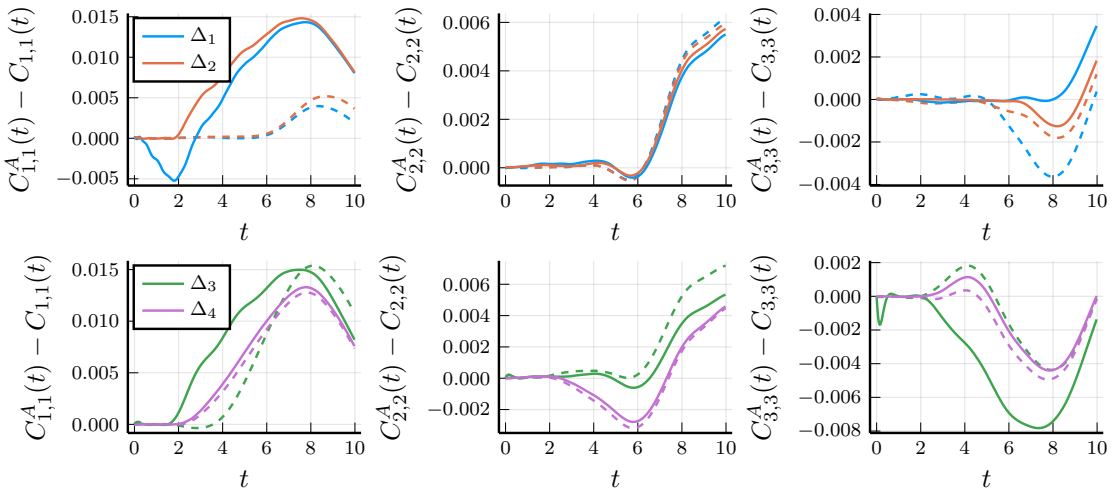


Figure 6.5.: Difference between the diagonal components of the input VACF and the VACF of the Ornstein-Uhlenbeck systems generated via the multidimensional (dashed line) and one-dimensional (solid line) Lanczos method. The top figure shows the results for Δ_1 and Δ_2 while the bottom figure shows the results for Δ_3 and Δ_4.

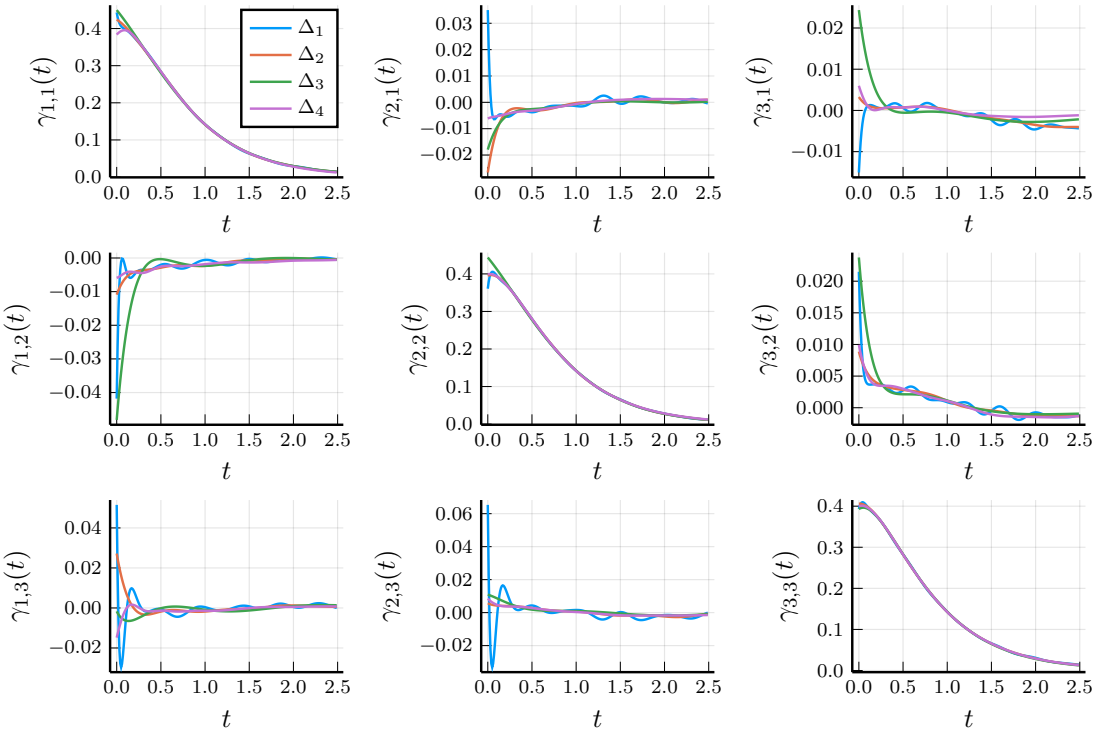


Figure 6.6.: Memory kernels belonging to the generated Ornstein-Uhlenbeck systems

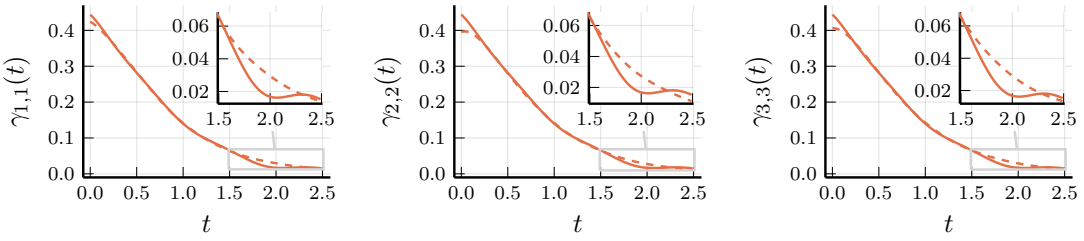


Figure 6.7.: Comparison of the diagonal memory components using the multidimensional (blue) and one-dimensional (red) Lanczos method with the grid Δ_2

reliable for $t \geq 0.3$. Figure 6.7 compares the diagonal components of the memory kernel obtained via the multidimensional method using the grid Δ_2 to the VACFs obtained via applying the one-dimensional method to each diagonal component of the input VACF individually.

Due to the increased number of polynomials compared to Section 3.4, stability concerns are more relevant here when using many grid points. When using 64-bit floating point numbers for the grid Δ_1, while most off-diagonal entries in the "Gram matrix" $[[\psi_i, \phi_j]_H]_{i,j=0,\dots,N-1}$ were close to zero (even during the later steps of the algorithm), a few were not; very few even had values larger than 10 (recall that ψ_i and ϕ_i are normalized such that $|[\psi_i, \phi_i]_H| = 1$), the largest value being 43.6. Consequently extra orthogonalization steps in the algorithm might make sense. Since we do not focus on orthogonalization techniques, we instead performed the Lanczos method from Chapter 5 for this grid using higher precision (256 bit) floating point numbers while using a precision of 64 bit for the remaining steps of the algorithm. While the difference between the VACF approximations using 64 bit and 256 bit precision was visible in the off-diagonal VACF entries of the plots, 64 bit precision still yielded results of comparable quality for this example. The other grids posed no problem concerning computational errors.

6.2.2. A Two-Dimensional S-Shaped Particle

We now use two-dimensional data where the individual components of the velocity are correlated, i.e. the off-diagonal entries are not pure noise.

The data were simulated using a bath of 5000 passive Langevin particles immersed in an implicit solvent. A Markovian Langevin thermostat models the implicit solvent which couples the Langevin particles to a thermal bath. The macromolecule is a rigid S-shaped particle consisting of 100 particles aligned to the two-dimensional plane. The positions of the constituent particles, to receive the S shape, are calculated via $10 \cdot (\sin(t), \sin(2t))$ with t evenly spaced in the interval $[-\pi + 1.5, \pi - 1.5]$. These particles are decoupled from the thermostat and interact with the Langevin particles via a Weeks-Chandler-Anderson potential

$$U(r_{i,j}) = \begin{cases} 4\epsilon\left(\left(\frac{\sigma}{r_{i,j}}\right)^{12} - \left(\frac{\sigma}{r_{i,j}}\right)^{6}\right) + \epsilon, & r_{i,j} \leq \sqrt[6]{2}\sigma, \\ 0, & r_{i,j} > \sqrt[6]{2}\sigma, \end{cases}$$

Grid	n_0	N_{sp}	N_{pt}	N	Error
Δ_1	19	9	0	13	0.0909
Δ_2	10	1	0	11	0.0450
Δ_3	5	0	0	6	0.1848
Δ_4	40	19	0	22	0.0693

Table 6.2.: Results of the multidimensional algorithm using different grids and the data from Section 6.2.2: Number of interpolation points n_0, number N_{sp} of spurious eigenvalues, number N_{pt} of other removed eigenvalues, size N of the final matrix, and relative approximation error (6.3)

where ϵ is set to unity for simplicity, $r_{i,j}$ is the distance and σ the diameter of the particles. The Langevin particles also interact among themselves via the same potential. The particles are considered to be solid, spherical particles. The macromolecule maintains a fixed orientation throughout the simulation.

The trajectories of the molecular dynamics simulations are generated via modeling a canonical ensemble, as implemented in the HOOMD-blue package [5]. Each simulation run is split into an equilibration period of 10^5 time steps and a production run of $5 \cdot 10^7$ time steps with a step size of $\Delta_t = 0.001$.

We consider the following time grids for applying Algorithm 3:

- Δ_1: $\tau = 0.5$, $n_0 = 20$
- Δ_2: $\tau = 1.0$, $n_0 = 10$
- Δ_3: $\tau = 2.0$, $n_0 = 5$
- Δ_4: $\tau = 0.5$, $n_0 = 40$

The first three grids cover roughly the same time interval $[0, 10]$ while Δ_4 covers the longer interval $[0, 20]$.

Table 6.2 summarizes the results of the algorithm for each grid, analogously to Table 6.1. Here the relative error is computed on the grid $\Delta = \{0, 0.01, \ldots, 50\}$. We see that the first tested n_0 was successful for all grids except for Δ_1, where two attempts were necessary.

Figure 6.8 compares the velocity auto-covariance functions of all four Ornstein-Uhlenbeck systems to the input VACF. We see that, as indicated in Table 6.2, Δ_3 produces the largest error while Δ_2 performs best.

Figure 6.9 displays the difference between the input VACF and the VACF of the Ornstein-Uhlenbeck systems generated via the Lanczos method. It shows that the oscillations of Δ_4 exist in all four VACF components. While the longer interpolation interval used by Δ_4 slightly reduces the approximation error compared to Δ_1, it contains spurious oscillations. These are caused by the fact that the extra grid points and larger size of A entail many more spurious eigenvalues in A, as Table 6.2 shows. The removal of these eigenvalues introduces these oscillations and increases the approximation error even

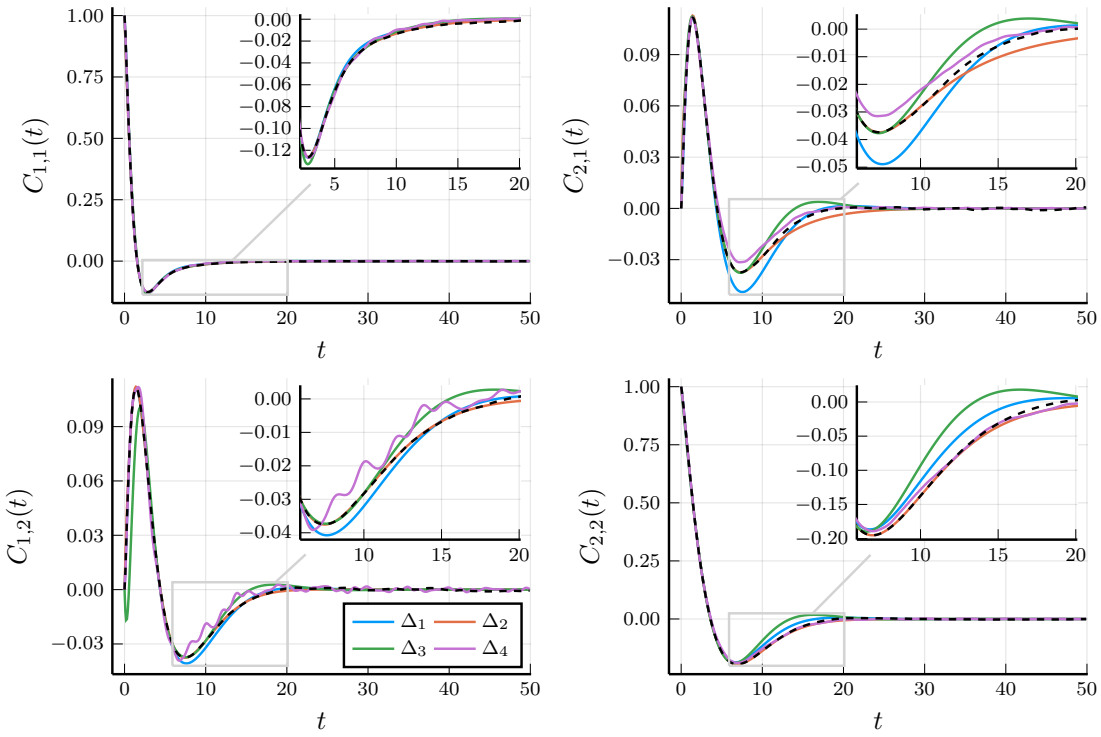


Figure 6.8.: Comparison of the given VACF (black) and the VACFs of the Ornstein-Uhlenbeck systems generated via the Lanczos method (colored)

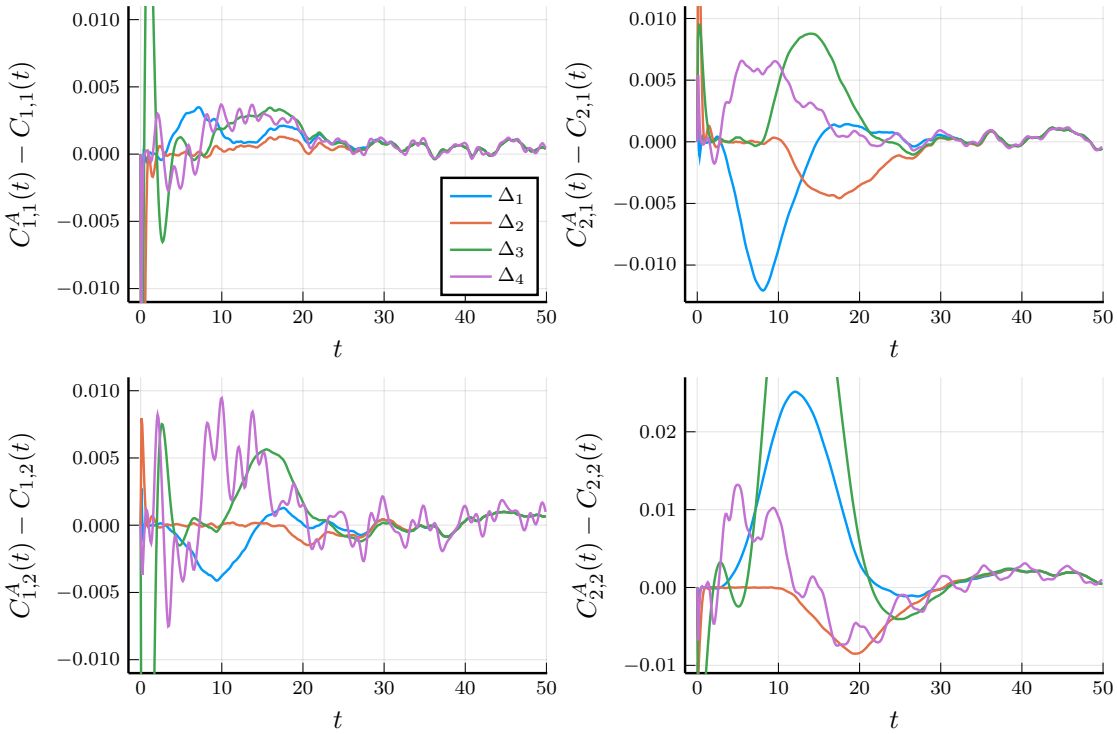


Figure 6.9.: Difference between the input VACF and the VACF of the Ornstein-Uhlenbeck systems generated via the Lanczos method for Δ_1, Δ_2, and Δ_3

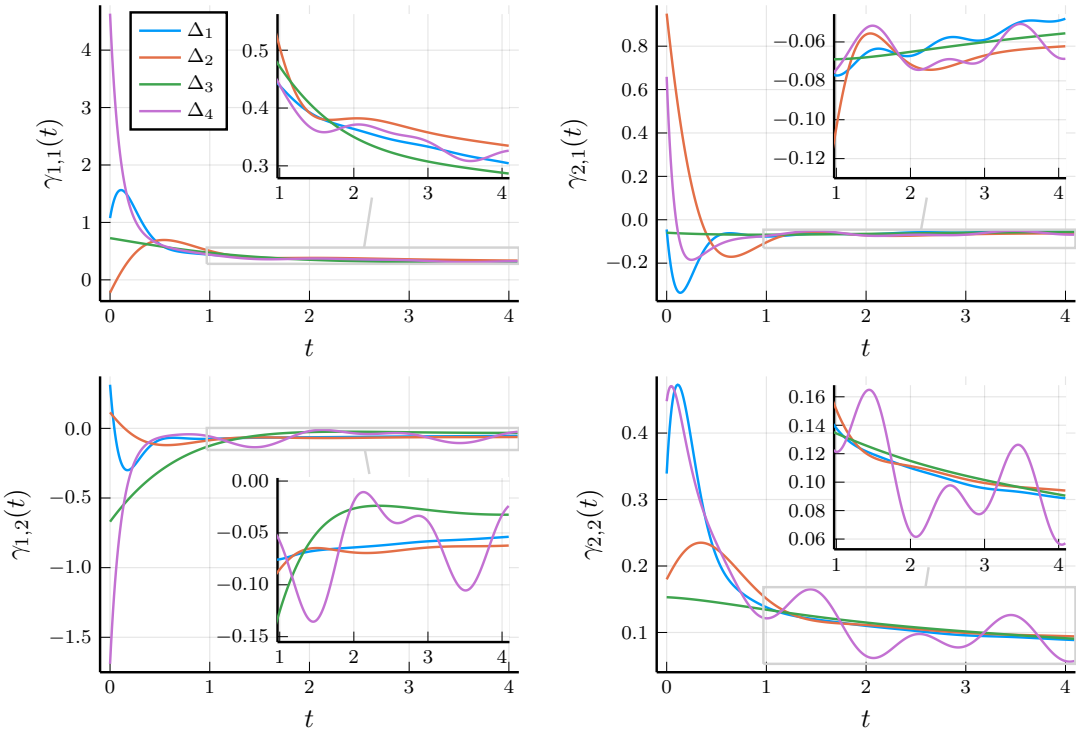


Figure 6.10.: Memory kernels belonging to the Ornstein-Uhlenbeck systems

before the last grid point, which can clearly be seen in Figures 6.8 and 6.9, where it leads to oscillations in the VACF approximation obtained via Δ_4.

Figure 6.10 shows the memory kernels of the obtained Ornstein-Uhlenbeck systems. For t close to 0 they differ significantly. Similarly to the VACF, the memory kernel obtained via Δ_4 contains oscillations which do not exist in the other kernels and which are caused by the large number of spurious eigenvalues in this grid. Due to these oscillations it decays far slower to 0 than the other kernels. As in Section 6.2.1, the approximation of the memory kernel does not seem to be very reliable for small t while the long-term rate of decay seems to be very similar for Δ_1, Δ_2, and Δ_3, except for the $\gamma_{1,2}$ component, where the kernel of Δ_3 decays faster than the other two kernels. A good approximation of the memory kernel is, however, not of great importance as long as the memory kernel leads to a good VACF approximation.

In contrast to Section 6.2.1, the matrices obtained here were small enough that computational errors posed no problem.

A. Solving the Lur'e Equations

Let $A_0 \in \mathbb{R}^{n\times n}$, $B, C \in \mathbb{R}^{n\times d}$, and $D \in \mathbb{R}^{d\times d}$ with $n \geq d$. The algorithm for solving the Lur'e equations

$$A_0 S + S A_0^T = -GG^T \tag{A.1a}$$
$$SB - C = -GL^T \tag{A.1b}$$
$$D + D^T = -LL^T \tag{A.1c}$$

differs depending on whether the matrix $D + D^T$ is singular or not. (Note that for $d > 1$ this distinction is different from the distinction between (2.19) and (2.21), which depends on whether $D = \mathbf{0}$ or not.) We will treat these two cases separately.

A.1. The Regular Lur'e Equations

The regular case (A.1) with $D + D^T$ negative definite can be solved by determining a symmetric, positive semi-definite solution X of the *(algebraic) Riccati equation*

$$QX + XQ^T - XRX + T = \mathbf{0} \tag{A.2}$$

where

$$Q := A_0 + C\Delta^{-1}B^T, \qquad R := B\Delta^{-1}B^T, \qquad T := -C\Delta^{-1}C^T \tag{A.3}$$

with $\Delta := D + D^T$, as the following lemma shows (cf. also [12, p. 584] and [6, Corollary 5.27]).

Lemma A.1. *Let $\Delta = D + D^T$ be negative definite and $LL^T = -\Delta$ a Cholesky decomposition of $-\Delta$. Let further X be a symmetric solution of* (A.2) *with Q, R, and T from* (A.3)*. Then $S := X$, $G := (C - XB)L^{-T}$, and L solve* (A.1)*.*

Proof. (A.1c) obviously follows from the definition of L. We can easily verify the other two equations:

$$\begin{aligned}
-GL^T &= -(C - XB)L^{-T}L^T = SB - C \\
-GG^T &= -(C - XB)L^{-T}L^{-1}(C - XB)^T \\
&= -C\Delta^{-1}B^T X - XB\Delta^{-1}C^T + C\Delta^{-1}C^T + XB\Delta^{-1}B^T X \\
&= A_0 X - QX + XA_0^T - XQ^T - T + XRX \\
&= A_0 X + XA_0^T,
\end{aligned}$$

where the last step is a direct application of (A.2). □

Conversely, given a symmetric solution of (A.1), $X = S$ solves (A.2) as one can directly verify by insertion. If no solution of (A.2) exists, (A.1) is not solvable either.

[31, Section 2] presents an algorithm for solving (A.2) using the Schur decomposition. Note, however, that in our situation R and T are not both positive semi-definite as required there, hence it is not guaranteed that the algorithm succeeds. Since this condition is only sufficient, the algorithm may still succeed. We first present the algorithm and subsequently state less strict conditions. Let

$$Z := \begin{bmatrix} Q^T & -R \\ -T & -Q \end{bmatrix} \in \mathbb{R}^{(n|n)\times(n|n)}. \tag{A.4}$$

Z is a *Hamiltonian matrix*, i.e. JZ is symmetric where

$$J := \begin{bmatrix} \mathbf{0} & -\mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}, \tag{A.5}$$

therefore $\lambda \in \sigma(Z)$ if and only if $-\lambda \in \sigma(Z)$ and the (algebraic and geometric) multiplicities of λ and $-\lambda$ are equal, i.e. the characteristic polynomial is even. (This known property can be proven as follows: For each left eigenvalue $\left[x_1^T, x_2^T\right]^T$, $x_1, x_2 \in \mathbb{R}^n$, of Z corresponding to λ, one can easily verify that $\left[x_2^T, -x_1^T\right]^T$ is a right eigenvalue of Z corresponding to $-\lambda$.)

We consider the *real Schur decomposition* of Z (cf. [31, Theorem 4] and [24, Theorem 2.3.4]),

$$U^T Z U =: S = \begin{bmatrix} S_{1,1} & S_{1,2} \\ S_{2,1} & S_{2,2} \end{bmatrix} \in \mathbb{R}^{(n|n)\times(n|n)} \tag{A.6}$$

where $U \in \mathbb{R}^{2n\times 2n}$ is an orthogonal matrix and S is an upper triangular matrix with the exception of single non-zero entries in the first diagonal below the main diagonal such that no two consecutive entries are non-zero and that for each such non-zero entry the 2×2 diagonal matrix block to which it belongs has two complex conjugated (non-real) eigenvalues of S. There exist standard algorithms for computing the real Schur decomposition where it is possible to arrange the diagonal entries and 2×2 blocks of S in any order. (To be precise, the exact form of the 2×2 blocks depends on the chosen order but if one identifies them with their eigenvalues, the order of the pairs of complex conjugated eigenvalues belonging to the 2×2 blocks and single real eigenvalues can be chosen arbitrarily, cf. [24, Theorem 2.3.4].)

Hence if Z has no pure imaginary eigenvalues, it is possible to arrange that $S_{1,1}$ contains only eigenvalues with negative real part and $S_{2,2}$ only eigenvalues with positive real part. We require this to be the case (which implies $S_{2,1} = \mathbf{0}$ since there is no 2×2 diagonal block consisting of the four central matrix entries). Let

$$U = \begin{bmatrix} U_{1,1} & U_{1,2} \\ U_{2,1} & U_{2,2} \end{bmatrix} \in \mathbb{R}^{(n|n)\times(n|n)}. \tag{A.7}$$

Then $U_{1,1}$ is non-singular and $X := U_{2,1}U_{1,1}^{-1}$ is a symmetric and positive semi-definite solution of (A.2), as proven in [31, Theorem 5] for the case that R and T are positive semi-definite. In the following we adapt the proof given there to our situation.

Definition A.2. *For $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times d}$, the* Krylov subspace $\mathcal{K}_k(A,B)$ *is defined as $\mathcal{K}_k(A,B) := \operatorname{span}\{B, AB, \ldots, A^{k-1}B\} \subseteq \mathbb{R}^n$. The pair (A,B) is called* controllable *if $\mathcal{K}_n(A,B) = \mathbb{R}^n$.*

Remark A.3. If (A,B) is controllable, so is $(A+BM,B)$ for every matrix $M \in \mathbb{R}^{d\times n}$. To verify this, one can easily prove by induction that

$$\mathcal{K}_k(A,B) = \mathcal{K}_k(A+BM,B)$$

for every $k \in \mathbb{N}_0$.

The following lemma is proven e.g. in [48, Theorem 27]:

Lemma A.4. *A triple (A,B,C) is minimal (see Definition 5.1) if and only if (A^T,B) and (A,C) are controllable.*

Theorem A.5. *Let R and T be symmetric and (Q^T,R) controllable. Let Z and U be defined as in* (A.4) *and* (A.6)*, respectively, where U has blocks as in* (A.7)*. If Z has no pure imaginary eigenvalues, then $U_{1,1}$ is non-singular and $X := U_{2,1}U_{1,1}^{-1}$ is symmetric and solves the Riccati equation* (A.2)*. If Q, R, and T are given by* (A.3) *with a stable A_0, then X is positive semi-definite.*

Proof. We follow the proof from [31, Theorem 5] and adapt it to our situation. Since Z has no pure imaginary eigenvalues and $\sigma(S_{1,1}) \cup \sigma(S_{2,2}) = \sigma(S) = \sigma(Z)$, we can choose S in (A.6) in such a way that $S_{1,1}$ has the eigenvalues of Z with negative real part while $S_{2,2}$ has the eigenvalues with positive real part, i.e. $S_{1,1}$ is stable.

We first prove that $U_{1,1}$ is non-singular: For simplicity we first consider the complex Schur decomposition (cf. [24, Theorem 2.3.1]), i.e. $Z = \widetilde{U}\widetilde{S}\widetilde{U}^{-1}$ with $\widetilde{S}, \widetilde{U} \in \mathbb{C}^{2n\times 2n}$ where $\widetilde{S}$ is upper triangular and $\widetilde{U}$ unitary. Assume that $\widetilde{U}_{1,1}$ is singular. Then we can assume without loss of generality that the first column of $\widetilde{U}_{1,1}$ is a zero column. Let λ be the first diagonal element of $\widetilde{S}$ and $u \in \mathbb{R}^n$ the first column of $\widetilde{U}_{2,1}$, $u \neq \mathbf{0}$. Then

$$\begin{bmatrix} -Ru \\ -Qu \end{bmatrix} = \begin{bmatrix} Q^T & -R \\ -T & -Q \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ u \end{bmatrix} = Z \begin{bmatrix} \mathbf{0} \\ u \end{bmatrix} = \widetilde{U}\widetilde{S}\widetilde{U}^{-1} \begin{bmatrix} \mathbf{0} \\ u \end{bmatrix} = \widetilde{U}\widetilde{S} \begin{bmatrix} e_1 \\ \mathbf{0} \end{bmatrix} = \lambda \widetilde{U} \begin{bmatrix} e_1 \\ \mathbf{0} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{0} \\ u \end{bmatrix},$$

i.e. $Ru = \mathbf{0}$ and $Qu = -\lambda u$, which implies $u^T(Q^T)^k R = (-\lambda)^k u^T R = \mathbf{0}$ for every k, i.e. u is orthogonal to $\mathcal{K}_k(Q^T,R)$. This contradicts the requirement that (Q^T,R) be controllable. Hence $\widetilde{U}_{1,1}$ must be non-singular.

Now let $Z = USU^{-1}$ be the real Schur decomposition of Z and $Z = \widetilde{U}\widetilde{S}\widetilde{U}^{-1}$ the complex Schur decomposition using the same order of eigenvalues. In this case $U = \widetilde{U}D$ and $\widetilde{S} = DSD^{-1}$ where the change-of-basis matrix $D \in \mathbb{R}^{2n\times 2n}$ is a block-diagonal matrix

which consists of single ones and unitary 2×2 blocks on the diagonal; a 2×2 block appears whenever the corresponding diagonal entries of S form a 2×2 block. Since $S_{1,1}$ and $\widetilde{S}_{1,1}$ correspond to the eigenvalues with negative real part while $S_{2,2}$ and $\widetilde{S}_{2,2}$ correspond to those with positive real part, the four central entries of D do not form a 2×2 diagonal block, i.e. D has the form

$$D = \begin{bmatrix} D_{1,1} & \mathbf{0} \\ \mathbf{0} & D_{2,2} \end{bmatrix} \in \mathbb{R}^{(n|n)\times(n|n)}.$$

Since $\widetilde{U}_{1,1}$ and $D_{1,1}$ are non-singular, so is $U_{1,1} = \widetilde{U}_{1,1} D_{1,1}$.

Inserting $X = U_{2,1} U_{1,1}^{-1}$ into (A.2) and using (A.6) and the fact that $S_{2,1} = \mathbf{0}$ yields

$$\begin{aligned} QX + XQ^T - XRX + T &= \begin{bmatrix} X & -\mathbf{I} \end{bmatrix} Z \begin{bmatrix} \mathbf{I} \\ X \end{bmatrix} \\ &= \begin{bmatrix} U_{2,1}U_{1,1}^{-1} & -\mathbf{I} \end{bmatrix} Z \begin{bmatrix} U_{1,1} \\ U_{2,1} \end{bmatrix} U_{1,1}^{-1} \\ &= \begin{bmatrix} U_{2,1}U_{1,1}^{-1} & -\mathbf{I} \end{bmatrix} \left(\begin{bmatrix} U_{1,1} \\ U_{2,1} \end{bmatrix} S_{1,1} + \begin{bmatrix} U_{1,2} \\ U_{2,2} \end{bmatrix} S_{2,1} \right) U_{1,1}^{-1} \\ &= \begin{bmatrix} U_{2,1}U_{1,1}^{-1} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} U_{1,1} \\ U_{2,1} \end{bmatrix} S_{1,1} U_{1,1}^{-1}. \end{aligned}$$

The right-hand side is $\mathbf{0}$ since

$$\begin{bmatrix} U_{2,1}U_{1,1}^{-1} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} U_{1,1} \\ U_{2,1} \end{bmatrix} = \mathbf{0};$$

consequently X solves the Riccati equation (A.2). It remains to prove that X is symmetric and positive semi-definite. For the symmetry let J be defined as in (A.5) and let

$$Y := \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} U^T J U \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} U_{1,1}^T & U_{2,1}^T \end{bmatrix} J \begin{bmatrix} U_{1,1} \\ U_{2,1} \end{bmatrix} = U_{2,1}^T U_{1,1} - U_{1,1}^T U_{2,1}.$$

$S_{2,1} = \mathbf{0}$ implies

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} U^T J U S \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} = Y S_{1,1} + \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} U^T J U \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} S_{2,1} = Y S_{1,1},$$

and analogously

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} S^T U^T J U \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} = S_{1,1}^T Y + S_{2,1}^T \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} U^T J U \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} = S_{1,1}^T Y.$$

Comparing both left-hand sides, we see that

$$U^T J U S = U^T J Z U = U^T Z^T J^T U = S^T U^T J^T U = -S^T U^T J U,$$

therefore Y is a solution of the Lyapunov equation (with known right-hand side)

$$S_{1,1}^T Y + Y S_{1,1} = \mathbf{0}.$$

Since $S_{1,1}$ is stable, $Y = \mathbf{0}$ is the unique solution according to Theorem 1.21. Hence $Y = \mathbf{0}$ and thus $U_{1,1}^T U_{2,1} = U_{2,1}^T U_{1,1}$. Multiplying by $U_{1,1}^{-1}$ and $U_{1,1}^{-T}$, respectively, from both sides shows that

$$X = U_{2,1}U_{1,1}^{-1} = U_{1,1}^{-T}U_{1,1}^T U_{2,1}U_{1,1}^{-1} = U_{1,1}^{-T}U_{2,1}^T U_{1,1}U_{1,1}^{-1} = U_{1,1}^{-T}U_{2,1}^T = X^T.$$

If Q, R, and T are given by (A.3), $S = X$ is a solution of the Lyapunov equation (A.1a) with G given by Lemma A.1. If A_0 is stable, S is positive semi-definite according to Theorem 1.21. □

Another proof of Theorem A.5 can be found in [4, Theorem 5.4.1].

Lemma A.6. *Let Q, R, and T be given by* (A.3)*. If the transfer function of (A_0, B, C, D), given by $\kappa(z) := -D + B^T(z\mathbf{I}_n - A_0)^{-1}C$, is strictly positive real (see Definition 2.14(ii)), Z from* (A.4) *has no pure imaginary eigenvalues.*

Proof. Assume that Z has a pure imaginary eigenvalue λ with corresponding eigenvector x:

$$\begin{bmatrix} Q^T & -R \\ -T & -Q \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \qquad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^{(n|n)},$$

which after inserting (A.3) is equivalent to

$$\begin{aligned} B\Delta^{-1}(B^T x_2 - C^T x_1) &= (-\lambda\mathbf{I} + A_0^T)x_1 \\ C\Delta^{-1}(C^T x_1 - B^T x_2) &= (\lambda\mathbf{I} + A_0)x_2 \end{aligned} \tag{A.8}$$

with $\Delta := D + D^T$. Let $y := \Delta^{-1}(B^T x_2 - C^T x_1)$. Then

$$\begin{aligned} (\kappa(-\lambda) + \kappa(-\lambda)^*)y &= \big(-D + B^T(-\lambda\mathbf{I} - A_0)^{-1}C - \\ &\qquad D^T + C^T(-\lambda\mathbf{I} - A_0)^{-*}B\big)\Delta^{-1}(B^T x_2 - C^T x_1) \\ &= (-D - D^T)\Delta^{-1}(B^T x_2 - C^T x_1) + \\ &\quad \big(B^T(-\lambda\mathbf{I} - A_0)^{-1}C + C^T(-\lambda\mathbf{I} - A_0)^{-*}B\big)\Delta^{-1}(B^T x_2 - C^T x_1) \\ &= -B^T x_2 + C^T x_1 + B^T(-\lambda\mathbf{I} - A_0)^{-1}C\Delta^{-1}(B^T x_2 - C^T x_1) + \\ &\quad C^T(-\lambda\mathbf{I} - A_0)^{-*}B\Delta^{-1}(B^T x_2 - C^T x_1). \end{aligned}$$

Inserting (A.8) and $(-\lambda\mathbf{I} - A_0)^{-*} = (\lambda\mathbf{I} - A_0^T)^{-1}$ yields

$$\begin{aligned} (\kappa(-\lambda) + \kappa(-\lambda)^*)y &= -B^T x_2 + C^T x_1 - B^T(-\lambda\mathbf{I} - A_0)^{-1}(\lambda\mathbf{I} + A_0)x_2 + \\ &\qquad C^T(\lambda\mathbf{I} - A_0^T)^{-1}(-\lambda\mathbf{I} + A_0^T)x_1 \\ &= \mathbf{0}. \end{aligned}$$

Consequently $\kappa(-\lambda)+\kappa(-\lambda)^*$ is not positive definite and κ is not strictly positive real. □

If (A_0, B, C) is minimal, (A_0^T, B) is controllable and so is $(A_0^T + B\Delta^{-1}C^T, B) = (Q^T, B)$, see Remark A.3 and Lemma A.4. This in turn implies that that $(Q^T, R) = (Q^T, B\Delta^{-1}B^T)$ is controllable since span B = span R. Since Algorithm 3 from Section 5.2 yields a minimal realization (A, P, P) (where P denotes the projection onto the first d coordinates), the following lemma asserts that the controllability conditions in Theorem A.5 are fulfilled.

Lemma A.7. *Let*

$$A := \begin{bmatrix} D & B^T \\ -C & A_0 \end{bmatrix} \in \mathbb{R}^{(d|n)\times(d|n)} \quad \textit{and} \quad P := \begin{bmatrix} \mathbf{I}_d \\ \mathbf{0}_{n\times d} \end{bmatrix}.$$

If (A, P) is controllable, then (A_0, C) is controllable, too. If (A^T, P) is controllable, then (A_0^T, B) is controllable, too.

Proof. First we note that $\mathcal{K}_k(A_0, C) \subseteq \mathcal{K}_{k+1}(A_0, C)$ for all k. If equality holds for any k, then

$$\mathcal{K}_{k+2}(A_0, C) \subseteq \mathcal{K}_{k+1}(A_0, A_0C) = \mathcal{K}_k(A_0, A_0C) \subseteq \mathcal{K}_{k+1}(A_0, C)$$

and thus $\mathcal{K}_{k+2}(A_0, C) = \mathcal{K}_{k+1}(A_0, C)$. Therefore if $\mathcal{K}_k(A_0, C) = \mathcal{K}_{k+1}(A_0, C)$ for a $k < n$, we have $\mathcal{K}_m(A_0, C) = \mathcal{K}_k(A_0, C)$ for all $m \geq k$.

Otherwise $\dim \mathcal{K}_k(A_0, C) < \dim \mathcal{K}_{k+1}(A_0, C)$ for all $k < n$, i.e. $\dim \mathcal{K}_k(A_0, C) \geq k$ for all $k \leq n$, hence $\mathcal{K}_n(A_0, C) = \mathbb{R}^n$. In both cases it follows that

$$\mathcal{K}_m(A_0, C) = \mathcal{K}_n(A_0, C) \qquad \text{for all } m \geq n.$$

Now let $A^k P =: \begin{bmatrix} D_k \\ C_k \end{bmatrix} \in \mathbb{R}^{(d|n)\times d}$. For C_k this yields the recursive formula

$$\begin{bmatrix} D_{k+1} \\ C_{k+1} \end{bmatrix} = A \begin{bmatrix} D_k \\ C_k \end{bmatrix} = \begin{bmatrix} DD_k + B^T C_k \\ -CD_k + A_0 C_k \end{bmatrix}, \qquad \begin{bmatrix} D_0 \\ C_0 \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix}.$$

By induction one easily proves that span $C_k \subseteq \mathcal{K}_k(A_0, C)$ for every k and therefore span $C_k \subseteq \mathcal{K}_n(A_0, C)$.

Let us assume that (A, P) is controllable while (A_0, C) is not, i.e. $y \perp \mathcal{K}_n(A_0, C)$ for a $y \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ and thus $y^T A_0^k C = 0$ for every k. This implies

$$\begin{bmatrix} \mathbf{0}_d \\ y \end{bmatrix}^T A^k P = y^T C_k = \mathbf{0}$$

for every $k \in \mathbb{N}_0$, which contradicts the assumption that (A, P) is controllable.

Analogously one proves the controllability of (A_0^T, B). □

We summarize the results of this section in the following corollary:

Corollary A.8. *Let* $A := \begin{bmatrix} D & B^T \\ -C & A_0 \end{bmatrix} \in \mathbb{R}^{(d|n)\times(d|n)}$ *such that* $D+D^T$ *is negative definite,* (A, P, P) *is minimal and* $\kappa(z) := -D + B^T(z\mathbf{I}_n - A_0)^{-1}C$ *is strictly positive real. Then the Riccati equation* (A.2) *with* Q, R, *and* T *given by* (A.3) *has a symmetric and positive semi-definite solution* X *given by* $X = U_{2,1}U_{1,1}^{-1}$ *with* $U_{1,1}$, $U_{2,1}$ *from* (A.6) *and* (A.7). *The solution of the corresponding Lur'e equations* (A.1) *is given by by Lemma A.1.*

Remark A.9. When implementing the above algorithm for solving (A.2), we test if Z has an imaginary eigenvalue instead of testing if κ is strictly positive real. However, eigenvalue algorithms for general matrices can introduce significant numerical errors in the obtained eigenvalues. In particular, eigenvalues which should be imaginary can have real parts several orders of magnitude larger than machine precision. There are, however, special eigenvalue algorithms for Hamiltonian matrices which respect the symmetry of their eigenvalues w.r.t. the imaginary axis and thus do not introduce a spurious real part to pure imaginary eigenvalues, cf. e.g. [7, Algorithm 4.2]. (The periodic QR algorithm, which is mentioned there, can be replaced by any other eigenvalue algorithm as long as it is guaranteed that the computed eigenvalues of a real matrix are exactly symmetric w.r.t to the real axis.) The algorithm described there applies Householder transformations and Givens rotations to transform Z into an equivalent matrix

$$U^*ZU = \begin{bmatrix} Z_{1,1} & Z_{1,2} \\ \mathbf{0} & Z_{2,2} \end{bmatrix} \in \mathbb{R}^{(n|n)\times(n|n)}$$

where U is orthogonal, $Z_{1,1}$ is an upper triangular matrix, and $Z_{2,2}^T$ an upper Hessenberg matrix, e.g. via [7, Algorithm 4.4]. Then the eigenvalues of Z are all complex square roots of the eigenvalues of $-Z_{2,2}^T Z_{1,1}$, cf. [7, Section 3]. Alternatively one can test if an eigenvalue close to the imaginary axis, whose real part might be 0 in exact arithmetic, is part of a quadruple $\pm a \pm b\mathrm{i}$ of eigenvalues of Z or not.

A.2. The Singular Lur'e Equations

In this section we describe the algorithm presented in [49] for solving the singular Lur'e equations (A.1) with $D + D^T$ singular. While we retain most of the notation introduced in [49], we slightly adapt it to our needs. We require B and C to have full rank. Let $r := \operatorname{rank}(D + D^T) < d$ and $k := d - r$. Then the reduction process described below will reduce the dimension of the Lur'e equations by k. The resulting equations can be regular, allowing for a solution via the Riccati equation (A.2), or singular, which means that another reduction step is necessary.

A.2.1. Normalizing the Equation

The first step is transforming the system (A_0, B, C, D) into the following normalized form:

$$\begin{aligned} A_0 &= \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \in \mathbb{R}^{(k|n-k)\times(k|n-k)}, \qquad B = \begin{bmatrix} B_s & B_r \end{bmatrix} \in \mathbb{R}^{n\times(k|r)}, \\ C &= \begin{bmatrix} C_s & C_r \end{bmatrix} \in \mathbb{R}^{n\times(k|r)}, \qquad \Delta := D + D^T = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Delta_r \end{bmatrix} \in \mathbb{R}^{(k|r)\times(k|r)}, \\ C_s &= \begin{bmatrix} C_{s1} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(k|n-k)\times k}, \qquad C_r = \begin{bmatrix} C_{r1} \\ C_{r2} \end{bmatrix} \in \mathbb{R}^{(k|n-k)\times r}, \\ B_s &= \begin{bmatrix} B_{s1} \\ B_{s2} \end{bmatrix} \in \mathbb{R}^{(k|n-k)\times k}, \qquad B_r = \begin{bmatrix} B_{r1} \\ B_{r2} \end{bmatrix} \in \mathbb{R}^{(k|n-k)\times r} \end{aligned} \tag{A.9}$$

where $r = \operatorname{rank}\Delta$. This form of Δ and the corresponding representation of C and B can be obtained as follows: We introduce the change-of-basis matrix $\Gamma = [\Gamma_1, \Gamma_2] \in \mathbb{R}^{d\times(k|r)}$ which contains the eigenvectors of Δ in its columns where the columns of Γ_1 contain eigenvectors of Δ corresponding to the eigenvalue 0 and the columns of Γ_2 contain eigenvectors of Δ corresponding to non-zero eigenvalues. (To take numerical errors into account, one of course has to regard eigenvalues as zero if they are close to zero.) The eigenvectors are chosen in such a way that Γ is an orthogonal matrix. This change of basis will lead to Δ having the desired form. (In the case $d = 1$ we can always choose $\Gamma = \mathbf{I}_d$.)

To obtain the desired form of C_s, we choose a second orthogonal change-of-basis matrix $Q = [Q_1, Q_2] \in \mathbb{R}^{n\times n}$ such that the columns of Q_2 span the orthogonal complement of the image of $C\Gamma_1$ (which may be trivial) and Q_1 is arbitrary as long as Q is an orthogonal matrix. This is the case if $C\Gamma_1 = QC_s$ is a QR decomposition of $C\Gamma_1$ for some matrix $C_s \in \mathbb{R}^{n\times k}$ whose upper part C_{s1} is an upper triangular matrix and whose lower part consists of zeros. Then applying both changes of basis, i.e. replacing A_0, B, C, and D by $Q^T A_0 Q$, $Q^T B\Gamma$, $Q^T C\Gamma$, and $\Gamma^T D\Gamma$, respectively, yields a system where C and Δ have the desired form, as one easily verifies. Given a solution $(\widetilde{S}, \widetilde{G}, \widetilde{L})$ of these transformed Lur'e equations,

$$\begin{aligned} Q^T A_0 Q\widetilde{S} + \widetilde{S}Q^T A_0^T Q &= \widetilde{G}\widetilde{G}^T \\ \widetilde{S}Q^T B\Gamma - Q^T C\Gamma &= \widetilde{G}\widetilde{L}^T \\ \Gamma^T D\Gamma + \Gamma^T D^T\Gamma &= \widetilde{L}\widetilde{L}^T, \end{aligned}$$

we can multiply all three equations by appropriate matrices from the left and right to transform them into

$$\begin{aligned} A_0 Q\widetilde{S}Q^T + Q\widetilde{S}Q^T A_0^T &= Q\widetilde{G}\widetilde{G}^T Q^T \\ Q\widetilde{S}Q^T B - C &= Q\widetilde{G}\widetilde{L}^T\Gamma^T \\ D + D^T &= \Gamma\widetilde{L}\widetilde{L}^T\Gamma^T \end{aligned}$$

and obtain the following solution of the original equations:

$$S = Q\widetilde{S}Q^T, \qquad L = \Gamma\widetilde{L}, \qquad G = Q\widetilde{G}.$$

A.2.2. Separating the Singular Dimensions

In the following we will assume that C and Δ have the form (A.9). This allows us to rewrite these "partially singular" (since $D + D^T$ has both zero and non-zero eigenvalues) into "completely singular" Lur'e equations: The Lyapunov equation corresponding to the equations is

$$-KK^T = \begin{bmatrix} \mathbf{0} & \mathbf{0} & B_s^T S - C_s^T \\ \mathbf{0} & \Delta_r & B_r^T S - C_r^T \\ SB_s - C_s & SB_r - C_r & A_0 S + SA_0^T \end{bmatrix} \in \mathbb{R}^{(k|r|n)\times(k|r|n)},$$

or (equivalently)

$$SB_s - C_s = 0, \qquad \begin{bmatrix} \Delta_r & B_r^T S - C_r^T \\ SB_r - C_r & A_0 S + SA_0^T \end{bmatrix} = -K_r K_r^T \tag{A.10}$$

with $K = \begin{bmatrix} \mathbf{0}_k \\ K_r \end{bmatrix}$. Note that $SB_s = C_s$ implies

$$C_s^T B_s = B_s^T S B_s = (C_s^T B_s)^T \quad \text{and thus} \quad C_{s1}^T B_{s1} = (C_{s1}^T B_{s1})^T,$$

(since $C_s^T B_s = C_{s1}^T B_{s1} + 0 \cdot B_{s2}$), i.e. $C_s^T B_s = C_{s1}^T B_{s1}$ is symmetric and positive definite, which implies that B_{s1} is non-singular.

Concerning the condition $SB_s = C_s$, [49, Lemma 3.1] proves the following lemma:

Lemma A.10. *Let B_s and C_s have the form* (A.9)*. Then $SB_s = C_s$ with a symmetric and positive definite S if and only if $C_s^T B_s$ is symmetric and positive definite and*

$$S = \begin{bmatrix} S_{11} & S_{12} \\ S_{12}^T & S_1 \end{bmatrix} \in \mathbb{R}^{(k|n-k)\times(k|n-k)} \tag{A.11}$$

with

$$\begin{aligned} S_{11} &= C_{s1} B_{s1}^{-1} + B_{s1}^{-T} B_{s2}^T S_1 B_{s2} B_{s1}^{-1} \\ S_{12} &= -B_{s1}^{-T} B_{s2}^T S_1 \end{aligned} \tag{A.12}$$

and some symmetric, positive definite $S_1 \in \mathbb{R}^{(n-k)\times(n-k)}$.

Proof. To prove sufficiency, we assume that $C_s^T B_s = C_{s1}^T B_{s1}$ is symmetric and positive definite and that (A.11) and (A.12) hold for some symmetric, positive definite S_1. With

$$F := \begin{bmatrix} \mathbf{I} & -S_{12} S_1^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & B_{s1}^{-T} B_{s2}^T \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

we obtain

$$FSF^T = \begin{bmatrix} \mathbf{I} & -S_{12}S_1^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} S_{11} & S_{12} \\ S_{12}^T & S_1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -S_1^{-1}S_{12}^T & \mathbf{I} \end{bmatrix}$$
$$= \begin{bmatrix} S_{11} - S_{12}S_1^{-1}S_{12}^T & \mathbf{0} \\ \mathbf{0} & S_1 \end{bmatrix} = \begin{bmatrix} C_{s1}B_{s1}^{-1} & \mathbf{0} \\ \mathbf{0} & S_1 \end{bmatrix}$$

where $C_{s1}B_{s1}^{-1}$ is symmetric and positive definite since

$$C_{s1}B_{s1}^{-1} = B_{s1}^{-T}B_{s1}^T C_{s1}B_{s1}^{-1} = B_{s1}^{-T}C_{s1}^T B_{s1}B_{s1}^{-1} = (C_{s1}B_{s1}^{-1})^T$$

and

$$x^T C_{s1}B_{s1}^{-1}x = x^T B_{s1}^{-T}B_{s1}^T C_{s1}B_{s1}^{-1}x = (B_{s1}^{-1}x)^T B_{s1}^T C_{s1}(B_{s1}^{-1}x) > 0$$

for every $x \neq 0$ as $B_{s1}^T C_{s1}$ is positive definite. This implies that S is positive definite and one directly verifies

$$SB_s - C_s = \begin{bmatrix} S_{11}B_{s1} + S_{12}B_{s2} \\ S_{12}^T B_{s1} + S_1 B_{s2} \end{bmatrix} - \begin{bmatrix} C_{s1} \\ \mathbf{0} \end{bmatrix} = \mathbf{0} \tag{A.13}$$

which proves the sufficiency. Necessity follows directly from the fact that S being positive definite implies S_1 being positive definite and that solving (A.13) yields (A.12). □

A.2.3. The Dimension Reduction

To obtain S_1, we apply another change of basis, using the matrix

$$T_r := \begin{bmatrix} \mathbf{I}_r & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & B_{s1} & \mathbf{0} \\ \mathbf{0} & B_{s2} & \mathbf{I}_{n-k} \end{bmatrix} \in \mathbb{R}^{(r|k|n-k)\times(r|k|n-k)}$$

with

$$T_r^{-1} = \begin{bmatrix} \mathbf{I}_r & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & B_{s1}^{-1} & \mathbf{0} \\ \mathbf{0} & -B_{s2}B_{s1}^{-1} & \mathbf{I}_{n-k} \end{bmatrix} \in \mathbb{R}^{(r|k|n-k)\times(r|k|n-k)}.$$

Setting

$$Z := \begin{bmatrix} \Delta_r & B_r^T S - C_r^T \\ SB_r - C_r & A_0 S + S A_0^T \end{bmatrix} \in \mathbb{R}^{(r|n)\times(r|n)}$$

yields

$$T_r^T Z T_r =: M = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{12}^T & M_{22} & M_{23} \\ M_{13}^T & M_{23}^T & M_{33} \end{bmatrix}$$

where (using (A.12))

$$
\begin{aligned}
M_{11} &= \begin{bmatrix}\mathbf{I}_r & \mathbf{0}_{r\times k} & \mathbf{0}_{r\times(n-k)}\end{bmatrix} Z \begin{bmatrix}\mathbf{I}_r \\ \mathbf{0}_{k\times r} \\ \mathbf{0}_{(n-k)\times r}\end{bmatrix} = \Delta_r \in \mathbb{R}^{r\times r} \\
M_{12} &= \begin{bmatrix}\mathbf{I}_r & \mathbf{0}_{r\times k} & \mathbf{0}_{r\times(n-k)}\end{bmatrix} Z \begin{bmatrix}\mathbf{0}_{r\times k} \\ B_{s1} \\ B_{s2}\end{bmatrix} = (B_r^T S - C_r^T) \begin{bmatrix}B_{s1} \\ B_{s2}\end{bmatrix} \\
&= (B_{r1}^T (C_{s1} B_{s1}^{-1} + B_{s1}^{-T} B_{s2}^T S_1 B_{s2} B_{s1}^{-1}) - B_{r2}^T S_1 B_{s2} B_{s1}^{-1} - C_{r1}^T) B_{s1} + \\
&\quad (-B_{r1}^T B_{s1}^{-T} B_{s2}^T S_1 + B_{r2}^T S_1 - C_{r2}^T) B_{s2} \\
&= B_{r1}^T C_{s1} - C_{r1}^T B_{s1} - C_{r2}^T B_{s2} \\
&= B_r^T C_s - C_r^T B_s \in \mathbb{R}^{r\times k} \\
M_{13} &= \begin{bmatrix}\mathbf{I}_r & \mathbf{0}_{r\times k} & \mathbf{0}_{r\times(n-k)}\end{bmatrix} Z \begin{bmatrix}\mathbf{0}_{r\times(n-k)} \\ \mathbf{0}_{k\times(n-k)} \\ \mathbf{I}_{n-k}\end{bmatrix} = \begin{bmatrix}B_{r1}^T & B_{r2}^T\end{bmatrix} \begin{bmatrix}-B_{s1}^{-T} B_{s2}^T S_1 \\ S_1\end{bmatrix} + C_{r2}^T \\
&= (-B_{r1}^T B_{s1}^{-T} B_{s2}^T + B_{r2}^T) S_1 + C_{r2}^T \in \mathbb{R}^{r\times(n-k)} \\
M_{22} &= \begin{bmatrix}\mathbf{0}_{k\times r} & B_{s1}^T & B_{s2}^T\end{bmatrix} Z \begin{bmatrix}\mathbf{0}_{r\times k} \\ B_{s1} \\ B_{s2}\end{bmatrix} = B_s^T (A_0 S + S A_0^T) B_s \\
&= B_{s1}^T (A_{11} (C_{s1} B_{s1}^{-1} + B_{s1}^{-T} B_{s2}^T S_1 B_{s2} B_{s1}^{-1}) - A_{12} S_1 B_{s2} B_{s1}^{-1} + \\
&\qquad (C_{s1} B_{s1}^{-1} + B_{s1}^{-T} B_{s2}^T S_1 B_{s2} B_{s1}^{-1}) A_{11}^T - B_{s1}^{-T} B_{s2}^T S_1 A_{12}^T) B_{s_1} + \\
&\quad B_{s2}^T (A_{21} (C_{s1} B_{s1}^{-1} + B_{s1}^{-T} B_{s2}^T S_1 B_{s2} B_{s1}^{-1}) - A_{22} S_1 B_{s2} B_{s1}^{-1} - \\
&\qquad S_1 B_{s2} B_{s1}^{-1} A_{11}^T + S_1 A_{12}^T) B_{s1} + \\
&\quad B_{s1}^T (-A_{11} B_{s1}^{-T} B_{s2}^T S_1 + A_{12} S_1 + \\
&\qquad (C_{s1} B_{s1}^{-1} + B_{s1}^{-T} B_{s2}^T S_1 B_{s2} B_{s1}^{-1}) A_{21}^T - B_{s1}^{-T} B_{s2}^T S_1 A_{22}^T) B_{s2} + \\
&\quad B_{s2}^T ((A_{22} - A_{21} B_{s1}^{-T} B_{s2}^T) S_1 + S_1 (A_{22}^T - B_{s2} B_{s1}^{-1} A_{21}^T)) B_{s2} \\
&= B_{s1}^T (A_{11} C_{s1} B_{s1}^{-1} + C_{s1} B_{s1}^{-1} A_{11}^T) B_{s1} + \\
&\quad B_{s2}^T A_{21} C_{s1} B_{s1}^{-1} B_{s1} + B_{s1}^T C_{s1} B_{s1}^{-1} A_{21}^T B_{s2} \\
&= C_{s1}^T A_{11}^T B_{s1} + C_{s1}^T A_{21}^T B_{s2} + B_{s1}^T A_{11} C_{s1} + B_{s2}^T A_{21} C_{s1} \\
&= C_s^T A_0^T B_s + B_s^T A_0 C_s \in \mathbb{R}^{k\times k} \\
M_{23} &= \begin{bmatrix}\mathbf{0}_{k\times r} & B_{s1}^T & B_{s2}^T\end{bmatrix} Z \begin{bmatrix}\mathbf{0}_{r\times(n-k)} \\ \mathbf{0}_{k\times(n-k)} \\ \mathbf{I}_{n-k}\end{bmatrix} \\
&= B_{s1}^T (-A_{11} B_{s1}^{-T} B_{s2}^T S_1 + A_{12} S_1 + \\
&\qquad (C_{s1} B_{s1}^{-1} + B_{s1}^{-T} B_{s2}^T S_1 B_{s2} B_{s1}^{-1}) A_{21}^T - B_{s1}^{-T} B_{s2}^T S_1 A_{22}^T) + \\
&\quad B_{s2}^T ((-A_{21} B_{s1}^{-T} B_{s2}^T S_1 + A_{22} S_1 - S_1 B_{s2} B_{s1}^{-1}) A_{21}^T + S_1 A_{22}^T) \\
&= (B_{s1}^T A_{12} + B_{s2}^T A_{22} - (B_{s1}^T A_{11} + B_{s2}^T A_{21}) B_{s1}^{-T} B_{s2}^T) S_1 + B_{s1}^T C_{s1} B_{s1}^{-1} A_{21}^T \\
&= (B_{s1}^T A_{12} + B_{s2}^T A_{22} - (B_{s1}^T A_{11} + B_{s2}^T A_{21}) B_{s1}^{-T} B_{s2}^T) S_1 + C_{s1}^T A_{21}^T \in \mathbb{R}^{k\times(n-k)}
\end{aligned}
$$

$$
\begin{aligned}
M_{33} &= \begin{bmatrix}\mathbf{0}_{(n-k)\times r} & \mathbf{0}_{(n-k)\times k} & \mathbf{I}_{n-k}\end{bmatrix} Z \begin{bmatrix}\mathbf{0}_{r\times(n-k)} \\ \mathbf{0}_{k\times(n-k)} \\ \mathbf{I}_{n-k}\end{bmatrix} \\
&= (A_{22} - A_{21}B_{s1}^{-T}B_{s2}^T)S_1 + S_1(A_{22} - A_{21}B_{s1}^{-T}B_{s2}^T)^T \in \mathbb{R}^{(n-k)\times(n-k)}.
\end{aligned}
$$

Comparing these results with (A.1), considering that $M = -T_r^T Z T_r = -T_r^T K_r K_r^T T_r$ (see (A.10)), and writing $K_1 := T_r^{-T} K_r$ with $K_1 = \begin{bmatrix}L_1 \\ G_1\end{bmatrix} \in \mathbb{R}^{(d|n-k)\times d}$ shows that they are equivalent to another system of Lur'e equations:

$$
\begin{aligned}
A_1 S_1 + S_1 A_1^T &= -G_1 G_1^T \\
S_1 B_1 - C_1 &= -G_1 L_1^T \\
\Delta_1 &= -L_1 L_1^T
\end{aligned}
$$

with

$$
\begin{aligned}
A_1 &= A_{22} - A_{21}B_{s1}^{-T}B_{s2}^T \\
B_1 &= \begin{bmatrix}-B_{s2}B_{s1}^{-1}B_{r1} + B_{r2} & A_{12}^T B_{s1} + A_{22}^T B_{s2} - B_{s2}B_{s1}^{-1}(A_{11}^T B_{s1} + A_{21}^T B_{s2})\end{bmatrix} \\
C_1 &= \begin{bmatrix}-C_{r2} & -A_{21}C_{s1}\end{bmatrix} \\
\Delta_1 &= \begin{bmatrix}\Delta_r & B_r^T C_s - C_r^T B_s \\ C_s^T B_r - B_s^T C_r & C_s^T A_0^T B_s + B_s^T A_0 C_s\end{bmatrix}.
\end{aligned}
$$

This system can be solved recursively, either via the Riccati equation (A.2) or via another "reduction" step, depending on whether these equations are singular or not. Solving the reduced system of equations yields matrices S_1 and K_1, $L_1 = \begin{bmatrix}L_r \\ L_s\end{bmatrix} \in \mathbb{R}^{(r|k)\times d}$. Then one easily verifies that the solution of the original system (A.1) is given by

$$
\begin{aligned}
\Sigma = T_r^{-T}\begin{bmatrix}\mathbf{I}_r & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & C_{s1}^T B_{s1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & S_1\end{bmatrix} T_r^{-1} &= \begin{bmatrix}\mathbf{I}_r & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & S_{11} & S_{12} \\ \mathbf{0} & S_{12}^T & S_1\end{bmatrix}, \\
K = \begin{bmatrix}\mathbf{0}_{k\times d} \\ K_r\end{bmatrix} = \begin{bmatrix}\mathbf{0}_{k\times d} \\ T_r^{-T}K_1\end{bmatrix} &= \begin{bmatrix}\mathbf{0}_{k\times d} \\ L_r \\ B_{s1}^{-T}(L_s - B_{s2}^T G_1) \\ G_1\end{bmatrix}.
\end{aligned}
$$

Since every solution of the reduced system corresponds to a solution of the original system (A.1) and vice versa, if the former's symmetric and positive definite solution is unique, this it true for the latter's solution, too.

Abbreviations

GLE generalized Langevin equation

VACF velocity auto-covariance function

Publication

- Niklas Bockius, Jeanine Shea, Gerhard Jung, Friederike Schmid, and Martin Hanke. "Model reduction techniques for the computation of extended Markov parameterizations for generalized Langevin equations." In: *Journal of Physics: Condensed Matter* 33.21 (May 2021), p. 14. DOI: 10.1088/1361-648x/abe6df

Bibliography

[1] José. I. Aliaga, Daniel. L. Boley, Roland W. Freund, and Vincente Hernández. "A Lanczos-type method for multiple starting vectors." In: *Mathematics of Computation* 69 (1999), pp. 1577–1601. DOI: 10.1090/S0025-5718-99-01163-1.

[2] Greg S. Ammar, Wijesuriya P. Dayawansa, and Clyde F. Martin. "Exponential interpolation: Theory and numerical algorithms." In: *Applied Mathematics and Computation* 41.3 (1991), pp. 189–232. DOI: 10.1016/0096-3003(91)90025-I.

[3] Brian D. O. Anderson. "A system theory criterion for positive real matrices." In: *SIAM Journal on Control* 5.2 (1967), pp. 171–182. DOI: 10.1137/0305011.

[4] Brian D. O. Anderson and Sumeth Vongpanitlerd. *Network analysis and synthesis. A modern systems theory approach.* 2nd ed. Dover Publications, 2006.

[5] Joshua A. Anderson, Jens Glaser, and Sharon C. Glotzer. "HOOMD-blue: A Python package for high-performance molecular dynamics and hard particle Monte Carlo simulations." In: *Computational Materials Science* 173 (2020), p. 109363. DOI: 10.1016/j.commatsci.2019.109363.

[6] Athanasios C. Antoulas. *Approximation of large-scale dynamical systems.* Advances in Design and Control. Philadelphia: Society for Industrial and Applied Mathematics, 2005. DOI: 10.1137/1.9780898718713.

[7] Peter Benner, Volker Mehrmann, and Hongguo Xu. "A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils." In: 78 (1998), pp. 329–358. DOI: 10.1007/s002110050315.

[8] Gregory Beylkin and Lucas Monzón. "On approximation of functions by exponential sums." In: *Applied and Computational Harmonic Analysis* 19.1 (2005), pp. 17–48. DOI: 10.1016/j.acha.2005.01.003.

[9] Salomon Bochner. "Monotone Funktionen, Stieltjessche Integrale und harmonische Analyse." In: *Journal of Physics: Condensed Matter* 108 (1933), pp. 378–410. DOI: 10.1007/BF01452844.

[10] Niklas Bockius, Jeanine Shea, Gerhard Jung, Friederike Schmid, and Martin Hanke. "Model reduction techniques for the computation of extended Markov parameterizations for generalized Langevin equations." In: *Journal of Physics: Condensed Matter* 33.21 (May 2021), p. 14. DOI: 10.1088/1361-648x/abe6df.

[11] Harald Cramér and Malcolm R. Leadbetter. *Stationary and related stochastic processes: sample function properties and their applications.* 1, corr. print. New York [u.a.]: Wiley, 1968, p. 348.

[12] R. F. Curtain. “Old and new perspectives on the positive-real lemma in systems and control theory.” In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 79.9 (1999), pp. 579–590. DOI: 10.1002/(SICI)1521-4001(199909)79:9<579::AID-ZAMM579>3.0.CO;2-8.

[13] Nadiia Derevianko and Gerlind Plonka. “Exact reconstruction of extended exponential sums using rational approximation of their Fourier coefficients.” In: (2021). DOI: 10.48550/arXiv.2103.07743.

[14] Roland W. Freund. “Computation of matrix-valued formally orthogonal polynomials and applications.” In: *Journal of Computational and Applied Mathematics* 127 (2001), pp. 173–199. DOI: 10.1016/S0377-0427(00)00505-7.

[15] Roland W. Freund. “Krylov-subspace methods for reduced-order modeling in circuit simulation.” In: *Journal of Computational and Applied Mathematics* 123.1 (2000), pp. 395–421. DOI: 10.1016/S0377-0427(00)00396-4.

[16] Roland W. Freund, Martin H. Gutknecht, and Noël M. Nachtigal. “An implementation of the look-ahead Lanczos Algorithm for non-hermitian matrices.” In: *SIAM Journal on Scientific Computing* 14.1 (1993), pp. 137–158. DOI: 10.1137/0914009.

[17] John Fricks, Lingxing Yao, Timothy C. Elston, and M. Gregory Forest. “Time-domain methods for diffusive transport in soft matter.” In: *SIAM Journal on Applied Mathematics* 69.5 (2009), pp. 1277–1308. DOI: 10.1137/070695186.

[18] William B. Gragg. “Matrix interpretations and applications of the continued fraction algorithm.” In: *Rocky Mountain Journal of Mathematics* 4.2 (1974), pp. 213–226. DOI: 10.1216/RMJ-1974-4-2-213.

[19] Gustaf Gripenberg, Stig-Olof Londen, and Olof Staffans. *Volterra integral and functional equations*. Vol. 34. Encyclopedia of mathematics and its applications. Cambridge et al.: Cambridge University Press, 1990. DOI: 10.1017/CBO9780511662805.

[20] Martin Hanke. “Mathematical analysis of some iterative methods for the reconstruction of memory kernels.” In: *Electronic Transactions on Numerical Analysis* 54 (2021), pp. 483–498. DOI: 10.1553/etna_vol54s483.

[21] Martin Hanke. *The second fluctuation-dissipation theorem for the generalized Langevin equation*. 2025. DOI: 10.48550/arXiv.2507.17350. In review.

[22] Nicholas J. Higham. *Functions of matrices. Theory and computation*. Philadelphia: Society for Industrial and Applied Mathematics, 2008. DOI: 10.1137/1.9780898717778.

[23] Christel Hohenegger and Scott A. McKinley. “Reconstructing complex fluid properties from the behavior of fluctuating immersed particles.” In: *SIAM Journal on Applied Mathematics* 78 (2018), pp. 2200–2226. DOI: 10.1137/17M1131660.

[24] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. 2nd ed. Cambridge: Cambridge University Press, 2012. DOI: 10.1017/CBO9781139020411.

[25] Gerhard Jung, Martin Hanke, and Friederike Schmid. “Iterative reconstruction of memory kernels.” In: *Journal of Chemical Theory and Computation* 13.6 (2017), pp. 2481–2488. DOI: 10.1021/acs.jctc.7b00274.

[26] Achim Klenke. *Probability theory*. 3rd ed. Berlin and Heidelberg: Springer Spektrum, 2020. XIV, 716. DOI: 10.1007/978-3-030-56402-5.

[27] Louis Komzsik. *The Lanczos method. Evolution and application*. Philadelphia: Society for Industrial and Applied Mathematics, 2003. DOI: 10.1137/1.9780898718188.

[28] R. Kubo. "The fluctuation-dissipation theorem." In: *Reports on Progress in Physics* 29.1 (Jan. 1966), pp. 255–284. DOI: 10.1088/0034-4885/29/1/306.

[29] Joseph La Salle and Solomon Lefschetz. *Stability by Liapunov's Direct Method with Applications*. New York and London: Academic Press Inc., 1961.

[30] Oliver F. Lange and Helmut Grubmüller. "Collective Langevin dynamics of conformational motions in proteins." In: *The Journal of Chemical Physics* 124.21 (June 2006), p. 214903. DOI: 10.1063/1.2199530.

[31] Alan J. Laub. "A schur method for solving algebraic Riccati equations." In: *1978 IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes* (Oct. 1978), pp. 60–65. DOI: 10.1109/CDC.1978.267893.

[32] Huan Lei, Nathan A. Baker, and Xiantao Li. "Data-driven parameterization of the generalized Langevin equation." In: *Proceedings of the National Academy of Sciences* 113.50 (Dec. 2016), pp. 14183–14188. DOI: 10.1073/pnas.1609587113.

[33] Peter Linz. *Analytical and numerical methods for Volterra equations*. Society for Industrial and Applied Mathematics, 1985. DOI: 10.1137/1.9781611970852.

[34] Lina Ma, Xiantao Li, and Chun Liu. "The derivation and approximation of coarse-grained dynamics from Langevin dynamics." In: *The Journal of Chemical Physics* 145.20 (Nov. 2016), p. 204117. DOI: 10.1063/1.4967936.

[35] Scott A. McKinley and Hung D. Nguyen. "Anomalous Diffusion and the Generalized Langevin Equation." In: *SIAM Journal on Mathematical Analysis* 50.5 (2018), pp. 5119–5160. DOI: 10.1137/17M115517X.

[36] Hugues Meyer, Philipp Pelagejcev, and Tanja Schilling. "Non-Markovian out-of-equilibrium dynamics: A general numerical procedure to construct time-dependent memory kernels for coarse-grained observables." In: *Europhysics Letters* 128.4 (Jan. 2020), 40001-p1–40001-p7. DOI: 10.1209/0295-5075/128/40001.

[37] Thomas P. Minka. *Old and new matrix algebra useful for statistics*. 2000. URL: https://tminka.github.io/papers/matrix/minka-matrix.pdf (manuscript, downloaded on 19th Dec. 2023).

[38] Yuji Nakatsukasa, Olivier Sète, and Lloyd N. Trefethen. "The AAA algorithm for rational approximation." In: *SIAM Journal on Scientific Computing* 40.3 (2018), A1494–A1522. DOI: 10.1137/16M1106122.

[39] Bernt K. Øksendal. *Stochastic differential equations. An introduction with applications*. 5th ed. Berlin and Heidelberg: Springer Spektrum, 1998. DOI: 10.1007/978-3-662-03620-4.

[40] Grigorios A. Pavliotis. *Stochastic processes and applications. Diffusion processes, the Fokker-Planck and Langevin equations*. Vol. 60. Texts in applied mathematics. New York: Springer, 2014. DOI: 10.1007/978-1-4939-1323-7.

[41] Steve Plimpton. “Fast Parallel Algorithms for Short-Range Molecular Dynamics.” In: *Journal of Computational Physics* 117.1 (1995), pp. 1–19. DOI: 10.1006/jcph.1995.1039.

[42] Gerlind Plonka, Daniel Potts, Gabriele Steidl, and Manfred Tasche. *Numerical Fourier analysis*. 1st ed. Applied and numerical harmonic analysis. Cham: Birkhäuser Cham, 2018. DOI: 10.1007/978-3-030-04306-3.

[43] Gerlind Plonka and Manfred Tasche. “Prony methods for recovery of structured functions.” In: *GAMM-Mitteilungen* 37.2 (2014), pp. 239–258. DOI: 10.1002/gamm.201410011.

[44] Daniel Potts and Manfred Tasche. “Parameter estimation for nonincreasing exponential sums by Prony-like methods.” In: *Linear Algebra and its Applications* 439 (2013), pp. 1024–1039. DOI: 10.1016/j.laa.2012.10.036.

[45] R. Roy and Thomas Kailath. “ESPRIT-estimation of signal parameters via rotational invariance techniques.” In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.7 (1989), pp. 984–995. DOI: 10.1109/29.32276.

[46] Walter Rudin. *Fourier Analysis on Groups*. New York et al.: Wiley-Interscience, 1990. DOI: 10.1002/9781118165621.

[47] Hyun Kyung Shin, Changho Kim, Peter Talkner, and Eok Kyun Lee. “Brownian motion from molecular dynamics.” In: *Chemical Physics* 375.2 (2010). Stochastic processes in Physics and Chemistry (in honor of Peter Hänggi), pp. 316–326. DOI: 10.1016/j.chemphys.2010.05.019.

[48] Eduardo D. Sontag. *Mathematical control theory. Deterministic finite dimensional systems*. 2nd ed. Texts in applied mathematics. New York, Berlin, and Heidelberg: Springer, 1998. DOI: 10.1007/978-1-4612-0577-7.

[49] Qinghong Wang, Jason L. Speyer, and Haim Weiss. “System characterization of positive real conditions.” In: *29th IEEE Conference on Decision and Control* 1 (1990), pp. 348–353. DOI: 10.1109/CDC.1990.203610.

[50] W. M. Wonham. “On a matrix Riccati equation of stochastic control.” In: *SIAM Journal on Control* 6.4 (1968), pp. 681–697. DOI: 10.1137/0306044.

Lebenslauf

Aus Datenschutzgründen wurde der Lebenslauf in der digitalen Version entfernt.