Efficient algorithms for simulation and analysis of many-body systems

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

am Fachbereich Mathematik und Informatik der Freien Universität Berlin

vorgelegt von

Moritz Hoffmann

Berlin, 2022

Erstgutachter:
Prof. Dr. Frank Noé
Freie Universität Berlin
Fachbereich Mathematik und Informatik
Arnimallee 12, 14195 Berlin

Zweitgutachterin:
PD. Dr. Jutta Rogal
Freie Universität Berlin
Fachbereich Physik
Arnimallee 14
14195 Berlin

Tag der Disputation: 28.10.2022

Contents

List of Figures

List of Tables

List of publications

First-author and shared first-author publications.

[P1] **M. Hoffmann**, C. Fröhner, F. Noé. "ReaDDy 2: Fast and flexible software framework for interacting-particle reaction dynamics". In: *PLoS Comput. Biol.* (2019). URL: `https://doi.org/10.1371/journal.pcbi.1006830`.

[P2] **M. Hoffmann, C. Fröhner**, F. Noé. "Reactive SINDy: Discovering governing reactions from concentration data". In: *J. Chem. Phys.* (2019). URL: `https://doi.org/10.1063/1.5066099`.

[P3] **M. Hoffmann**, F. Noé. "Generating valid Euclidean distance matrices". In: *arXiv preprint* (2019). URL: `https://arxiv.org/abs/1910.03131`.

[P4] **M. Hoffmann**, M. K. Scherer, T. Hempel, A. Mardt, B. de Silva, B. E. Husic, S. Klus, H. Wu, N. Kutz, S. L. Brunton, F. Noé. "Deeptime: a Python library for machine learning dynamical models from time series data". In: *Mach. learn.: sci. technol.* (2022). URL: `https://doi.org/10.1088/2632-2153/ac3de0`.

Other publications.

[P5] **M. K. Scherer, B. Trendelkamp-Schroer**, F. Paul, G. Pérez-Hernández, M. Hoffmann, N. Plattner, C. Wehmeyer, J.-H. Prinz, F. Noé. "PyEMMA 2: A software package for estimation, validation, and analysis of Markov models". In: *J. Chem. Theory Comput.* (2015). URL: `https://dx.doi.org/10.1021/acs.jctc.5b00743`.

[P6] **M. K. Scherer, B. E. Husic**, M. Hoffmann, F. Paul, H. Wu, F. Noé. "Variational selection of features for molecular kinetics". In: *J. Chem. Phys.* (2019). URL: `https://doi.org/10.1063/1.5083040`.

[P7] **K. Shmilovich**, M. Stieffenhofer, N. E. Charron, M. Hoffmann. "Temporally coherent backmapping of molecular trajectories from coarse-grained to atomistic resolution". In: *arXiv preprint* (2022). URL: `https://arxiv.org/abs/2205.05213`.

First authors are highlighted in bold font. This thesis contains parts of publications [P1], [P2], and [P4].

Chapter 1

Introduction

This thesis introduces methods to efficiently generate and analyze time series data of many-body systems. While we have a strong focus on biomolecular processes, the presented methods can also be applied more generally. Due to limitations of microscope resolution in both space and time, biomolecular processes are especially hard to observe experimentally. Computer models offer an opportunity to work around these limitations. However, as these models are bound by computational effort, careful selection of the model as well as its efficient implementation play a fundamental role in their successful sampling and/or estimation.

Especially for high levels of resolution, computer simulations can produce vast amounts of high-dimensional data and in general it is not straightforward to visualize, let alone to identify the relevant features and processes. To this end, we cover tools for projecting time series data onto important processes, finding over time geometrically stable features in observable space, and identifying governing dynamics. We introduce the novel software library `deeptime` with two main goals: (1) making methods which were developed in different communities (such as molecular dynamics and fluid dynamics) accessible to a broad user base by implementing them in a general-purpose way, and (2) providing an easy to install, extend, and maintain library by employing a high degree of modularity and introducing as few hard dependencies as possible. We demonstrate and compare the capabilities of the provided methods based on numerical examples.

Subsequently, the particle-based reaction-diffusion simulation software package `ReaDDy2` is introduced. It can simulate dynamics which are more complicated than what is usually analyzed with the methods available in `deeptime`. It is a significantly more efficient, feature-rich, flexible, and user-friendly version of its predecessor `ReaDDy`. As such, it enables—at the simulation model's resolution—the possibility to study larger systems and to cover longer timescales. In particular, `ReaDDy2` is capable of modeling complex processes featuring particle crowding, space exclusion, association and dissociation events, dynamic formation and dissolution of particle geometries on a mesoscopic scale. The validity of the `ReaDDy2` model is asserted by several numerical studies which are compared to analytically obtained results, simulations from other packages, or literature data.

Finally, we present `reactive SINDy`, a method that can detect reaction networks from concentration curves of chemical species. It extends the sparse identification of nonlinear dynamics (SINDy) method—contained in `deeptime`—by introducing coupling terms over a system of ordinary differential equations in an ansatz reaction space. As such, it transforms an ordinary linear regression problem to a linear tensor regression. The method employs a sparsity-promoting regularization which leads to especially simple and interpretable models. We show in biologically motivated example systems that the method is indeed capable of detecting the correct underlying reaction dynamics and that the sparsity regularization plays a key role in pruning otherwise spuriously detected reactions.

1.1 Motivation

To set this thesis into context, let us highlight why efficient computer simulations of molecular processes and subsequent data analysis are important tools to gain insights into their properties.

Molecular processes themselves are fundamental to life. For the processes' general understanding it is of importance to identify and describe their structural and dynamical properties. The involved—sometimes intracellular—components can be resolved via microscopy techniques such as STORM with a resolution of 20 nm to 50 nm [1], cryo-EM with a resolution of 3 Å to 15 Å [2, 3], or X-ray diffraction with a resolution of < 1 nm (which was first used to resolve a protein structure in the 1950s [4]). This enables the construction of atomistic models. However, the discovery of mechanisms and temporal correlations often requires not only high spatial resolution but detailed spatiotemporal data [5, 6].

Microscopy techniques that yield high temporal resolution for biological particles (e.g., PALM [7, 8] or LLSM [9]) on the other hand possess a significantly lower spatial resolution. While some of the lower spatial resolution can be gained back by longer exposure time, this leads to radiation damage and a lower temporal resolution; furthermore, the techniques are still bounded by the diffraction limit. The diffraction limit is inversely proportional to the wavelength used in the respective microscopy technique. A logical step might be to use shorter wavelengths, however, the shorter the wavelength, the greater the radiation damage inflicted on biological particles for a given exposure time [10]. This prevents the use of short wavelengths for resolving biological processes when temporal resolution is important.

One example of a nontrivial biological process is the widely studied mitogen activated protein kinases (MAPK) pathway: a signaling cascade that is microscopically triggered by receptors on the surface of a cell and relays the signal to the cell's DNA [5, 11]. The molecule involved here, MAPK, plays a key role in cell proliferation, differentiation, inflammation, and apoptosis [12, 13]. Therefore, the behavior of these molecules is of great interest in, e.g., cancer research. It is possible to observe the MAPK pathway kinetics experimentally for single cells [14]. Many of such pathways involve G protein coupled receptors (GPCRs) [15] on the cell membrane triggering a localized synthesis of cyclic adenosine monophosphate (cAMP) as second messengers [16]—both of which are involved in various diseases [17–19].

Since spatial locality (e.g., inhomogeneous distribution of cAMP molecules [20, 21]) and effects such as crowding, space exclusion, association and dissociation of macromolecules as well as the complex and inhomogeneous topology of a cell play important roles in these processes, a high spatiotemporal resolution is required to understand the function of the involved particles. This was demonstrated in, e.g., Refs. 22 and 5.

A possible remedy for the fundamental limitations of microscopy can be found in computer models. They can, in principle, achieve arbitrary levels of resolution in both time and space. However, they are bounded by computational effort: since molecular processes often span multiple spatiotemporal scales (e.g., a MAPK pathway that is triggered on an atomic level on a cell's surface and leads to a macroscopic response), careful consideration has to go into the design and selection of models and simulation algorithms.

Since the advent of modern day computer hardware, computational models for physical processes spanning a vast range of different spatial resolutions have been developed. The highest resolution is provided by simulations operating on the quantum scale. They describe physical properties on a (sub-)atomic level. Models at this level of detail however are computationally expensive: a popular but still relatively low precision method is DFT [23, 24], which in most implementations (for B3LYP functionals) scales $\mathcal{O}(N^3)$ where N is the number of electrons in the system [25], quickly becoming infeasible for larger systems.

On a microscopic scale, a popular method is molecular dynamics (MD) with empirical force fields such as CHARMM [26] or Amber [27, 28] force fields for proteins or their generalized versions for small molecules [29, 30]. MD scales in computational complexity with $\mathcal{O}(N \log N)$, where N is the number of simulated atoms. Long-range electrostatic interactions are typically integrated using particle mesh Ewald [31] and dominate the computational complexity. As an intermediate, there are methods to locally back-introduce quantum scale effects into MD with the hybrid QM/MM model [32]. A typical integration time-step is in the range of femtoseconds so that rapid fluctuations of atoms can be captured, in particular of bound hydrogen atoms [33, 34]. However, many interesting biological processes happen on scales of microseconds to seconds or even slower—, e.g., (un-)folding and (un-)binding events of proteins [35, 36]. Therefore, the staggering amount of $5 \cdot 10^{14}$ simulation time steps is required to achieve 1 s of simulation time under a standard MD integration step of 2 fs. On very specialized and rare hardware it is possible to produce 85 µs of simulation data per day for a dihydrofolate reductase system containing 23558 atoms [37]; present-day GPU-parallelized software on high-end hardware is capable of producing hundreds of nanoseconds per day [38–41]: OpenMM produces 367 ns per day for the same system on a NVIDIA Titan X GPU [39].

In order to relate the computation times to biological systems, we consider the already mentioned MAPK protein: After removing an applied stimulus, the concentration of phosphorylated MAPK fluctuates on the scale of minutes [42]. Considering that each MAPK protein contains roughly 3000 atoms [43] and that multiple copies are required to reproduce the full system, adequate sampling of such a system goes well beyond the capabilities of MD on current hardware.

On the larger end of physical scales is the macroscopic scale. Something being "macroscopic" in itself is used differently in different fields and is often situational [44]; here, the term is used with respect to objects that are visible to the naked eye. Consequently, simulations on a macroscopic scale are too coarse to adequately describe phenomena such as the MAPK pathway.

Scales between the micro- and macroscopic scale are referred to as mesoscopic. Whereas simulation models on the mesoscopic scale lack the level of detail of microscopic (e.g., atomistic) simulations, the computational effort required to produce mesoscopic data is reduced. The reduction can be described in terms of the scaling of computational effort required for a specific system size per time step but also in the time step itself, as the fast vibrations of individual atoms are typically no longer resolved. This makes mesoscopic models suitable and feasible candidates for describing phenomena such as signal cascades, concentration gradients, or membrane interactions in terms of spatiotemporal resolution.

1.2 Chapter overview

In this thesis, we introduce methods for the analysis of data that is produced by the simulation of dynamical models as well as for the simulation of multi-body systems at a mesoscopic scale. Both the data analysis and the simulation are focused on biological or biologically motivated systems. A key concept for the remainder of this thesis are Markov processes. These are chains of events in which each individual event depends only on its predecessor and has no dependency on events further in the past (known as the Markov property). The concept of Markovianity is introduced more formally in Chapter 2 and all of the following simulation as well as analysis methods are based thereon.

Figure 1.1 contains a flowchart style overview of chapters, sections, and their dependencies.

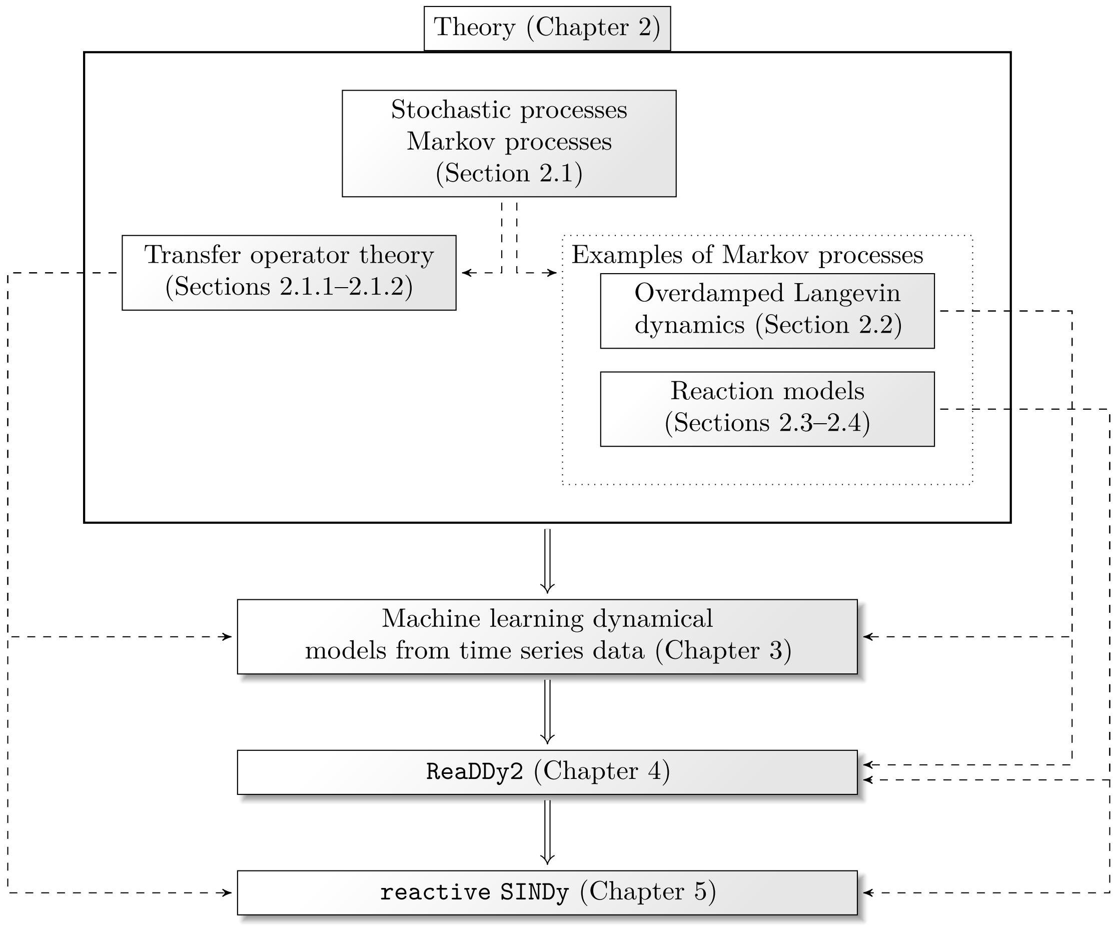


Figure 1.1: **Chapter and section overview.** Flowchart of this thesis' chapters and theory sections. Implication arrows ($\Rightarrow$) show the chapter progression, dashed arrows denote that a certain component is used in a later section or chapter, i.e., $A \dashrightarrow B$ means A is used in B.

From the top, we give a formal introduction to stochastic processes and more specifically Markov processes in Chapter 2, but also cover transfer operator theory (Sections 2.1.1–2.1.2) as well as several examples of Markov processes which are of relevance for this thesis. Transfer operators are generally infinite-dimensional but *linear* operators describing the temporal evolution of observables and/or densities under Markovian dynamics. They are the basis for the methods introduced in Chapter 3 as well as Chapter 5. One particular family of Markov processes is given by overdamped Langevin dynamics, which models the movement of interacting particles in space and is introduced in Section 2.2. These dynamics are used for some of the numerical studies in Chapter 3 but are of particular relevance for the `ReaDDy2` interacting-particle reaction diffusion integrator presented in Chapter 4. Furthermore, we introduce several reaction models, one of which is implemented in Chapter 4 and others are used in Chapter 5 for numerical studies. Chapter 3 is based on Ref. P4, Chapter 4 is based on Ref. P1, and Chapter 5 is based on Ref. P2.

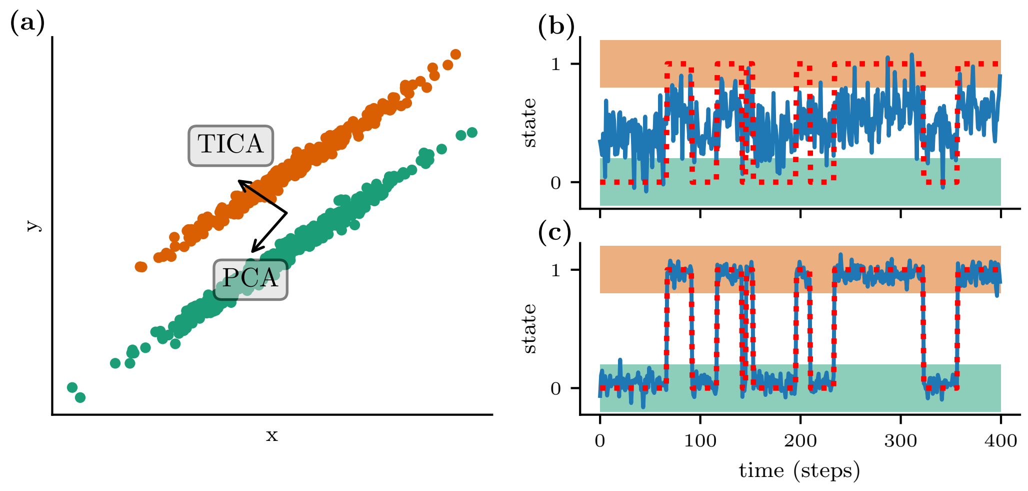


Figure 1.2: **Projection of time series data.** We show a two-dimensional jump process between two ellipsoidal distributions. **(a)** The two distributions are in orange and green, respectively. We project the data into one dimension using principle component analysis (PCA) and time-lagged independent component analysis (TICA), respectively. The arrows indicate the projection axes. **(b), (c)** PCA and TICA projection in blue, respectively. The red dotted line is the ground truth jump process.

1.2.1 Machine learning dynamical models from time series data

When dealing with particle-based simulations describing biological phenomena, the obtained amount of data can easily become an issue: assuming that only particle positions are recorded, the result of such simulations are trajectories with frames of dimension $N \cdot d$, where N is the number of particles and d the dimension of the system (typically $d = 2$ or $d = 3$). Sometimes and in particular for systems involving reactions, N depends on time, i.e., $N = N(t)$. The number of particles can easily be in the tens of thousands, making for a very high-dimensional space. Also, if a system is modeled with an Eulerian description (e.g., via the finite element method or similar), the resulting dimension may be very high, depending on the resolution of spatial meshes and ansatz spaces. The task of finding mechanisms and temporal structures from visualizations alone may become difficult if not impossible, especially when dealing with systems whose state is three- or higher-dimensional.

To get an idea of processes and structures, one might try to cluster together similar frames. However, clustering becomes infeasible with higher dimensions, as the distance between two points under increasing dimensions quickly becomes less and less interpretable [45]. A possible remedy is to project the data into a lower-dimensional space, which should preserve the “important” structure. A popular data-driven method for dimension reduction is principle component analysis (PCA) [46, 47]. However, PCA makes no use of one crucial aspect of such simulation data: it is a time series. Instead, PCA projects onto the axes which maximize variance (or equivalently: minimize reconstruction error). Based on this projection, the relevant process might not be picked up or even be lost.

In Figure 1.2 exactly such a case is presented. We show two ellipsoidal distributions between which a hidden process jumps back and forth. The transitioning is a slow process (meaning we assume it to be important, see Section 3.3), which occurs with a probability of 3% per step. The PCA projection uses the direction of the largest variance for explaining the data and almost completely disregards the slowly transitioning jump process between ellipsoidal distributions (Figure 1.2a). This leads to a loss of most of the signal (Figure 1.2b). A method which takes into account the time series aspect of the given data is time-lagged independent component analysis (TICA) and is presented in Figure 1.2c. Indeed, the slow (important) signal of transitioning between the two states is nicely captured in the projection.

TICA belongs to a family of methods based on approximating transfer operators, which are introduced in Sections 2.1.1–2.1.2. As descriptions of the temporal evolution of probability distributions or observables, transfer operators explicitly take into account the time-dependent aspect of stochastic processes.

Approximations of transfer operators typically try to find 'slow' processes like the jumping between state 0 (green) and state 1 (orange) respective distributions in Figure 1.2. For biologically motivated systems, a slow process could correspond to, e.g., the folding and unfolding or binding and unbinding of proteins. A fast process, on the other hand, could be the vibrations of hydrogen atoms in a protein, which are essentially noise and mostly uninteresting for the thermodynamic, kinetic, and mechanistic properties of a protein.

In Chapter 3 we introduce the open-source general-purpose Python software library `deeptime` for time series analysis. It features various tools based on and around transfer operator approximations (see Sections 3.3–3.4). In particular, it contains conventional linear learning methods such as TICA, but also Markov state models (MSMs), hidden Markov models (HMMs), and other transfer operator methods (see Section 3.3 for a list). Besides linear learning methods, `deeptime` also provides kernel and deep learning approaches such as kernel CCA [48] and VAMPNets [49].

In contrast to other time series analysis libraries, `deeptime` is designed for providing kinetic models rather than, e.g., forecasting, and is community-agnostic. It explicitly aims to make methods which are used and developed in different communities available to a broad user base by implementing them in a general-purpose way. For example, the time-lagged independent component analysis (TICA) method is mostly used in the MD community and can also be found in libraries like PyEMMA [P5] or MSMBuilder [50]; these, however, are tightly geared towards MD data. We highlight the differences and similarities of the methods from a theoretical point of view as well as by numerical experiments (Section 3.3). Furthermore, the library is designed to be easy to install, extend, and maintain. This is achieved by having a very low amount of hard dependencies as well as employing a high degree of modularity.

In order to demonstrate and experiment with the implemented methods, `deeptime` offers a set of benchmark datasets. To give some examples: it features a two-dimensional position-based fluids implementation [51], which is a method from computer graphics for the real-time simulation of fluids, an implementation of the Bickley jet [52, 53] which is an idealized model for stratospheric flow, and several datasets which are based on the simulation of particles in a potential landscape (which might also be time-dependent) using overdamped Langevin dynamics (see Section 2.2). In particular, all datasets are given via their generators and dynamics, meaning the binary size is kept small and users have the opportunity to study the effect of their parameters on the dynamics and methods.

While most methods in `deeptime` (version 0.4.1) focus on the estimation of transfer operators, there is the exception of the sparse identification of nonlinear dynamics (SINDy) method [54] (cf. Section 3.5). SINDy also has a mathematical connection to transfer operators, but instead of trying to approximate the propagators of observables and probability densities with respect to the system's slow processes, it tries to find an approximation of the corresponding generator. To that end, SINDy parsimoniously identifies the governing dynamics as coefficients in a set of ordinary differential equations (ODEs) by performing linear regression with a sparsity-encouraging

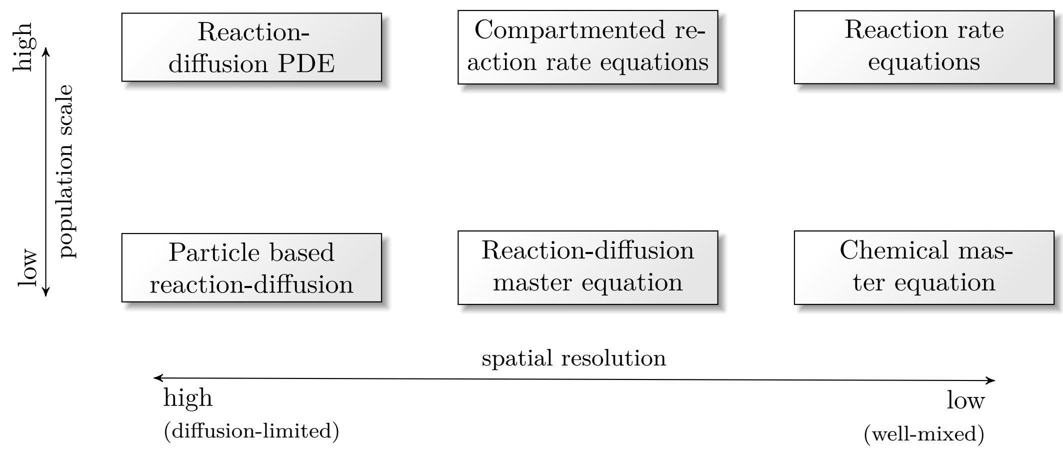


Figure 1.3: **Reaction-diffusion model overview.** Different models of reaction-diffusion processes, organized horizontally according to the population scale (where high populations, i.e., high particle numbers, are often represented as densities instead of discrete entities) and vertically according to their spatial resolution, in particular the ratio of encounter rate over formation rate when considering individual reactions (well-mixed corresponding to a high ratio, diffusion-limited corresponding to a low ratio).

regularization term. The sparsity fosters simplicity of the estimated model, which not only regularizes the solution and can prevent overfitting but also vastly improves interpretability. Following the rule of parsimony (or Occam's razor), this leads to models which are more likely to be related to the true underlying physics and/or chemistry of the observed process.

1.2.2 ReaDDy2

The methods presented in Chapter 3 are typically used for approaches with microscopic resolution such as MD or in fluid dynamics. However, when going to coarser resolution levels in order to capture longer time scales and especially when including reactions, the dynamics are often more complicated.

There are many different approaches and algorithms available for reaction dynamics, all of which are applicable in different situations with different strengths and weaknesses. A non-exhaustive overview of available models is given in Figure 1.3 by classifying them into regions of model resolution and applicability. The y-axis classifies models by how particle populations are modeled. The decision whether a model is suitable with respect to population scale (y-axis) largely depends on whether the average concentration (or density) $c = N/V$ for a given number of particles N in a volume V is a good description of the dynamics—leading to methods operating on concentrations such as reaction-diffusion partial differential equations (PDEs)—or whether individual particles are required to adequately describe the system—leading to methods based on stochastic processes of individual particles or discrete particle counts.

On the other hand, the x-axis in Figure 1.3 describes the spatial resolution of models. The spatial resolution required to adequately represent a process largely depends on the speed at which particles move (giving rise to an encounter rate k_E) and the speed at which reactions occur (giving rise to a reaction rate k_R). If the ratio $r = k_E/k_R$ is large it can be interpreted as particles rapidly moving and relatively slow reactions, i.e., particles do not have to diffuse over long periods of time before finding a reaction partner, spatial resolution is not at all required to describe the dynamics (referred to as "well-mixed" regime). On the other hand, a low value of r indicates that particles are slowly moving and do not react for a long time, leading to an overall kinetic behavior that depends on time *and* place at which reactions occur. This regime is typically termed "diffusion-limited". [55]

Considering concentration-based models we find the reaction-diffusion PDE with the highest spatial resolution. Reaction rate equations (RREs)—corresponding to a set of ordinary differential equations on the concentrations—do not resolve space at all. At the intermediate level are compartmented reaction rate equations which operate on a discretized (compartmentalized) space, each compartment described by its own set of RREs.

When populations are low and individual particles or counts of particles are used, we find particle-based reaction dynamics (PBRD) [56–58] with a high spatial resolution by describing each particle individually and on the other hand the chemical master equation (CME) with no spatial resolution but only defining a stochastic process on particle counts based on reaction propensities. As an intermediate, one can find the reaction-diffusion master equation (RDME) [59, 60], which operates on a box-discretized space with jump probabilities defined between adjacent boxes and a CME description inside each individual box. The CME is described in some more detail in Section 2.3.2, a good description of RDME can be found in [61]. The RDME model can be generalized to the so-called "spatiotemporal CME" [62], which allows an arbitrary compartmentalization of space instead of boxes.

As particle numbers fluctuate, it may also be instructive to build multiscalar/hybrid models that, depending on threshold values, switch between concentration and particle representations [62, 63].

We introduce RREs, CMEs, and a specific instance of PBRD in Section 2.3.

We cover `ReaDDy2` in Chapter 4: a tool for the efficient simulation of interacting-particle reaction dynamics (iPRD), which are conceptually introduced in Section 2.4 as a combination of overdamped Langevin dynamics (Section 2.2) and the Doi reaction model [64–67] (see Section 2.3.3). The iPRD model can also be understood as an extension of PBRD and was first introduced in Ref. 68 to combine the benefits of PBRD and MD simulations by modeling particle-based reaction dynamics while enabling full-blown interactions between particles as well as particles and the environment. This makes it one of the most detailed PBRD approaches—which also means that in comparison to the other methods a lot of computational power is required. When computational demands are high, efficient codes become particularly important.

In this thesis we develop and present the novel simulation library `ReaDDy2` [P1]. It aims to be a high-performance and end-user convenient code for iPRD. While efficient implementations for classical PBRD exist, they either do not offer the same degree of resolution or are simply less efficient in terms of accessible simulation times and system sizes (see, e.g., Refs. 57, 69–75). `ReaDDy2` is a complete rewrite of its Java-based predecessor `ReaDDy` (also introduced in Ref. 68) and provides—opposed to a collection of configuration files like in `ReaDDy`—a programmable Python interface in which the simulation environment, particle interactions, and reaction rules can be defined and the simulation can be run, stored, and analyzed. We show that `ReaDDy2` is significantly faster, feature-rich, flexible, and conveniently usable than `ReaDDy` (see Section 4.3.2). A C++ interface is available to enable deeper interactions with the library. The main computational work of `ReaDDy2` is performed not in Python but in hardware-specific simulation kernels. Due to the significant increase in performance and in particular due to much better scaling behavior with respect to system sizes, `ReaDDy2` is able to simulate larger systems and accumulate more simulation time. This is particularly important to be able to study biological processes which possess timescales on the order of minutes or more.

Compared to PBRD, the addition of interaction potentials in iPRD allows modeling volume

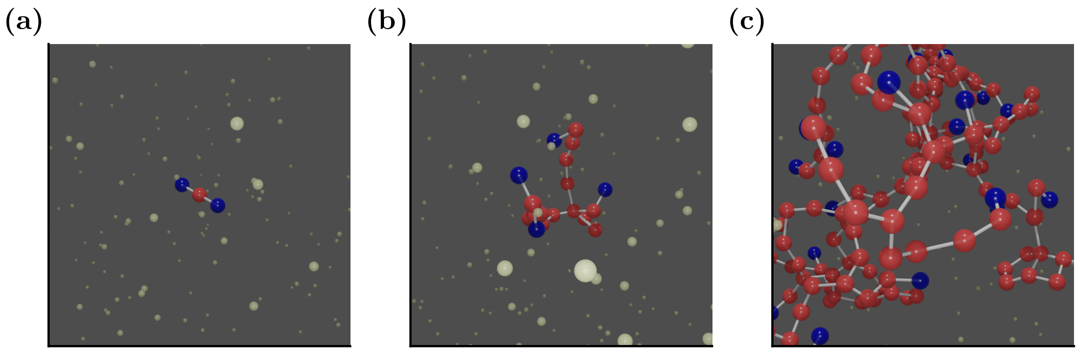


Figure 1.4: **Topology reactions in ReaDDy2.** Substrate particles (green) diffuse and can attach to blue particles (likely) as well as red particles (less likely). We show the temporal progression of the topology from its initial state in **(a)** via an intermediate **(b)** to its final state **(c)**.

exclusion effects, clustering, and self-assembly. A novel feature of `ReaDDy2` are topology reactions, generalizing the concept of chemical reactions. Topologies are groups of particles which are connected with harmonic bonds and can optionally be stiffened with angle and torsion potentials. Topology reactions are either spatially triggered (like a catalytic reaction) or occur with a fixed probability per unit time. Depending on their specification, they allow to concatenate, modify, split, or even completely dissolve topologies. For example, the growth and decay of polymer chains can be modeled (cf. Section 4.4.4), but it is also possible to model the self-assembly of more complex geometries like actin filaments, viral capsids [76, 77], or even flexible particle-based membrane models as in Ref. 78, which studies the effect of malaria parasites on red blood cell cytoskeletons.

In Figure 1.4 we show an example of this generalized concept. Small substrate particles are diffusing freely in space and upon spatial proximity with blue beads there is a fixed probability per unit time of forming a bond and being included in the topology. This transforms a three-particle initial topology in Figure 1.4a to a more complicated structure in Figure 1.4b. There is also a lower probability of forming bonds with red particles, resulting in branches. Furthermore, the topology can interact with itself: blue end beads have a smaller probability of attaching themselves to a red bead. This leads to closed loops, which can be observed in Figure 1.4c.

1.2.3 Discovering governing reactions from concentration data

In Chapter 5 we introduce `reactive SINDy`: an extension of SINDy that can estimate a governing network of reactions based on concentration time series data. To that end it uses the classical RREs subject to the law of mass action (LMA) (cf. Figure 1.3). The linear regression based SINDy method is extended to a linear tensor regression in order to accommodate terms within an ansatz reaction library which are coupled over a system of ODEs. These coupled terms are crucial to realize, e.g., fusion-type reactions, where educt species concentrations decrease and the product species concentrations increase accordingly. As an application example—among others—the MAPK pathway makes a comeback: concentration time series are simulated using a ground truth model and later being recovered from data. We show that we can indeed recover the true underlying reaction network if imposing sparsity. The lack of sparsity-encouraging regularization leads to spurious reactions.

Chapter 2

Theory

This chapter introduces the main theoretical concepts that are required for the rest of the thesis. In particular, we define stochastic processes with a focus on Markov processes, including an introduction into the theory of transfer operators (in particular Koopman and Perron–Frobenius operators). Transfer operators are of significance for the analysis of time series data and are heavily used in Chapter 3. Given these theoretical tools, overdamped Langevin dynamics is introduced (Section 2.2) as an example of Markovian stochastic processes. Also chemical reactions (in particular in view of reaction–diffusion models) can be understood as stochastic processes and we subsequently introduce the reaction rate equations (RREs) in Section 2.3.1, the chemical master equations (CMEs) in Section 2.3.2, and the Doi reaction model in Section 2.3.3. The combination of overdamped Langevin dynamics together with the Doi reaction model is a possible model for iPRD as detailed in Section 2.4. An implementation of iPRD is presented in the subsequent Chapter 4.

2.1 Stochastic processes

Everything in this thesis either is about generating time series data or analyzing it. The main focus for what is to follow is on Markov processes, for whose definition the concepts of probability spaces and random variables are briefly introduced. For a more thorough introduction any textbook on probability theory will do (e.g., Ref. [79]).

Let Ω be a non-empty sample space containing elementary outcomes, e.g., the set $\{1, 2, \ldots, 6\}$ when throwing a six-sided die. We associate a so-called σ-algebra $\mathcal{A}$ to Ω consisting of subsets of Ω with the (not minimally) defining properties

(i) empty set and sample space being contained: $\emptyset, \Omega \in \mathcal{A}$,

(ii) closedness under complements with respect to the sample space: $A \in \mathcal{A} \Rightarrow \Omega \setminus A \in \mathcal{A}$,

(iii) and closedness under countable intersections: $\bigcap_{i=1}^{\infty} A_i \in \mathcal{A}$ with $A_i \in \mathcal{A}$.

This algebra contains the set of all possible events, e.g., $\{2, 4, 6\} \in \mathcal{A}$ being the event of seeing an even die roll. Based on $\mathcal{A}$, a probability measure $\mathbb{P} : \mathcal{A} \to [0, 1]$ is defined, i.e.,

(i) a non-negative $\mathbb{P}[A] \geq 0 \; \forall A \in \mathcal{A}$,

(ii) null preserving $\mathbb{P}[\emptyset] = 0$,

(iii) and σ-additive (meaning $\mathbb{P}[\bigcup_{i=1}^{\infty} A_i] = \sum_{i=1}^{\infty} \mathbb{P}[A_i]$ for pairwise disjoint $A_i \in \mathcal{A}$) function.

The triplet $(\Omega, \mathcal{A}, \mathbb{P})$ is called a *probability space.* Measure theory teaches us why it is important to construct such σ-algebras as pre-image spaces for $\mathbb{P}$ instead of simply using the power set 2^Ω: if every possible subset was "measurable" by a probability measure, it sometimes is possible to create sinks and sources of volume simply by decomposition of a set into finitely many subsets and subsequent reassembly, as demonstrated in the Banach–Tarski paradox [80].

Another important and for this thesis very relevant concept is that of *random variables*, as they allow us to precisely define what we understand under a time series and more specifically Markovian stochastic processes. A random variable on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ is a map $X : \Omega \to E$, such that

$$\{\omega \in \Omega \mid X(\omega) \in A\} = X^{-1}(A) \in \mathcal{A} \quad \forall A \in \mathcal{E},$$

where E is the so-called *state space* and $\mathcal{E}$ a σ-algebra over E.

Here, we focus on state spaces which are subsets of the Euclidean space $E \subseteq \mathbb{R}^d$ equipped with the standard Borel σ-algebra $\mathcal{E}$, the smallest σ-algebra containing (usually referred to as "generated by") all possible half-open rectangles $\{(a_1, b_1] \times \ldots \times (a_d, b_d] \subset E : a_i \leq b_i\}$ [81].

A *stochastic process* on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ can now be defined as follows: a family of random variables $(\mathbf{x}_t)_{t\in I}$, $\mathbf{x}_t : \Omega \to E$ in a common measurable space $(E, \mathcal{E})$, where the parameter set I represents time and—while it can in principle be a more general object—is in the following assumed to be $I \subseteq \mathbb{R}_{\geq 0}$. For every $\omega \in \Omega$ the map

$$t \mapsto \mathbf{x}_t(\omega)$$

is called a path, realization, or trajectory of the process. In our context, a trajectory is understood as the movement of one or many particles in time given an initial configuration ω subject to some (usually stochastic but potentially also deterministic) dynamic.

We can characterize a stochastic process using its finite-dimensional distributions (FDDs). To this end, consider a non-empty but finite subset $J = \{t_1, t_2, \ldots, t_{|J|}\} \subseteq I$. This subset represents individual points in time and we denote $\mathcal{H}(I) \coloneqq \{J \subseteq I : J \neq \emptyset \wedge |J| < \infty\}$ the set of all possible subsets of such kind. The FDDs of the stochastic process $\mathbf{x}_t$ then refer to all push forward measures

$$\mathbb{P}_J^{\mathbf{x}_t}\left[B_1 \times \ldots \times B_{|J|}\right] \coloneqq \mathbb{P}\left[\mathbf{x}_{t_1} \in B_1, \ldots, \mathbf{x}_{t_{|J|}} \in B_{|J|}\right], \tag{2.1}$$

where $J \in \mathcal{H}(I)$ and $(B_1 \times \ldots \times B_{|J|}) \in \bigotimes_{t\in J} \mathcal{E}_t$ is an event in the product σ-algebra (with $\mathcal{E}_t = \mathcal{E}$ for all t). For a single particle under Brownian motion (see Section 2.1.1 and Equation (2.9) for more details), the expression $\mathbb{P}_J^{\mathbf{x}_t}[\cdot]$ denotes the probability that the particle is within certain parts B_i of the domain at points in time $t \in J$. A similar example is shown in Figure 2.1 of particles exploring a two-dimensional potential energy landscape.

The FDDs of a stochastic process allow us to determine many of its important properties, however most of the time—at least when dealing with the analysis of, e.g., simulation data—we are presented with the reversed situation: given finite measurements at fixed points in time, we seek to gain insight into an underlying stochastic process. Therefore, it is important that such a process indeed exists. The consistency theorem of Daniell and Kolmogorov (sometimes also called Kolmogorov's extension theorem) [82, Corollary 35.4], [83, Theorem 2.1.5] gives this guarantee under mild conditions on the state space and FDDs. In particular, E needs to be a Polish space (which are, e.g., Euclidean spaces with the usual Borel σ-algebra) and the FDDs need to be projective, meaning that if $J, H \in \mathcal{H}(I)$ with $J \subseteq H$, then $\mathbb{P}_J^X = \mathrm{pr}_H^J(\mathbb{P}_H^X)$ where pr_H^J refers to the projection of $E^{|H|}$ into $E^{|J|}$, cf. Equation (2.1). In other words, there should be consistency with respect to the implied measures when looking at subsets $I \subseteq J \in \mathcal{H}(I)$.

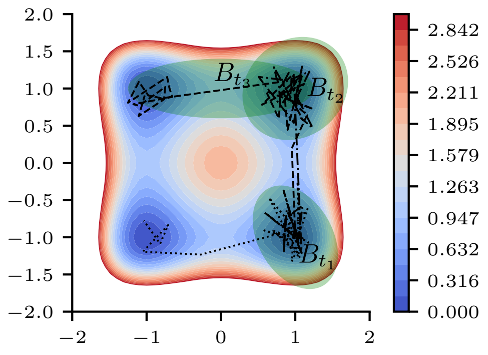


Figure 2.1: **Finite-dimensional distributions example.** Trajectories of three particles in a potential energy landscape (color-coded contour background). High energies correspond to unfavorable states. The figure also shows three sets $B_{t_1}, B_{t_2}, B_{t_3} \subset E$ corresponding to points in time $J = \{t_1, t_2, t_3\}$. This is an example of one finite-dimensional distribution (FDD) which is the probability of finding a particle in any measurable B_{t_i} at time t_i for the depicted stochastic process. The FDDs of the process comprise all possible finite and non-empty sets of points in time $\mathcal{H}(\mathbb{R}_{\geq 0})$.

All simulation and analysis methods introduced in this thesis deal with one particular class of stochastic processes, namely with *Markov processes*. The defining property is that

$$\mathbb{P}[\mathbf{x}_t \in B \mid \mathbf{x}_{t_1}, \ldots, \mathbf{x}_{t_{|J|}}] = \mathbb{P}[\mathbf{x}_t \in B \mid \mathbf{x}_{t_{|J|}}] \tag{2.2}$$

for all $J = \{t_1, \ldots, t_{|J|}\} \in \mathcal{H}(\mathbb{R}_{\geq 0})$ with $t_1 < \ldots < t_{|J|} < t$, i.e., the conditional probability to see $\mathbf{x}_t \in B$ at most depends on its most recent state but not on any further history.

Although Equation (2.2) tells us when we are dealing with a Markov process, it does not specify the family of Markov processes. To this end, we introduce transition probability kernels (and functions), which encode the information of the probability to find the process in set B at time t given it being in state x at time $s \leq t$.

We call a set of maps $(P_t)_{t\geq 0} : E \times \mathcal{E} \to \mathcal{E}$ a *Markov semigroup of kernels* on E if all of the following conditions hold:

1. $P_t(\mathbf{x}, B) \in \mathcal{E}$ for all $\mathbf{x} \in E, B \in \mathcal{E}$,
2. $B \mapsto P_t(\mathbf{x}, B)$ is a measure on $\mathcal{E}$ for all $\mathbf{x} \in E$,
3. $P_t(\mathbf{x}, \Omega) = 1$ for all $\mathbf{x} \in E$,
4. and the Chapman–Kolmogorov equation

$$P_{s+t} = P_s P_t \quad \forall s, t \in \mathbb{R}_{\geq 0}. \tag{2.3}$$

The product is defined as $[P_s P_t](\mathbf{x}, B) \coloneqq P_s(\mathbf{x}, P_t(\mathbf{x}, B))$ for all $\mathbf{x} \in E$ and $B \in \mathcal{E}$.

It should be noted that these conditions can be stated in greater generality, cf. [82, Chapter 36].

Using such a Markov semigroup of kernels and a probability measure μ_0 on $(E, \mathcal{E})$ we can define the FDDs of a Markov process via

$$\mathbb{P}_J^{\mathbf{x}_t}[B] \\ = \int_{B_1} \cdots \int_{B_{n-1}} \int_{B_n} P_{t_n - t_{n-1}}(\mathbf{x}_{n-1}, \mathrm{d}\mathbf{x}_n) P_{t_{n-1}-t_{n-2}}(\mathbf{x}_{n-2}, \mathrm{d}\mathbf{x}_{n-1}) \dots P_{t_1}(\mathbf{x}, \mathrm{d}\mathbf{x}_1) \tag{2.4}$$

for a given $\mathbf{x} \in E$, for all measurable $B = B_1 \times \dots B_n$, and for all $J \in \mathcal{H}(\mathbb{R}_{\geq 0})$, where $J = \{t_1, \dots, t_n\}$ with $i < j \Rightarrow t_i < t_j$. A stochastic process with these FDDs does indeed exist—see [82, Thm. 36.4])—and fulfills the Markov process property (2.2)—see [82, Thm. 42.3].

From now on we will assume that each measure $B \mapsto P_t(\mathbf{x}, B)$ is absolutely continuous (with respect to the Lebesgue measure), meaning that using the Radon–Nikodym theorem, there is a density $p_{s,t}(\cdot, \cdot)$, so that for each $\mathbf{x}$ and $t > 0$

$$B \mapsto P_t(\mathbf{x}, B) = \int_B p_{0,t}(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y}.$$

We call the function

$$p_{s,t} : E \times E \to \mathbb{R}_{\geq 0}, \quad \mathbb{P}[\mathbf{x}_t \in B \mid \mathbf{x}_s = \mathbf{x}] = \int_B p_{s,t}(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} \tag{2.5}$$

a *Markov transition kernel function* or *transition density.* In this case, the Chapman–Kolmogorov equation (2.3) can be reformulated as

$$p_{r,t}(\mathbf{x}, \mathbf{z}) = \int_E p_{r,s}(\mathbf{x}, \mathbf{y}) p_{s,t}(\mathbf{y}, \mathbf{z}) \mathrm{d}\mathbf{y}$$

for $0 \leq r \leq s \leq t < \infty$ and $\mathbf{x}, \mathbf{z} \in E$.

For now we only have studied individual trajectories with fixed initial configuration $\omega \in \Omega$. Often, however, one is interested in how a distribution over initial configuration evolves in time. With the acquired tools it is possible to do exactly that. Let μ_0 be an initial distribution, then we can evolve it to a later point in time $t > 0$ using (2.4): one obtains

$$\mu_t(B) = \int_E \int_B P_t(\mathbf{x}_0, \mathrm{d}\mathbf{x}_1) \mu_0(\mathrm{d}\mathbf{x}_0) = \int_E P_t(\mathbf{x}_0, B) \mu_0(\mathrm{d}\mathbf{x}_0), \tag{2.6}$$

which—assuming that the *measure* μ_0 is absolutely continuous with respect to the Lebesgue measure with *density* μ_0—can be reformulated to

$$\mu_t(B) = \int_B \int_E p_{0,t}(\mathbf{x}, \mathbf{y}) \mu_0(\mathbf{x}) \mathrm{d}\mathbf{x} \mathrm{d}\mathbf{y} \tag{2.7}$$

using the transition density (2.5) and Fubini's theorem. This introduces the second assumption that is used throughout this thesis: all distributions over states are assumed to be absolutely continuous with respect to the Lebesgue measure. To reduce the notation overhead, we denote the corresponding density with the same symbol as its measure.

Now we have all the tools to distinguish two different cases: time-homogeneous processes, which possess transition probabilities that do not depend on a particular point in time (this is the case for, e.g., autonomous differential equations) and the more general case of time-inhomogeneous processes.

The description of such processes relates to the mathematical framework of transfer operators [84–91]. We regard all operators that describe the temporal evolution of, e.g., probability densities or observables of the system's state as transfer operators. The operators we consider here are all linear operators (although in general not finite-dimensional). We encountered examples of transfer operators in Equations (2.6)–(2.7), mapping state densities forward in time. For an introduction to these operators we follow the presentations of [88] and [P4].

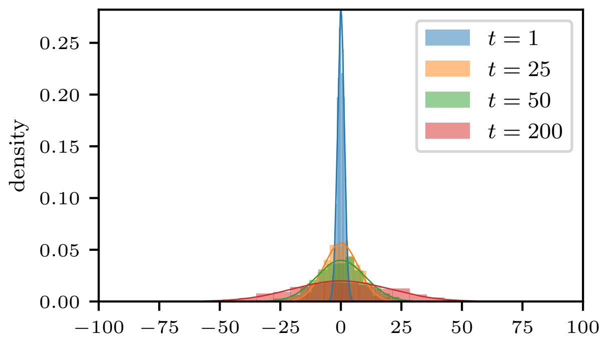


Figure 2.2: **Law of a Wiener process.** The law of a Wiener process is plotted after selected times. Estimated empirically by simulating $N = 1000$ instances of the process (bar chart) and computed analytically based on the probability density.

2.1.1 Time-homogeneous processes

This subsection is based on parts of Ref. P4, which is open access and distributed under the terms of the Creative Commons Attribution License (`CC BY 4.0`). Parts of the text have been adopted unchanged in this document.

Let $\{\mathbf{x}_t\}_{t\geq 0}$ be a Markovian stochastic process in state space $E \subset \mathbb{R}^d$ with its transition density as given in (2.5).

Time-homogeneity means that $p_{s,t}$ only depends on a lag-time $\tau := t - s$ but not on specific start and end times s and t individually, i.e.,

$$p_{s,t}(\mathbf{x}, \mathbf{y}) = p_\tau(\mathbf{x}, \mathbf{y}). \tag{2.8}$$

One of the most popular Markovian stochastic processes is a one-dimensional Wiener process (see Section 2.2 for details), which is a sequence of random variables $\{W_t\}_{t\geq 0}$ with $W_0 = 0$ subject to the transition kernel function

$$p_\tau(x, y) = \frac{1}{\sqrt{2\pi\tau}} e^{-\frac{(y-x)^2}{2\tau}}. \tag{2.9}$$

Since the density only depends on τ, the process is time-homogeneous. However, this does not mean that the law (or distribution) of the process

$$B \mapsto \mathrm{law}(W_t)[B] := \mathbb{P}[W_t \in B]$$

for sets $B \subset \Omega$ is time-independent, see Figure 2.2.

Generally speaking, transfer operators describe the effect of the underlying dynamics on functions of the state $\mathbf{x}_t$. A particularly important transfer operator, the Koopman operator (first introduced in [84]), is defined as

$$\mathcal{K}_\tau : L^\infty(\Omega) \to L^\infty(\Omega), \quad [\mathcal{K}_\tau g](\mathbf{x}) := \int g(\mathbf{y}) p_\tau(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} = \mathbb{E}[g(\mathbf{x}_{t+\tau}) \mid \mathbf{x}_t = \mathbf{x}], \tag{2.10}$$

evolving the observable g for a lag-time $\tau > 0$. The function space $g \in L^\infty(\Omega)$ is of the family of L^p spaces with

$$L^p(\Omega) := \left\{ f : \Omega \to \mathbb{C} \text{ s.t. } f \text{ measurable } \wedge \|f\|_p := \left(\int_\Omega |f|^p \right)^{1/p} < \infty \right\}$$

for $1 \leq p < \infty$ and $L^\infty(\Omega) := \{f : \Omega \to \mathbb{C} \text{ s.t. } f \text{ measurable } \wedge \exists C \geq 0 : |f(x)| \leq C \text{ a.e.}\}$. Strictly speaking L^p consists of equivalence classes of measurable functions where the equivalence relation is defined by functions being equal "almost everywhere", i.e., can differ on sets of measure zero; all statements therefore refer to an equivalence class or one representative of that class. In case of deterministic dynamics $\mathbf{x}_{t+\tau} = \mathbf{\Psi}(\mathbf{x}_t)$, the transition density consists of delta peaks and the Koopman operator is simply the composition $\mathcal{K}_\tau g = g \circ \mathbf{\Psi}$.

Another commonly used transfer operator to describe Markovian dynamics is the Perron–Frobenius (PF) operator [92, 93]

$$\mathcal{P}_\tau : L^1(\Omega) \to L^1(\Omega), \quad [\mathcal{P}_\tau f](\mathbf{y}) = \int \mathbf{f}(\mathbf{x}) p_\tau(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{x}, \tag{2.11}$$

which evolves probability density functions $f \in L^1(\Omega)$. Since it is a Markov operator ($\mathcal{P}_\tau f \geq 0$ and $\|\mathcal{P}_\tau f\| = \|f\|$ for all $f \geq 0$), probability density functions are mapped to probability density functions [92]. The definition's expression—although not framed in the transfer operator setting—already appeared in Equation (2.7).

The PF operator is the adjoint of the Koopman operator [92, 93], i.e.,

$$\langle \mathcal{P}_\tau f, g \rangle = \langle f, \mathcal{K}_\tau g \rangle \quad \forall f \in L^1(\Omega), g \in L^\infty(\Omega), \tag{2.12}$$

where the bracket is defined as $\langle h_1, h_2 \rangle := \int_\Omega h_1(\mathbf{x}) h_2(\mathbf{x}) \mathrm{d}\mathbf{x}$. Although L^p spaces with $p \neq 2$ are not Hilbert spaces, the product of two functions $h_1 \in L^p(\Omega)$ and $h_2 \in L^q(\Omega)$ is integrable as long as $1/p + 1/q = 1$ for $1 \leq p, q \leq \infty$.

For the rest of this subsection we assume that there exists a stationary distribution $\mu \in L^1(\Omega)$ satisfying $\mathcal{P}_\tau \mu = \mu$. If such a stationary distribution μ exists and $\mu(x) > 0$ almost everywhere, then the time-homogeneous processes $\{\mathbf{x}_t\}_{t \geq 0}$ is ergodic and the stationary distribution is unique. Vice versa, if $\{\mathbf{x}_t\}_{t \geq 0}$ is ergodic, there exists at most one stationary distribution [92].

Given the stationary distribution we can define a PF operator with respect to μ (also simply called the transfer operator),

$$\mathcal{T}_\tau : L^1(\Omega) \to L^1(\Omega), \quad [\mathcal{T}_\tau u](\mathbf{y}) = \frac{1}{\mu(\mathbf{y})} \int \mu(\mathbf{x}) u(\mathbf{x}) p_\tau(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{x}. \tag{2.13}$$

Instead of evolving probability densities f, it evolves densities $u = f/\mu$ with respect to the stationary distribution. Due to this construction we obtain the normalization $\mathcal{T}_\tau \mathbb{1} = \mathbb{1}$, encoding that the stationary distribution is preserved under propagation in time.

Under some conditions [87, 89, 90, 94], the function spaces from and to which the operators map can be assumed to be reweighted L^2 spaces (and therefore Hilbert spaces),

$$L^2_\rho(\Omega) := \left\{ h : \|h\|^2_\rho < \infty \text{ with } \langle f, g \rangle_\rho := \int_\Omega f(\mathbf{x}) \overline{g(\mathbf{x})} \rho(\mathbf{x}) \mathrm{d}\mathbf{x} \right\}, \tag{2.14}$$

where $\mathcal{P}_\tau : L^2_{\mu^{-1}}(\Omega) \to L^2_{\mu^{-1}}(\Omega)$, $\mathcal{T}_\tau : L^2_\mu(\Omega) \to L^2_\mu(\Omega)$, and $\mathcal{K}_\tau : L^2_\mu(\Omega) \to L^2_\mu(\Omega)$. In what follows we assume that this is the case.

Via a straightforward calculation using (2.12) one obtains that Koopman operator and transfer operator are also adjoint in the reweighted spaces, i.e.,

$$\langle \mathcal{T}_\tau f, g \rangle_\mu = \langle f, \mathcal{K}_\tau g \rangle_\mu \quad \forall f, g \in L^2_\mu(\Omega). \tag{2.15}$$

2.1.2 Time-inhomogeneous processes

In the case of time-inhomogeneous processes, the transition density (2.5) depends directly on the initial and/or final time; i.e., Equation (2.8) no longer holds. This also means that the operators (2.10)–(2.13) no longer depend on the lag-time τ but rather on specific start and end times s and t, respectively (equivalently: on start time s and with lag-time τ). For such systems there is in general no stationary distribution μ, so we consider the distribution μ_s at initial time s and μ_t at final time t, related by $\mu_t = \mathcal{P}_{s,t}\mu_s$. The transfer operator can be defined as

$$\mathcal{T}_{s,t} : L^2_{\mu_s}(\Omega) \to L^2_{\mu_t}(\Omega), \quad \mathcal{T}_{s,t}u = \frac{1}{\mu_t}\mathcal{P}_{s,t}(u\mu_s). \tag{2.16}$$

As in the time-homogeneous case, this operator is the adjoint of the time-inhomogeneous Koopman operator [95]

$$\langle \mathcal{T}_{s,t}f, g\rangle_{\mu_t} = \langle f, \mathcal{K}_{s,t}g\rangle_{\mu_s} \quad \forall f \in L^2_{\mu_s}(\Omega)\forall g \in L^2_{\mu_t}(\Omega),$$

where $\mathcal{K}_{s,t} : L^2_{\mu_t}(\Omega) \to L^2_{\mu_s}(\Omega)$.

One particular advantage of considering any of the transfer operators over directly analyzing the (in general highly nonlinear) temporal evolution of processes' full states is their linearity. While the considered operator usually cannot be represented as a finite-dimensional matrix, one can seek projections and/or approximations. These approximations can be used to identify and project onto the slow processes as well as metastable and coherent sets [87, 96, 97]. There are different methods available for making the approximations which vary in their assumptions, approximation power, and interpretability, some of which are accompanied by variational theorems.

In Sections 3.3–3.5 transfer operators are used to build dynamical models from data.

2.2 Overdamped Langevin dynamics

With the knowledge of Markov processes, we can introduce overdamped Langevin dynamics with isotropic diffusion: a class of (Markovian) systems, which can be understood as an extension of Wiener processes and/or Brownian dynamics (free diffusion, as in Figure 2.2) to particles which can interact with each other and their environment, being of particular importance for this thesis.

To this end, let us first introduce Wiener processes (in the literature sometimes also referred to as Brownian motion) in some more detail, as these are a component of overdamped Langevin dynamics. Such processes describe the movement of a particle in a bath of solvent particles which constantly collide with one another, so that velocities immediately decorrelate. As a result, the motion is modeled as independent Gaussian increments. Formally, a d-dimensional Wiener process with variance σ^2 may be described (see, e.g., Ref. 98) as a stochastic process $\{\mathbf{W}_t\}_{t\geq 0}$ with

1. initial condition $\boldsymbol{W}_0(\omega) = \mathbf{0} \in \mathbb{R}^d$ almost surely,
2. statistically independent increments $(\mathbf{W}_t(\omega) - \mathbf{W}_s(\omega)) \sim \bigotimes_{i=1}^d \mathcal{N}(0, \sigma^2(t-s))$ for all $0 \leq s < t$,
3. and trajectories $t \mapsto \mathbf{W}_t(\omega)$ being continuous almost surely.

If not noted otherwise, we assume $\sigma^2 = 1$. Such stochastic processes also solve the stochastic differential equation (SDE)

$$\mathrm{d}\mathbf{x}_t = \sigma \mathrm{d}\mathbf{W}_t. \tag{2.17}$$

It furthermore can be seen that $\mathbb{E}[\mathbf{W}_t] = \mathbf{0}$, i.e., the process is mean-free, and possesses a delta-correlated covariance

$$\mathbb{E}\left[\mathbf{W}_t \mathbf{W}_{t'}^\top\right] = \mathrm{diag}(\sigma^2)\delta(t-t').$$

Based on the notion of a Wiener process, we define *overdamped Langevin dynamics*. The motion of $N > 0$ interacting particles $\mathbf{x}_t^{(i)} \in \mathbb{R}^d$, $i = 1, \ldots, N$ can be described by the SDE

$$\mathrm{d}\mathbf{x}_t^{(i)} = -\frac{D^{(i)}(T)}{k_B T}\mathbf{f}_t^{(i)}\mathrm{d}t + \sqrt{2D^{(i)}(T)}\mathrm{d}\mathbf{W}_t^{(i)}, \tag{2.18}$$

where T is a fixed system temperature, k_B the Boltzmann constant, and $D^{(i)}(T) \in \mathbb{R}$ the particle's diffusion constant. Particles can interact with the environment or other particles via the force $\mathbf{f}_t^{(i)} \in \mathbb{R}^d$. Typically the forces are gradients of a potential function (force field), meaning $\mathbf{f}_t^{(i)} = -\nabla_i U(\mathbf{x}_t^{(1)}, \mathbf{x}_t^{(2)}, \ldots, \mathbf{x}_t^N)$ with $U : \mathbb{R}^{N\times d} \to \mathbb{R}$. The particle's movement can be decomposed into a deterministic part which depends on the force $\mathbf{f}_t^{(i)} \in \mathbb{R}^3$ and the stochastic term $\sqrt{2D^{(i)}(T)}\boldsymbol{W}_t^{(i)}$. The stochastic terms $\mathbf{W}_t^{(i)}$ and $\mathbf{W}_t^{(j)}$ are uncorrelated for particles $i \neq j$. Whether the process is time-homogeneous or time-inhomogeneous entirely depends on $\mathbf{f}_t^{(i)}$: an explicit dependence on time t leads to a time-inhomogeneous process, if $\mathbf{f}_t^{(i)} = \mathbf{f}^{(i)}$, it is time-homogeneous. For example a periodically acting external force which drives the system would enter as explicit time dependency and consequently the dynamics are classified as time-inhomogeneous.

There is a relation to the theory of transfer operators introduced the previous two Sections 2.1.1 and 2.1.2 (see, e.g., Section 2.5 of Ref. 99): For overdamped Langevin dynamics, the actions of the Koopman (Equation (2.10)) and Perron–Frobenius (Equation (2.11)) operators are solutions of the Kolmogorov backward and Kolmogorov forward equation [100], respectively. These equations are parameterized by the deterministic and stochastic coefficients of the SDE (2.18). The Kolmogorov forward equation was previously discovered by Adriaan Fokker [101] and Max Planck [102] and is therefore also known as the Fokker–Planck equation. It should be noted that these relationships can be stated in greater generality for diffusion processes (of which overdamped Langevin dynamics are a special case). The naming is due to the forward equations being solved forward in time (as initial value problem) and the backward equations being solved backward in time (as final value problem). Consequently, the Perron–Frobenius and Koopman operator are sometimes referred to as forward and backward operator, respectively.

Both Equation (2.17) and more generally Equation (2.18) can be numerically integrated by the Euler–Maruyama scheme [103]

$$\mathbf{x}_{t+\tau}^{(i)} = \mathbf{x}_t^{(i)} - \tau\frac{D^{(i)}(T)}{k_B T}\mathbf{f}_t^{(i)} + \sqrt{2D^{(i)}(T)\tau}\boldsymbol{\eta}_t, \tag{2.19}$$

where $\tau > 0$ is a finite integration time step size and $\boldsymbol{\eta}_t \sim \bigotimes_{i=1}^d \mathcal{N}(0,1)$ is a normally-distributed random variable. The diffusion constant $D^{(i)}$ effects the magnitude of the random displacement.

For example, in Figure 2.3 we depict (time-homogeneous) overdamped Langevin dynamics in $d = 2$ dimensions with a potential landscape

$$U : \mathbb{R}^2 \to \mathbb{R}, \quad \mathbf{x} \mapsto \begin{cases} \frac{1}{2}k(\|\mathbf{x}\|_2 - r)^2 & \text{, if } \|\mathbf{x}\| \geq r, \\ 0 & \text{, otherwise,} \end{cases}$$

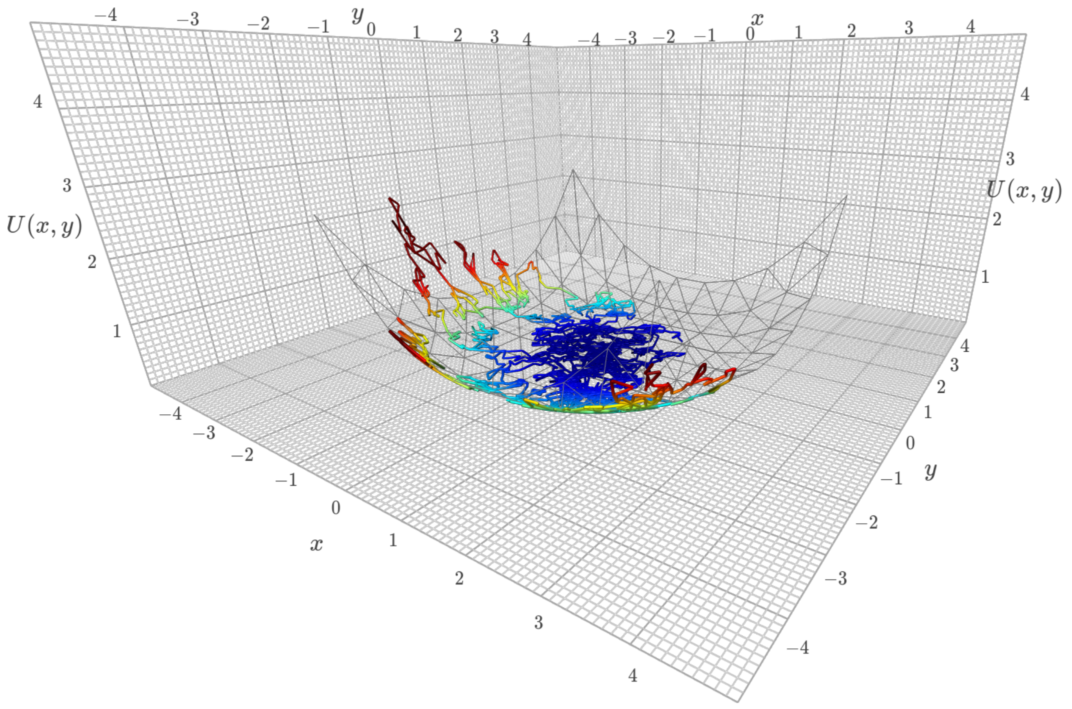


Figure 2.3: **Overdamped Langevin dynamics example.** A particle diffuses in a two-dimensional potential landscape $U : \mathbb{R}^2 \to \mathbb{R}$, which has zero potential in a circular area around the origin with a radius of $r = 3$ and outside of that region acts as harmonic repulsion. The figure shows the potential energy as wireframe and three realizations of the process, each starting in $\mathbf{x}_0 = (0, 0)$ and then being integrated numerically using Euler–Maruyama. For visualization purposes, the z component of the realizations corresponds to the potential energy and the trajectories are also color-coded according to their energy (energetically favorable state in dark blue, unfavorable in dark red).

for $r = 3$ and $k = 1$, i.e., a harmonic spherical inclusion potential which is flat in a circle of radius $r = 3$ around the origin and otherwise acts on the particle with force constant $k = 1$. We show three realizations of the process, integrated by Euler–Maruyama. Since here the potential landscape does not depend on the position of particles relative to each other, we consider it as a function that maps from the spatial position of a particle to its energy.

Overdamped Langevin dynamics is used in `ReaDDy2` (introduced in Chapter 4) for the propagation of particles.

2.3 Reaction models

As depicted in Figure 1.3, there are different kinds of models describing systems containing chemical reactions. Here we want to introduce the ones which are of relevance for this thesis. It should be noted that there is compatibility between these models in certain limits and it is possible to create hybrid models, too [55, 62].

2.3.1 Reaction rate equations

The reaction rate equation (RRE) is a formulation that assumes a well-mixed system in which particle numbers are high enough, so that a description by densities is appropriate (cf. Figure 1.3). Here we further assume the LMA, meaning that there is proportionality between reaction rates and concentrations of reactants [55, 104].

We are observing the concentrations of S chemical species in time t

$$\mathbf{c}_t = \begin{pmatrix} c_t^{(1)} \\ \vdots \\ c_t^{(S)} \end{pmatrix} \in \mathbb{R}^S. \tag{2.20}$$

The RRE can be formulated as a system of ODEs

$$\dot{\mathbf{c}}_t = \sum_{k=1}^{K} \alpha_k(\mathbf{c}_t)\boldsymbol{\nu}_k, \tag{2.21}$$

where $\boldsymbol{\nu}_k \in \mathbb{R}^S$ is a so-called stochiometric vector describing the change in population for each species with respect to reaction k and $\alpha_k : \mathbb{R}^S \to \mathbb{R}$ are propensity functions. In general, a propensity function can be defined as probability that the reaction occurs within an infinitesimal time interval given a particular state of the system [105], i.e.,

$$\lambda \mathrm{d}t = \mathbb{P}[\text{reaction occurs within } [t, t + \mathrm{d}t) \mid \mathbf{x}_t = \mathbf{x}]. \tag{2.22}$$

A general expression for the change of concentration of reactant s as a result of order-0 reactions (creation), order-1 reactions (transitions of other species into s, transitions of s into other species, or annihilation), order-2 reactions (production or consumption of s by the encounter of two species), etc. is given by

$$\dot{c}_t^{(s)} = \sum_i \beta_{s,0}^{(i)} + \sum_i \beta_{s,1}^{(i)} c_t^{(i)} + \sum_{i,j} \beta_{s,2}^{(i,j)} c_t^{(i)} c_t^{(j)} + \ldots, \tag{2.23}$$

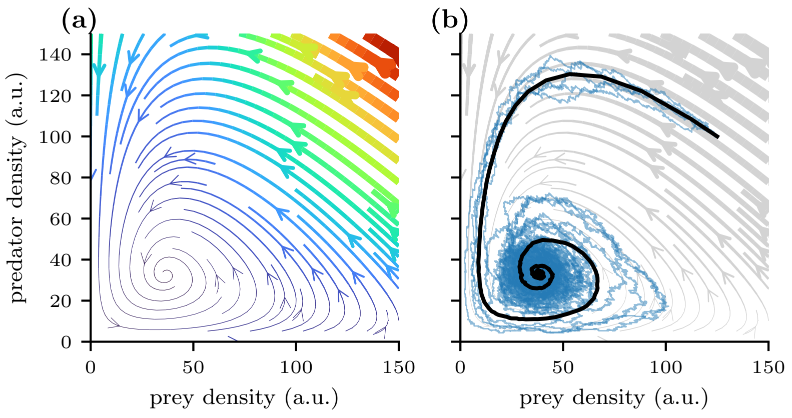


Figure 2.4: **Lotka–Volterra system with social friction.** We demonstrate the Lotka–Volterra system with a social friction term. **(a)** The plot shows the reaction rate equation direction field of population change given a certain state of the system. Thickness and color of the streamlines correspond to the magnitude (velocity) of change. **(b)** Additionally we show 150 averaged simulations of the corresponding chemical master equation in black with five individual realizations in light blue.

where the $\beta_{s,k}^{(\dots)}$ values are constants belonging to the reactions of order k. These rate constants however can incorporate several underlying reactions at once. For example, the two reactions

$$s_1 \xrightarrow{\xi_1} s_2 \tag{2.24}$$

$$s_1 \xrightarrow{\xi_2} s_3 \tag{2.25}$$

both contribute to $\dot{c}_t^{(1)} = \beta_{1,1}^{(1)} c_t^{(1)} = -(\xi_1 + \xi_2) c_t^{(1)}$.

A widely studied RRE is the so-called Lotka–Volterra system [106, 107]. It was independently formulated by A. Lotka and V. Volterra in the 1920s and describes a predator-prey situation. It reads

$$\begin{aligned} \dot{c}_t^{(1)} &= (\alpha - \beta c_t^{(2)} - \lambda c_t^{(1)}) c_t^{(1)} \\ \dot{c}_t^{(2)} &= (\delta c_t^{(1)} - \gamma - \mu c_t^{(2)}) c_t^{(2)}, \end{aligned} \tag{2.26}$$

where $c^{(1)}$ corresponds to a concentration of prey, $c^{(2)}$ corresponds to a concentration of predators, and $\alpha, \beta, \gamma, \delta, \lambda, \mu \geq 0$ are constants. This is a slightly modified version of the original Lotka–Volterra model, which is recovered by $\lambda = \mu = 0$. The modification introduces social friction which prohibits unlimited growth of prey if there are no predators. In Figure 2.4a the corresponding direction field is shown with parameters $\alpha = 2$, $\beta = 0.05$, $\gamma = 1.5$, $\delta = 0.05$, and $\lambda = \mu = 0.01$.

2.3.2 Chemical master equation

The chemical master equation (CME) can be understood as the counterpart to the RRE if the particle numbers are low so that concentrations are no longer an adequate description of populations but rather discrete counts should be used. It therefore finds itself in the lower right corner of overview Figure 1.3.

The CME is typically given in terms of a stochastic process (called the *reaction jump process*) $\{\mathbf{c}_t\}_{t\geq 0}$ with $\mathbf{c}_t \in \mathbb{N}^S_{\geq 0}$ and $\mathbf{c}_0 = \mathbf{x}_0$ being the number of particles for each of S species. The temporal evolution can be described by

$$\mathbf{c}_t = \mathbf{c}_0 + \sum_{k=1}^{K} R_t^{(k)} \boldsymbol{\nu}_k, \tag{2.27}$$

where—similar to the RRE model—$\boldsymbol{\nu}_k \in \mathbb{N}^S_{\geq 0}$ are stochiometric vectors and $\{R_t^{(k)}\}_{t\geq 0}$ with $R_t^{(k)} \in \mathbb{N}_{\geq 0}$ is a Poisson process describing the number of times a reaction k has occurred in the time interval $[0, t]$ [55].

Instead of characterizing the stochastic process itself, the CME is formulated with respect to transition densities (2.5). For brevity, we define

$$p_t(\mathbf{x}) \coloneqq \mathbb{P}[\mathbf{c}_t = \mathbf{x} \mid \mathbf{c}_0 = \mathbf{x}_0].$$

Then, the CME is defined as [55, 108, 109]

$$\dot{p}_t(\mathbf{x}) = \sum_{k=1}^{K} \left[\alpha_k(\mathbf{x} - \boldsymbol{\nu}_k) p_t(\mathbf{x} - \boldsymbol{\nu}_k) - \alpha_k(\mathbf{x}) p_t(\mathbf{x})\right], \tag{2.28}$$

where—similar to the RRE model—$\boldsymbol{\nu}_k \in \mathbb{N}^S_{\geq 0}$ are stochiometric vectors and $\alpha_k : \mathbb{N}^S_{\geq 0} \to \mathbb{R}$ are propensity functions. The term $-\alpha_k(\mathbf{x})p_t(\mathbf{x})$ can be understood as “outflow” of probability from state $\mathbf{x}$ to other states, vice versa the term $\alpha_k(\mathbf{x} - \boldsymbol{\nu}_k)p_t(\mathbf{x} - \boldsymbol{\nu}_k)$ corresponds to “inflow” from other states [55].

While the model is a finite set of ODEs, it is yet defined state-wise for each individual $\mathbf{x}$. This means that in general there are infinitely many equations to solve. One possibility to obtain moments of Equation (2.28) is generating realizations of the underlying stochastic process $\{\mathbf{c}_t\}_{t\geq 0}$ with, e.g., kinetic Monte Carlo (also called the Gillespie stochastic simulation algorithm (SSA)) [110, 111].

The CME is compatible to the RRE using population scaling: by increasing the volume but keeping the particle concentration constant, i.e., considering a rescaled stochastic process $\{\tilde{\mathbf{c}}_t^V\}_{t\geq 0}$ with $\tilde{\mathbf{c}}_t^V \coloneqq 1/V \mathbf{c}_t$ and rescaled propensities $\tilde{\alpha}_k^V(\mathbf{c}) \coloneqq \alpha_k^V(\mathbf{c}V)$, the RRE is recovered in the limit of $V \to \infty$. [55]

In Figure 2.4b we show a Lotka–Volterra system with population friction term with volume rescaling $V = 3$. The involved chemical reactions (corresponding to Equation (2.26)) are

$$\begin{cases} A & \xrightarrow{\alpha} A + A & \text{, prey birth,} \\ A + B & \xrightarrow{\beta} B + B & \text{, predator eats prey,} \\ B & \xrightarrow{\gamma} \emptyset & \text{, predator death,} \\ A + A & \xrightarrow{\lambda} A & \text{, prey social friction,} \\ B + B & \xrightarrow{\mu} B & \text{, predator social friction,} \end{cases} \tag{2.29}$$

where the rates are chosen identical to the ones used in Section 2.3.1 and $\beta = \delta$. All kinetic Monte Carlo sampled trajectories start with 375 prey and 300 predator particles in a volume of $V = 3$ and are evolved until final time $t = 150$ is reached. The black line is the mean over 150 such trajectories using a time discretization with 1000 bins. In light blue five example realizations of the stochastic process are shown.

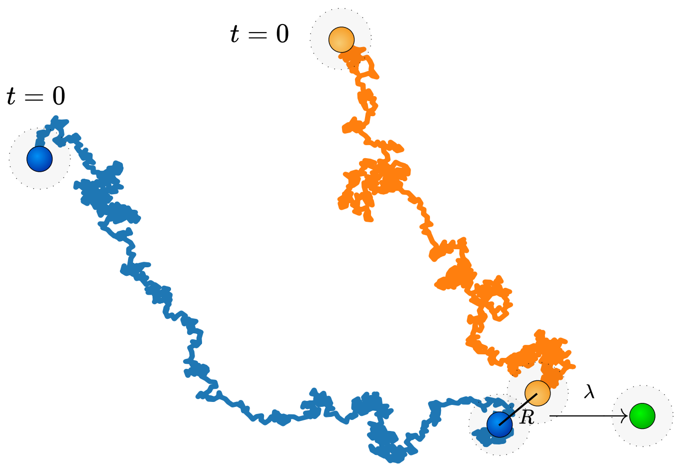


Figure 2.5: **Doi reaction model illustration.** Two educt particles p_1, p_2 (blue and orange, respectively) approach each other via a diffusive process (for example with overdamped Langevin dynamics, cf. Section 2.2). Once they are within reaction distance $\|p_2 - p_1\| \leq R$, there is a microscopic reaction rate $\lambda > 0$ with which they form a product particle (green). Such a process could model, e.g., a binding event between two proteins.

2.3.3 Doi model

A popular and spatiotemporally high-resolution mathematical model for bimolecular reactions $A + B \to C$ to study the formation of bimolecular complexes ("clumping" of particles) is by Smoluchowski [112]. In terms of Figure 1.3 it finds itself in the lower left corner, best describing low-population and diffusion-limited systems. In this model, particles are represented by spheres with a prescribed radius, diffuse in space and form complexes upon contact. Despite the popularity of the Smoluchowski model, it possesses at the very least some difficulty in the implementation of particle-based computer simulations due to the immediate nature of reactions. Typically, particles are propagated using a fixed step size, so it is hard to determine if, when, and where a collision occurred. Vice versa, products of dissociation reactions should not be placed in contact with another.

A later model takes a different approach: in the Doi model [64–67] (sometimes also referred to as the λ–ρ model [56]), particles no longer possess a radius. Instead, point particles diffuse freely, subject to, e.g., overdamped Langevin dynamics (2.18). Upon encountering a reaction partner (meaning being within a prescribed reaction distance R), there is a microscopic rate $\lambda > 0$, with which the reaction occurs, cf. Figure 2.5. This makes the model more suitable and popular for PBRD simulations. The Doi model can be understood as a generalization of the Smoluchowski model for $\lambda \to \infty$ and R representing the radii of the two reaction partners.

Given a system with two diffusing particles giving rise to the stochastic process $\{\mathbf{x}_t\}_{t\geq 0} = \{(\mathbf{a}_t, \mathbf{b}_t)\}_{t\geq 0}$ with $\mathbf{a}_t, \mathbf{b}_t \in \mathbb{R}^3$, the Doi reaction model describes a propensity function $\lambda = \lambda(\mathbf{a}_t, \mathbf{b}_t)$, cf. Equation (2.22). For the Doi model and for λ to be non-zero, the particles should be within a certain reactive radius $\|\mathbf{a}_t - \mathbf{b}_t\|_2 \leq R$. In that case the reaction probability is uniform in time and follows a Poisson process with rate λ. Poisson processes can be defined via a stochastic process $\{N_t\}_{t\geq 0}$ with $N_0 = 0$ almost surely, independent increments (cf. definition of Wiener processes in Section 2.2), and a Poisson distribution

$$\mathrm{Pois}(\lambda) = \frac{\lambda^n e^{-\lambda}}{n!}$$

with parameter $(t-s)\lambda$ for each random variable $N_t - N_s$, $0 \leq s < t < \infty$ for $n \in \mathbb{N}_{\geq 0}$ [79]. Such processes describe the probability of observing a number n of events within a certain time. In particular, for observing no events within the time interval $I = [0, \tau)$ one obtains

$$\mathbb{P}[N_\tau = 0] = e^{-\lambda\tau}.$$

Vice versa, for observing any number of events it is

$$\mathbb{P}[N_\tau \neq 0] = 1 - e^{-\lambda\tau}.$$

This means if using a numerical scheme with step-size $\tau > 0$ to propagate two particles A and B within reaction distance R, the probability that a reaction $A + B \to C$ occurs within $[t, t+\tau)$ can be evaluated as

$$p(\lambda; \tau) \coloneqq \mathbb{P}[\text{reaction occurs in } [t, t+\tau)] = 1 - e^{-\lambda\tau}. \tag{2.30}$$

Equation (2.30) can also be used for enzymatic (or catalysis) type reactions $A + C \to B + C$, which occur in the proximity of a catalyst C. If there is only one educt as in conversions $A \to B$ or fission reactions $A \to B + C$, there is no reactive area with radius R but the reactions occur spontaneously according to the Poisson clock.

2.4 Interacting-particle reaction dynamics

Interacting-particle reaction dynamics (iPRD) were introduced in [68] to combine the benefits of PBRD and MD simulations by modeling particle-based reaction dynamics while enabling full-blown interactions between particles as well as particles and their environment.

IPRD can be implemented as a combination of overdamped Langevin dynamics (Section 2.2) and the Doi reaction model (Section 2.3.3). This combination facilitates the simulation of reaction kinetics in crowded environments, involving complex molecular geometries such as polymers, and employing complex reaction mechanisms such as breaking and fusion of polymers. iPRD simulations are ideal to simulate the detailed spatiotemporal reaction mechanism in complex and dense environments, such as in signaling processes at cellular membranes, or in nano- to microscale chemical reactors. This kind of model therefore finds itself in the lower left corner of Figure 1.3: it operates at a high spatial resolution and performs best if reactions are diffusion-limited and populations are low so that the description of the system by individual particles over densities is appropriate or even necessary.

When simulating systems with iPRD opposed to, e.g., sampling from the CME (cf. Figure 2.4), nontrivial—in this case wave-like—spatial patterns can emerge. We illustrate this in Figure 2.6. It shows the temporal progression of a system simulated using Lotka–Volterra dynamics (see, e.g., Equation (2.29)) together with immobile particles (in black) which act as barriers and induce spatial exclusion. Besides the wave-like patterns, the predator and prey densities are still showing their characteristic oscillatory behavior. The exact simulation setup can be found in Appendix A.1.

The mathematical and theoretical treatment of such models in terms of stochastic processes and transfer operators however is difficult due to their reactive nature: particle numbers fluctuate

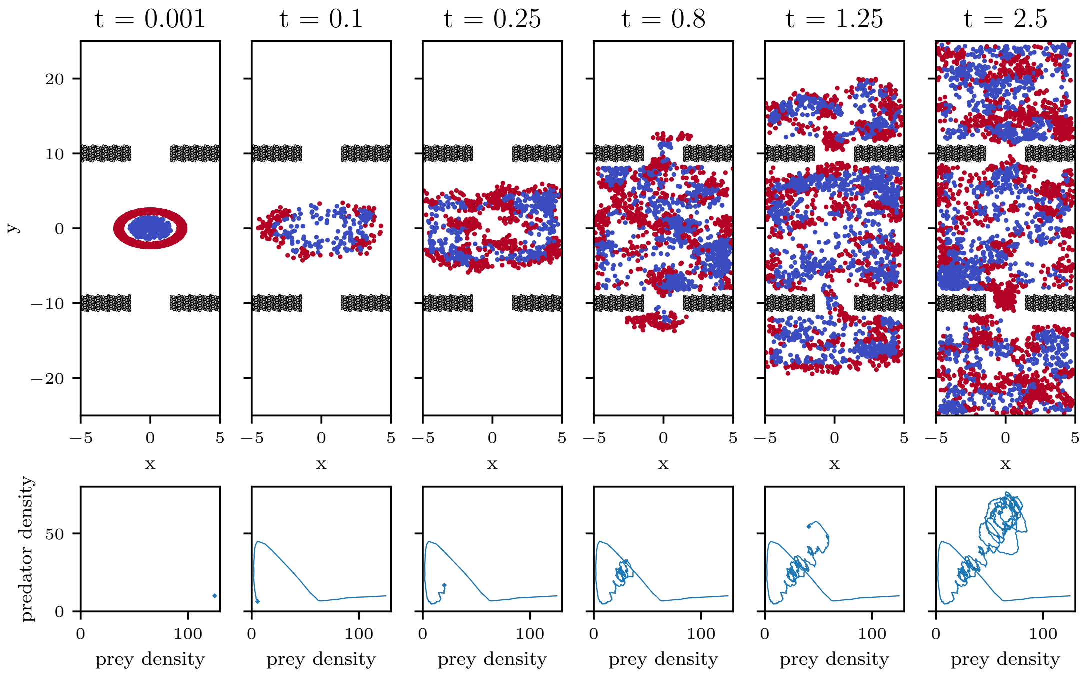


Figure 2.6: **Lotka–Volterra system with spatial resolution.** We simulate Lotka–Volterra dynamics with social friction using an interacting-particle reaction dynamics (iPRD) model. In the upper row, red dots correspond to prey particles and blue dots correspond to predator particles. Black dots correspond to immobile particles. The black dots correspond to particles, which act as spatial barriers via repulsive interaction potentials and induce a nontrivial geometry. In the lower row we show the temporal evolution of predator and prey densities.

and the state space may change its effective dimension at any point in time. Also particles of the same species are indistinguishable to one another giving rise to permutational invariances which have to be respected in mathematical descriptions. One possibility is to build on the idea of Fock spaces [113] from quantum dynamics and construct state spaces containing variable numbers of particles together with appropriate reaction and diffusion operators to formulate a probabilistic evolution equation [55, 114]. This description embeds iPRD into the framework of stochastic processes as presented in Section 2.1.

Aside from the mathematical description, also the efficient implementation of iPRD integrators is challenging but ultimately necessary to reach relevant time scales and sufficient statistics of the simulated processes. To this end, we introduce the iPRD simulation software package `ReaDDy2` in Chapter 4, which combines overdamped Langevin dynamics (Section 2.2) for particle propagation with the Doi reaction model (Section 2.3.3).

Chapter 3

Machine learning dynamical models from time series data

The results of this chapter were originally presented in Ref. P4:

> M. Hoffmann, M. K. Scherer, T. Hempel, A. Mardt, B. de Silva, B. E. Husic, S. Klus, H. Wu, N. Kutz, S. L. Brunton, F. Noé. "Deeptime: a Python library for machine learning dynamical models from time series data". In: *Mach. learn.: sci. technol.* (2021). URL: `https://doi.org/10.1088/2632-2153/ac3de0`.

Moritz Hoffmann (MH) was lead author and sole first author in this project. The author contributions were as follows: MH, Steven L. Brunton (SLB), and Frank Noé (FN) conceived the project. MH designed and implemented the majority of `deeptime`: Architecture of Koopman operator methods (Section 3.3.1) as well as most of their implementations, flexible integration of deep learning components (Section 3.3.2), architecture of MSM and HMM code (Section 3.4) as well as most of its implementation, architecture of the SINDy integration (Section 3.5), architecture and most implementations of example datasets (Section 3.6), architecture of clustering algorithms as well as most of their implementations, flexible integration and implementation of basis and kernel functions, most of top- and low-level documentation, packaging, testing, fixing and improving legacy code. Martin K. Scherer had contributions to the code in the early stages of the project, especially in terms of software architecture as well as refactoring and integrating legacy codes. Tim Hempel (TH) has contributed to tests and architecture of MSM/HMM code. Brooke E. Husic helped with the original structure and organization of the repository. Stefan Klus (SK) contributed some implementations of Koopman operator and Perron–Frobenius operator methods, in particular kernel methods, as well as their documentation and description in the paper. SK contributed some example datasets, particularly for coherent set detection tasks. MH laid out, performed, and visualized the dimension reduction method comparison presented in Section 3.3.3.1. MH and SK laid out the coherent set benchmark presented in Section 3.3.3.2, MH carried out corresponding simulations, analyses, and visualizations. TH contributed to the MSM section (Section 3.4) and implemented, carried out, analyzed, and visualized the therein contained Prinz potential example. Brian de Silva implemented the SINDy method and laid out, performed, analyzed, and visualized the SINDy example presented in Section 3.5. MH laid out, performed, analyzed, and visualized the two-dimensional double-well benchmark presented in Section 3.6. All contributors wrote the paper.

Realizing stochastic processes in computer simulations can yield very high-dimensional and long trajectories. When particle-based models are used and N particles are parameterized only by their position in three-dimensional space (meaning no, e.g., velocities), the dimension of each simulated frame is $d = 3 \cdot N$, making the visual discovery and analysis of dominant processes very difficult. This chapter tries to answer the following question: Given time series data, what is a reduced dynamical model which captures the underlying processes' most important features? Importance here is understood in terms of slowness, e.g., the folding and unfolding process of a protein is deemed more important than its hydrogen bonds' high-frequency vibrations. All of the following methods build on or are related to transfer operators (see Sections 2.1.1–2.1.2) and in particular finite-dimensional approximations thereof.

3.1 Introduction

`deeptime` is an open source Python library for the analysis of time series data; i.e., the provided methods relate to finding relationships between instantaneous data $\mathbf{x}_t$ for some $t \in [0, \infty)$ and corresponding future data $\mathbf{x}_{t+\tau}$ for some so-called *lag-time* $\tau > 0$ based on an underlying stochastic process $\{\mathbf{x}_t\}_{t\geq 0}$ (cf. Section 2.1). Most of the implemented methods try to estimate the behavior of processes when going from $\mathbf{x}_t$ to $\mathbf{x}_{t+\tau}$ by predicting the latter based on the former. The API is similar to what is implemented in the well-known software package scikit-learn [115] and there is basic compatibility of the methods in the two packages via duck typing. Duck typing refers to the objects' behavior defining in which contexts it can be used, not so much its concrete type. `deeptime` has two main goals: (1) making methods which were developed in different communities (such as molecular dynamics and fluid dynamics) accessible to a broad user base by implementing them in a general-purpose way, and (2) easy to extend and maintain due to modularity and very few hard dependencies.

`deeptime` offers the following main groups of methods:

- *Dimension reduction of dynamical data.* One vitally important ingredient to understanding high-dimensional data is projecting them onto a low-dimensional manifold which preserves the "interesting" parts of the signal. One prominent linear method which can perform this task is principle component analysis (PCA) [46, 47]. While PCA is a widely implemented method, it is not designed for extracting dynamically relevant features from time series data. Time-lagged independent component analysis (TICA) [116, 117] and dynamic mode decomposition (DMD) [118, 119] provide dimension reduction along with a best-fit linear model. `deeptime` offers a range of methods which are based on the mathematical framework of transfer operators (see Sections 2.1.1–2.1.2), enabling users to study in particular kinetic properties of the data as well as find temporally metastable and coherent regions.

- *Nonlinear dimension reduction.* While linear methods are widely used due to their simplicity, the high-dimensional data manifold might not always be structured in a way in which important processes are linearly separable. For that reason, `deeptime` offers some featurizations, explicitly defined basis functions, and kernel methods.

- *Deep dimension reduction.* For nonlinear dimension reduction—especially in the high-data regime—there is also a weak dependency (i.e., no strict requirement for installation) to PyTorch [120], enabling the use of deep learning techniques for dimension reduction of time series data. A variety of such methods has recently been developed, for example time-lagged Autoencoders [121], linearly-recurrent Autoencoder networks [122], VAMPNets [49], deep generative Markov state models [123, 124], deep Koopman networks [125], and variational dynamical encoders [126]. Some of the mentioned methods are implemented in this software.

- *Markov state models (MSMs).* MSMs [127–135] are stochastic models describing temporal transitions between states in chains of events where each event only depends on its predecessor and has no dependency on events further in the past (known as the Markov property). They also fit into the mathematical framework of transfer operators. Based on MSMs one can estimate in particular kinetic properties from data.

- *Hidden markov models (HMMs).* HMMs [136, 137] are a type of model consisting of a hidden (i.e., not observable) Markov process emitting an observable output process depending on the hidden process. In comparison to MSMs, HMMs are more expressive and can produce good results where MSMs would not, but are harder to estimate. A more effective/efficient model for hidden Markov processes with discrete output probability distributions is the observable operator model MSM [138] that can also be found within the `deeptime` package.

- *Sparse identification of nonlinear dynamics (SINDy).* SINDy [54] identifies nonlinear governing equations with as few terms as possible from a library of candidate terms that best fit the data. In that way, it complements the dimension-reduction techniques. In particular, while most methods model and analyze the relationships of time-shifted pairs of data, SINDy predicts maps yielding the infinitesimal expected temporal change of the system's current state. On the other hand, SINDy can also predict discrete-time maps by directly relating the system's future state to the system's current state (see Section 3.5 for details).

`deeptime` currently focuses on the domain-agnostic estimation of dynamical models and their analysis in terms of physically relevant quantities describing equilibrium or nonequilibrium behavior. The aim of `deeptime` is not to provide tools specific for a single domain, such as molecular dynamics, but it can be easily combined with python packages that, e.g., load and featurize domain-specific data files in order to prepare such data for analysis with `deeptime` [139–144]. Alternatively there also exist time series analysis packages that are more domain-specific [P5, 49, 50, 145, 146] or implement a subset of `deeptime`'s methods but with a wider range of options and/or more flexibility [147–149]. As the dynamical model and its properties take the center stage in `deeptime`, its aim is also *not* to perform time series forecasting, e.g., for weather or financial data, or clustering, regression, and annotation directly on the dynamical data itself. For these types of tasks there is, e.g., the `sktime` project [150]. `sktime` also provides a curated overview of various projects dealing with time series data: `https://www.sktime.org/en/latest/related_software.html`.

3.2 Design and implementation

`deeptime` is mainly implemented in and available for Python 3.7+ and available for all major operating systems via the Python package index (PyPI) and conda-forge [151] . Some computationally expensive calculations are implemented in C++ using pybind11 [152] or if appropriate using NumPy [153] and SciPy [154].

The API itself is inspired (and largely compatible with) the one used by scikit-learn [115]. In particular, `deeptime` offers `Estimator` classes, which can be fit on data. An important point at which `deeptime`'s implementation is different to what is offered by scikit-learn is the following: a call to `fit` leads to the creation of a `Model` instance; in particular, estimators can be fit multiple times and each time produce an independent model instance (therefore are model factories). Regarding the structure of data they store, `Models` carry the estimation results and are rather simple classes, that are akin to Python dictionaries. If possible, estimators offer a `partial_fit` method that allows the user to continuously update a model with a stream of data. This is particularly useful if the dataset does not fit into the computer's main memory. Additionally, `Models` may also be `Transformers`, meaning they can `transform` data based on the state of the `Model` instance. In such cases the corresponding `Estimator` also implements the `Transformer` interface, dispatching the call to the latest estimated model.

In comparison, in scikit-learn an `Estimator` is also a `Model` and the estimation results are dynamically attached to the estimator instance. Given that our models come with a large variety of attached methods and properties, we deviate from this paradigm to ensure clarity and component separation and to avoid an overcrowded interface. Furthermore, as our `Models` are relatively lightweighted objects that are divorced from the data they have been trained on, it is straightforward to use the Python pickle module for serialization. This way, `Estimator` instances can be re-used on existing models without side effects, fostering `deeptime`'s applicability to parameter studies.

The number of dependencies is kept as low as possible to reduce maintenance efforts. The base functionality of `deeptime` only depends on the established packages NumPy [153], Scipy [154], and scikit-learn [115]. Dependencies to plotting routines (matplotlib [155]) and deep learning components (PyTorch [120]) are optional.

The code is hosted on GitHub (`https://github.com/deeptime-ml/deeptime`) and licensed under LGPLv3, meaning it uses a license with weak copyleft so the library can be used also in proprietary codes. The repository is coupled to the continuous integration service Azure Pipelines, performing automated testing upon changes or proposed changes to the main branch. The project uses the pytest testing framework [156].

The documentation aims for maximal transparency with respect to the implemented methods and the implementation details. To that end, the main methods and their basic usage are explained in Jupyter notebooks [157] with some theoretical background, references, and illustrative examples. The detailed API documentation is generated directly from the Python source code, so that it can be referred to while using the software but also while developing new components or fixing bugs. Furthermore, there are short example scripts for the datasets and selected methods, compiled into example galleries. All this is rendered into HTML and transparently hosted on GitHub pages using Sphinx under https://deeptime-ml.github.io/.

The `deeptime` library is structured in such a way that the entire user interface is exposed at package-level. We structure the (sub-)packages as follows:

- `deeptime.base`: Contains all the basic classes of `deeptime`, in particular the interface definitions for `Estimators`, `Models`, and `Transformers`.
- `deeptime.basis`: A set of basis functions which can be used for SINDy and some of the dimension reduction algorithms as ansatz and/or featurization.
- `deeptime.kernels`: A set of predefined kernels which can be used in kernel methods. Some of these possess subclasses with a `Torch` prefix, indicating that they are PyTorch-ready and support batched evaluation as well as backpropagation.
- `deeptime.sindy`: Contains an implementation of the SINDy estimator (see Section 3.5).

- `deeptime.covariance`: Methods for estimating covariance and autocorrelation matrices from time series data in an online fashion. These are mainly used by some of the decomposition methods (see Section 3.3.1).
- `deeptime.decomposition`: Decomposition methods for time series data (see Section 3.3 for a comprehensive list of implemented estimators).
- `deeptime.markov`: Analysis tools, validators, and estimators for MSMs and HMMs (see Section 3.4).
- `deeptime.clustering`: A collection of clustering/discretization algorithms. These are mostly intended for assigning frames to discrete states (potentially after using one of the dimension reduction algorithms) and subsequently estimating MSMs or HMMs.
- `deeptime.numeric`: A collection of numerical utilities, most notably for eigenvalue problems and regularized inverses of symmetric positive semi-definite matrices.
- `deeptime.data`: A selection of example data on which the algorithms can be tested (see Section 3.6).
- `deeptime.util.data`: Utilities which relate to data processing, e.g., time series specific `DataSet` implementations which can be used in conjunction with PyTorch.

Some of the implementations are based on the molecular-dynamics analysis package PyEMMA 2 [P5, 145] including its dependencies bhmm [158] and msmtools*—modified so that they are no longer dependent on any molecular-dynamics specific libraries and offer greater flexibility—and on the dynamical systems toolbox d3s†. The `deeptime.sindy` package is based on and compatible to PySINDy [149]. The `deeptime.decomposition` package contains an implementation of dynamic mode decomposition (DMD) [118, 119, 159, 160]. For a richer feature set and different variants and flavors of DMD we recommend the PyDMD package [147].

3.3 Dimension reduction and decomposition methods

All of the following methods make use of transfer operators (see Sections 2.1.1–2.1.2). In particular the Koopman operator $\mathcal{K}_{s,t}$ (see Equation (2.10) which propagates observables from time s to time t, the Perron–Frobenius operator $\mathcal{P}_{s,t}$ (see Equation (2.11)) propagating densities from time s to time t, and the reweighted Perron–Frobenis operator $\mathcal{T}_{s,t}$ (see Equation (2.16)), which is often just called "transfer operator".

Depending whether the process is time-homogeneous or time-inhomogeneous, there is—in the time-homogeneous case—no explicit dependency of either of the operators on both s and t but just on the time delta $\tau := t - s$, referred to as lag time (cf. Section 2.1.1). For the remainder of this section we will simplify the notation to $\mathcal{P}$, $\mathcal{T}$, and $\mathcal{K}$ for the Perron–Frobenius, transfer, and Koopman operators, respectively. Also we often wish to consider/use vector-valued feature functions, in which case it is assumed that the transfer operators act component-wise.

3.3.1 Conventional dimension reduction and decomposition

The conventional machine learning estimators for dimension reduction supported by `deeptime` are detailed below. For more thorough introductions to available methods and overviews of their

*`https://github.com/markovmodel/msmtools`

†`https://github.com/sklus/d3s`

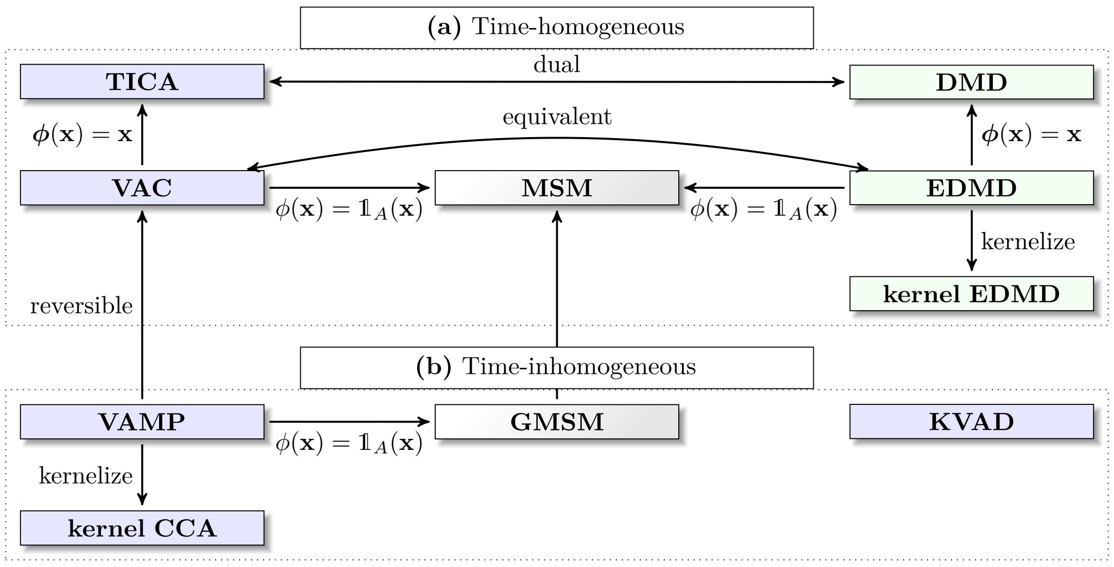

(a) Time-homogeneous
TICA
dual
DMD
equivalent
$\phi(\mathbf{x}) = \mathbf{x}$
$\phi(\mathbf{x}) = \mathbf{x}$
VAC
MSM
EDMD
$\phi(\mathbf{x}) = \mathbb{1}_A(\mathbf{x})$
$\phi(\mathbf{x}) = \mathbb{1}_A(\mathbf{x})$
kernelize
kernel EDMD
reversible
(b) Time-inhomogeneous
VAMP
GMSM
KVAD
$\phi(\mathbf{x}) = \mathbb{1}_A(\mathbf{x})$
kernelize
kernel CCA

Figure 3.1: **Relationships of conventional dimension reduction methods.** Methods shaded in green are regression-based, while methods shaded in blue are based on a variational principle. **(a)** Methods assuming time-homogeneous dynamics. While the variational approach to conformational dynamics (VAC) can only be applied under time-reversible dynamics, extended dynamic mode decomposition (EDMD) makes no assumptions about the dynamics' reversibility. **(b)** Methods supporting time-inhomogeneous dynamics. While the kernel embedding based variational approach for dynamical systems (KVAD) is based on a variational principle, the ansatz and in particular the estimated transfer operator is a Perron–Frobenius operator in contrast to the variational approach for Markov processes (VAMP), which estimates an approximation of the Koopman operator.

relationships, we refer the reader to Refs. 89, 161, 162. Most of the following methods seek a matrix $K \in \mathbb{R}^{m\times m}$, a finite-dimensional approximation of a transfer operator that should fulfill

$$\mathbb{E}[\mathbf{g}(\mathbf{x}_{t+\tau})] = K^\top \mathbb{E}[\mathbf{f}(\mathbf{x}_t)] \tag{3.1}$$

as closely as possible for time series data $\mathbf{x}_t$. The system's state $\mathbf{x}_t$ is transformed into feature space by $\mathbf{f}, \mathbf{g} \in \mathcal{F}^m$, where $\mathcal{F}$ is the space of scalar feature functions.

We give an overview of conventional dimension reduction methods in Figure 3.1, all of which reside in the `deeptime.decomposition` subpackage. Roughly, the methods can be divided into groups of estimators that are restricted to data observed from time-homogeneous systems (Figure 3.1a, see Section 2.1.1) and estimators that are also capable of working with data of time-inhomogeneous systems (Figure 3.1b, see Section 2.1.2).

Another distinction can be made by considering the estimation approach of the respective methods. While some are regression-based (green shade in Figure 3.1), others (purple shade) operate within the framework of an underlying variational principle.

In what follows, we have instantaneous data $\mathbf{x}_i \in \mathbb{R}^d$ and time-lagged data $\mathbf{y}_i \in \mathbb{R}^d$ organized into matrices $X = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ and $Y = [\mathbf{y}_1, \dots, \mathbf{y}_n] \in \mathbb{R}^{d\times n}$, respectively.

3.3.1.0.1 Dynamic mode decomposition (DMD).

DMD [118, 119, 159, 160] was introduced in the fluid dynamics community to extract spatiotemporal coherent structures from high-dimensional time series data. It is closely related to (3.1) in the sense that its objective is to solve the regression problem

$$\min_{M} \|Y - M_{\mathrm{DMD}} X\|_F \tag{3.2}$$

for a matrix $M_{\mathrm{DMD}} \in \mathbb{R}^{d\times d}$. A subsequent spectral analysis of M_{DMD} can reveal information about the dominant dynamics of the system.

There are a variety of extensions to DMD. For example, DMD algorithms have been developed that incorporate control [163], promote sparsity [164], are randomized [165], and act on time delay vectors [166] (the last has a relationship to Koopman operator analysis). Bagheri [167] demonstrated the sensitivity of the DMD algorithm to measurement noise, motivating several noise-robust variants: total least squares DMD [168], forward backward DMD [169], Bayesian DMD [170], optimized DMD [171], and variational DMD [172].

While `deeptime` offers a basic version of DMD, most of these extensions are currently not available. The PyDMD Python package [147] offers a broad range of DMD based methods.

3.3.1.0.2 Extended dynamic mode decomposition (EDMD).

EDMD [173] defines a basis set of functions or observables $B := \{\psi_1, \ldots, \psi_m\} \subset \mathcal{F}$ to construct the vector-valued function $\mathbf{\Psi}(x) = (\psi_1(x), \ldots, \psi_m(x))^\top \in \mathcal{F}^m$. The sought-after matrix K with respect to $\mathbf{f} = \mathbf{g} = \mathbf{\Psi}$ is the solution of the regression problem

$$\hat{K} = \mathrm{argmin}_K \|\mathbf{\Psi}(Y) - K\mathbf{\Psi}(X)\|_F \in \mathbb{R}^{m\times m}, \tag{3.3}$$

where the application of $\mathbf{\Psi}$ is column-wise.

A projection onto dominant processes can be found by applying the eigenfunctions of the Koopman operator deduced from $\hat{K}$ and $\mathbf{\Psi}$ corresponding to the largest eigenvalues to the transformed input data.

The solution of the regression problem (3.3) is an approximate version of the desired property (2.13) for specific choices of $\mathbf{f}$ and $\mathbf{g}$. In `deeptime` this is implemented by the model of the EDMD estimator being a `TransferOperatorModel` (see Figure 3.2).

As its name suggests, DMD can be understood as a special case of EDMD in which the feature basis contains only the identity function, i.e., $\mathbf{\Psi}(\mathbf{x}) = \mathbf{x}$. If we define the set of basis functions to contain indicator functions for a given discretization of the state space, EDMD estimates MSMs (see Figure 3.1 and Section 3.4 for details on MSMs).

3.3.1.0.3 Time-lagged independent component analysis (TICA).

TICA [116] is a linear transformation method which was introduced for molecular dynamics in [117], was independently derived as a method for extracting the slow molecular order parameters by invoking the variational approach for conformation dynamics (see below) [174], and introduced as a method for constructing high-accuracy MSMs in [174, 175]. TICA is designed for time-homogeneous processes and also assumes that the process is reversible, although it may still perform well practically when applied outside these constraints. A process is defined to be reversible if it fulfills the detailed balance condition

$$\mu(\mathbf{x})p_\tau(\mathbf{x},\mathbf{y}) = \mu(\mathbf{y})p_\tau(\mathbf{y},\mathbf{x}) \quad \forall \mathbf{x},\mathbf{y} \in \Omega. \tag{3.4}$$

As a consequence, the transfer (2.13) and Koopman (2.10) operators are identical and therefore self-adjoint. Assuming $\mathcal{K}$ to be a Hilbert–Schmidt operator, this means that (using the Hilbert–Schmidt theorem) there is an eigenvalue decomposition

$$\mathcal{K} = \sum_{i=1}^{\infty} \lambda_i \langle \cdot, \varphi_i \rangle_\mu \varphi_i, \tag{3.5}$$

where φ_i are eigenfunctions with $\langle \varphi_i, \varphi_{i'} \rangle_\mu = \delta_{ii'}$ and λ_i are eigenvalues which are real and bounded by the eigenvalue $\max_i \lambda_i = 1$ with a multiplicity of one (see Ref. [176]).

The objective of TICA is to yield components which are uncorrelated and also maximize the time-autocorrelation under lag-time τ. To this end, one can solve the generalized eigenvalue problem

$$C_{0\tau}\hat{\varphi}_i = \hat{\lambda}_i C_{00}\hat{\varphi}_i, \tag{3.6}$$

where $C_{00} = \frac{1}{n-1}XX^\top$ is the instantaneous covariance matrix and $C_{0\tau} = \frac{1}{n-1}XY^\top$ is the time-lagged covariance matrix. The reversibility assumption leads to $C_{00} = C_{\tau\tau}$ and $C_{0\tau} = C_{\tau 0} = C_{0\tau}^\top$ and therefore eigenvalues $\hat{\lambda}_i \in \mathbb{R}$. Because real numbers possess a total order, we can assume that the eigenvalues $\hat{\lambda}_i$ are in a descending order and the transformation $\hat{\boldsymbol{\varphi}}(\cdot) = [\hat{\varphi}_1(\cdot), \ldots, \hat{\varphi}_k(\cdot)]$ is the TICA projection onto the first k dominant components. The corresponding eigenvalues can be related to relaxation timescales of the processes $\hat{\varphi}_i$ [174]. Therefore, if we know a priori that the system is time-homogeneous and reversible, TICA can be more data-efficient and yield more interpretable results compared to methods, which do not make these assumptions.

For a comparison with DMD it is useful to identify TICA with $M_{\text{TICA}} = C_{00}^\dagger C_{0\tau}$, where $C_{00}^\dagger$ denotes the Moore–Penrose pseudoinverse. It can be seen that $M_{\text{TICA}} = M_{\text{DMD}}^\top$ [89]. Therefore, the TICA transformation consists of Koopman eigenfunctions projected onto the basis spanned by $\psi_k(\mathbf{x}) = \mathbf{x}_k$ and the DMD modes are the corresponding Koopman modes—, i.e., the coefficients $\eta_k = (\eta_{k1}, \ldots, \eta_{kd})^\top$ required to write the k-th component of the full-state observable in terms of eigenfunctions $g_k(\mathbf{x}) = \mathbf{x}_k = \sum_i \eta_{ki}\varphi_i(\mathbf{x})$ [89, 173].

This relationship is reflected in Fig. 3.1 by identifying DMD and TICA as "dual". This duality can also be found within the `deeptime` software: in contrast to DMD, TICA is a subclass of `TransferOperatorModel` (see Figure 3.2).

3.3.1.0.4 Variational approach for conformational dynamics (VAC). Like TICA, VAC [176, 177] assumes time-homogeneous and reversible dynamics. Similar to the generalization from DMD to EDMD, VAC generalizes TICA using a basis $B \coloneqq \{\psi_1, \ldots, \psi_m\} \subset \mathcal{F}$ to construct a transformation $\boldsymbol{\Psi}(\mathbf{x}) = (\psi_1(\mathbf{x}), \ldots, \psi_m(\mathbf{x}))^\top$. Subsequently the instantaneous and time-lagged data is transformed to $\boldsymbol{\Psi}(X)$ and $\boldsymbol{\Psi}(Y)$, respectively, and used in the TICA problem instead of X and Y. From this it becomes clear that TICA can be understood as a special case of VAC with $\boldsymbol{\Psi}(\mathbf{x}) = \mathbf{x}$ (see Figure 3.1). Because it is algorithmically identical to TICA under a prior featurization of data, there is no dedicated VAC estimator in `deeptime`. Under the particular choice of basis functions being indicator functions, VAC estimates MSMs (see Figure 3.1 and Section 3.4 for details on MSMs).

As its name suggests, VAC involves a variational bound. It defines the score $s_{\text{VAC}} \coloneqq \sum_i \hat{\lambda}_i$ which is bounded from above by the sum over the eigenvalues of the true Koopman operator and therefore expresses how much of the slow dynamics is captured in the projection [176, 177]. The score can be used to optimize the feature functions $\boldsymbol{\Psi}$. We will see in the following paragraph that under the assumption of reversible dynamics, the VAC score is equal to the VAMP-1 score, which is why `deeptime` only offers a VAMP score implementation. Assuming reversible dynamics, VAC is equivalent to EDMD (see Figure 3.1).

3.3.1.0.5 Variational approach for Markov processes (VAMP). VAMP [90], sometimes also referred to as "time-lagged canonical correlation analysis" (TCCA) [178], not only optimizes for K but also optimizes for $\mathbf{f}$ and $\mathbf{g}$. This cannot be achieved by merely solving the regression problem (3.3)—as, e.g., the trivial model $\mathbf{f} = \mathbf{g} \equiv (1, \ldots, 1)^\top$, $K = \mathrm{Id}$ is not informative but yields zero error. Instead, VAMP minimizes the left-hand side of

$$\|\mathcal{K} - \hat{\mathcal{K}}\|_{\mathrm{HS}}^2 = -\mathcal{R}(\mathbf{f}, \mathbf{g}) + \|\mathcal{K}\|_{\mathrm{HS}}^2, \tag{3.7}$$

the Hilbert–Schmidt norm of the difference between true Koopman operator (2.10) and approximated Koopman operator $\hat{\mathcal{K}}$ deduced from K, $\mathbf{f}$, and $\mathbf{g}$. The minimization is achieved by maximizing $\mathcal{R}$, a variational score. The decomposition (3.7) of the modelling error assumes that $\mathcal{K}$ is indeed a Hilbert–Schmidt operator.

In [90] it was shown that the smallest approximation error (3.7) is achieved for

$$\hat{\mathcal{K}} = \sum_{i=1}^{m} \sigma_i \langle \cdot, \phi_i \rangle \psi_i, \tag{3.8}$$

where $K = \mathrm{diag}(\sigma_1, \ldots, \sigma_m)$, $\mathbf{f} = (\psi_1, \ldots, \psi_m)$, $\mathbf{g} = (\phi_1, \ldots, \phi_m)$, and σ_i, ψ_i, ϕ_i are the square root of the i-th eigenvalue, left eigenfunction, and right eigenfunction of the forward-backward operator $\mathcal{K}^*\mathcal{K}$, respectively.

During estimation (similar to TICA and VAC), covariance matrices are estimated and under regularization inverted to perform whitening operations to finally construct an approximation of the Koopman operator. One obtains coefficient matrices $U, V \in \mathbb{R}^{m\times k}$ and the matrix $K \in \mathbb{R}^{k\times k}$, so that

$$\mathbb{E}[V^\top \boldsymbol{\chi}_1(\mathbf{x}_{t+\tau})] \approx K^\top \mathbb{E}[U^\top \boldsymbol{\chi}_0(\mathbf{x}_t)], \tag{3.9}$$

where χ_0 and χ_1 are vectors of basis functions which optimally should contain ψ_i and ϕ_i in their span, respectively.

The family of VAMP-r scores,

$$\mathcal{R}_r \coloneqq \sum_i \sigma_i^r, \tag{3.10}$$

as well as the VAMP-E score (see Ref. 90 for a definition) can be optimized to minimize the model error on the left-hand side of (3.7) and therefore can be used to select optimal features and/or observables by using cross-validation techniques (see, e.g., [P6]). These scores give rise to the "variational" aspect of VAMP as they are bounded from above and their maximization leads to better approximations.

VAMP therefore generalizes VAC to a time-inhomogeneous and nonreversible setting (recall Figure 3.1). While VAMP is applicable in more situations, i.e., because it possesses greater generality and nonequilibrium dynamics are more common in nature, it also loses some of its interpretability—as, e.g., singular values can in general no longer by related to relaxation timescales of processes.

The `deeptime` library reflects the mathematics of the VAMP approach by the VAMP estimator producing a `CovarianceKoopmanModel`, an extension of the `TransferOperatorModel`, which in particular allows the evaluation of VAMP scores (see Figure 3.2). The estimator can deal with large amounts of data, because the estimation procedure is based on the decomposition of covariance matrices, which can be constructed incrementally [179]. Furthermore, TICA is a subclass of VAMP, as the two methods are algorithmically closely related.

It should be noted that while TICA and VAC are special cases of VAMP, in `deeptime` the estimators are not combined into one due to differences in how covariance matrices are estimated—in particular, TICA's stronger inductive bias is implemented by forced symmetrizations which are not applicable to VAMP—and differences in the decomposition (eigenvalue decomposition and singular value decomposition for TICA and VAMP, respectively).

Analogously to VAC, the choice of indicator feature functions leads to generalized MSMs (GMSMs) [88], which are also applicable to time-inhomogeneous systems (see Figure 3.1).

3.3.1.0.6 Kernel canonical correlation analysis (Kernel CCA). Kernel CCA [48] is a kernelized version of canonical correlation analysis (CCA) [180] that seeks to maximize the correlation between two multidimensional random variables X and Y (pairs of instantaneous and time-lagged data, respectively). In kernel CCA, the standard inner products are replaced by a kernel function $\kappa(\cdot,\cdot)$ using the "kernel trick". `deeptime` has a subpackage dedicated to kernel implementations (`deeptime.kernels`), containing (amongst others) vectorized versions of the popular Gaussian kernel

$$\kappa(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2}\|\mathbf{x}-\mathbf{x}'\|_2^2/\sigma^2\right) \tag{3.11}$$

as well as the polynomial kernel $\kappa(\mathbf{x},\mathbf{x}') = (c+\mathbf{x}^\top\mathbf{x}')^p$.

It was shown in [161] that kernel CCA can be derived from optimizing the VAMP-1 score (3.10) within a kernel approach and thus can be understood as a kernelized version of VAMP (for this reason it is sometimes referred to as kernel VAMP).

In addition to the kernel parameters, the estimator also possesses a regularization parameter ε, as kernel CCA involves inverting covariance operators (which on their own are generally not invertible).

3.3.1.0.7 Kernel extended dynamic mode decomposition (Kernel EDMD). Kernel EDMD [181, 182] is, analogously to kernel CCA, a kernelized version of EDMD. In contrast to kernel CCA, it assumes a time-homogeneous process. Furthermore, kernel EDMD requires a regularziation parameter ε in order to ensure invertibility of covariance operators.

3.3.1.0.8 Kernel embedding based variational approach for dynamical systems (KVAD). KVAD [94] is an alternative to VAMP which can also be applied to systems in which the transfer operator $\mathcal{T}$ is not Hilbert–Schmidt as an operator from $L^2_{\mu_s}$ to $L^2_{\mu_t}$, which is (e.g.) the case for some deterministic systems. In case of time-homogeneous processes, we have $\mu_s = \mu_t = \mu$, which is the stationary distribution.

To this end, the similarity of functions of interest is not determined using norms of L^2 function spaces but rather using kernel embeddings of said functions. In particular, for a given kernel $\kappa(\mathbf{x},\mathbf{x}') = \langle\varphi(\mathbf{x}),\varphi(\mathbf{x}')\rangle$, functions q can be embedded via

$$\mathcal{E}q = \int \varphi(\mathbf{x})q(\mathbf{x})\mathrm{d}\mathbf{x}. \tag{3.12}$$

The similarity between functions q and q' can then be measured as

$$\|q-q'\|_{\mathcal{E}} = \langle\mathcal{E}(q-q'),\mathcal{E}(q-q')\rangle. \tag{3.13}$$

In Ref. 94 it was shown that for universal and bounded kernels κ, the Hilbert–Schmidt assumption is always fulfilled if the PF operator is considered as

$$\mathcal{P}_\tau : L^2_{\mu_s^{-1}} \to L^2_{\mathcal{E}}, \tag{3.14}$$

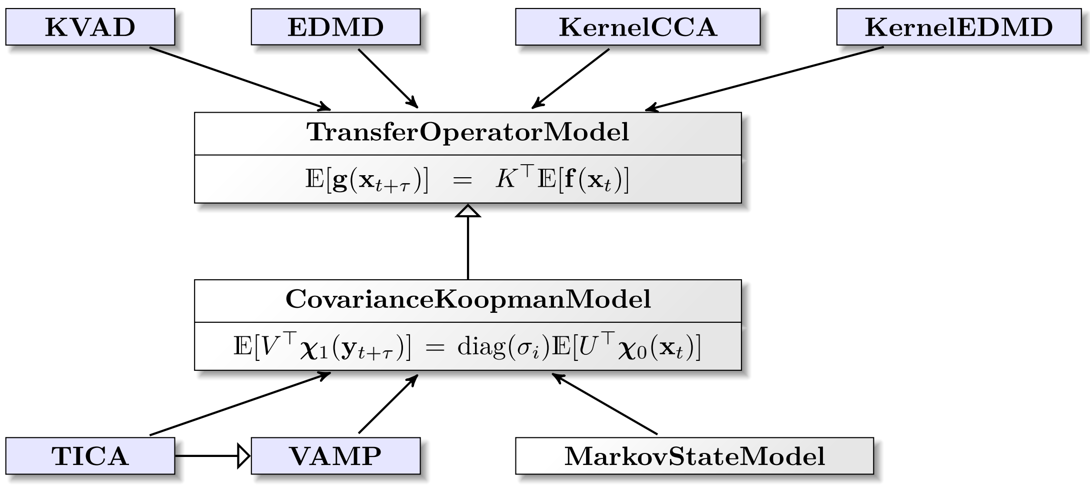

KVAD
EDMD
KernelCCA
KernelEDMD
TransferOperatorModel
$\mathbb{E}[\mathbf{g}(\mathbf{x}_{t+\tau})] = K^\top \mathbb{E}[\mathbf{f}(\mathbf{x}_t)]$
CovarianceKoopmanModel
$\mathbb{E}[V^\top \boldsymbol{\chi}_1(\mathbf{y}_{t+\tau})] = \mathrm{diag}(\sigma_i)\mathbb{E}[U^\top \boldsymbol{\chi}_0(\mathbf{x}_t)]$
TICA
VAMP
MarkovStateModel

Figure 3.2: **Class diagram illustrating relationships between estimators producing approximations of transfer operators.** Estimators have a blue background while models are shaded in gray. The `TransferOperatorModel` implements observable transforms $f(\cdot)$ and $g(\cdot)$ as well as propagation of $f(x_t)$ with the Koopman matrix K. It is produced ($\rightarrow$) by `EDMD`, `KernelEDMD`, `KernelCCA`, and `KVAD` estimators. The the `CovarianceKoopmanModel` extends ($\rightarrow$) the `TransferOperatorModel`. It assumes the estimation to be based on covariance matrices and defines the Koopman matrix in a whitened space, where χ_0 and χ_1 are basis transformations of the state x_t and $x_{t+\tau}$, respectively, and U and V are basis transform matrices. The Koopman matrix is then a diagonal matrix. The `CovarianceKoopmanModel` can be produced by `TICA`, `VAMP`, and `MarkovStateModel`s and additionally possesses a `score()` function.

where $L^2_{\mathcal{E}} = \{f \in L^2 : \|f\|_{\mathcal{E}} < \infty\}$ is an L^2 space equipped with the kernel similarity measure (3.13). Note that in this case the Perron–Frobenius operator as defined in (3.14) is in general no longer the adjoint of the Koopman operator.

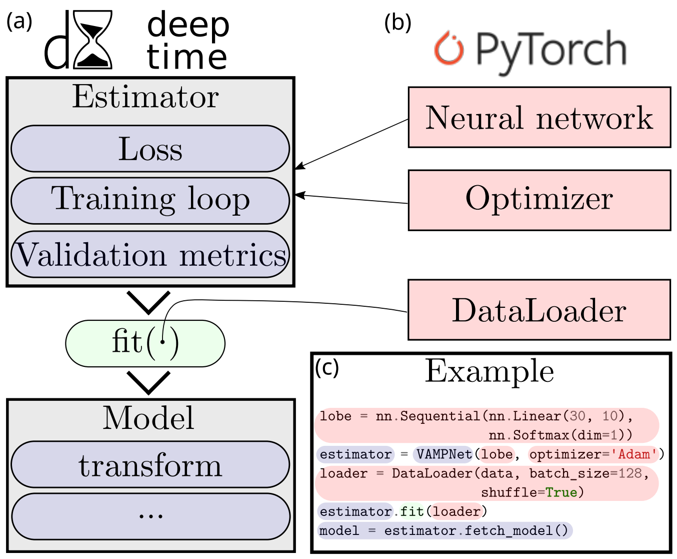
Like VAMP, KVAD is based on the optimization of a (variational) score that is bounded from above and expresses the quality of the found approximation. A key difference is the ansatz: While VAMP yields approximations of the Koopman operator, KVAD estimates its adjoint, the PF operator (adjoint in the sense of Equation (2.12)). To this end, KVAD uses the transition density (2.5) and assumes that it can be represented as

$$\hat{p}_\tau(\mathbf{x}_t, \mathbf{x}_{t+\tau}) = \mathbf{f}(\mathbf{x}_t)^\top \mathbf{q}(\mathbf{x}_{t+\tau}), \tag{3.15}$$

where $\mathbf{q} = (q_1, \ldots, q_m)^\top$ are m density basis functions and $\mathbf{f}$ are, as in (3.1), feature functions of the system's state. This leads to the linear model (3.1) with $\mathbf{f} = \mathbf{g}$ and

$$K = \int \mathbf{q}(\mathbf{y})\mathbf{f}(\mathbf{y})^\top \mathrm{d}\mathbf{y}.$$

It has been shown [94] that $\mathbf{q}$ can be estimated directly from data in a nonparametric fashion, which means that all the model's parameters reside inside the definition of $\mathbf{f}$. With the help of estimated $\mathbf{f}$ and $\mathbf{q}$, also the transition matrix K can be constructed. This kind of ansatz—sans the modified codomain in (3.14)—is similar to what was used in Ref. 123.



```python
lobe = nn.Sequential(nn.Linear(30, 10),
                     nn.Softmax(dim=1))
estimator = VAMPNet(lobe, optimizer='Adam')
loader = DataLoader(data, batch_size=128,
                    shuffle=True)
estimator.fit(loader)
model = estimator.fetch_model()
```

Figure 3.3: **Schematic overview of the flexible integration of deep learning components in deeptime.** **(a)** `deeptime` interface: Some `Estimator` instances in `deeptime`, for example VAMPNets and Time-lagged Autoencoders, take neural networks as configurational input. These estimators offer a `fit( )` method which takes PyTorch data loaders. The data loaders are configured by the users and manage, e.g., shuffling and batch sizes. After a `Model` has been fit, it can be used for further analysis of the input data. Models containing deep learning components are designed so that they can also seamlessly work with NumPy arrays and interact with the rest of the `deeptime` library. **(b)** PyTorch interface: Neural networks can be defined using PyTorch and also be trained using optimizers which either are already included in PyTorch or at the very least are PyTorch compatible. Instances thereof can be used with certain estimators in `deeptime`. **(c)** Example: A typical workflow for a deep learning estimator, color-coded according to which library the classes and respective instances belong.

3.3.2 Deep dimension reduction and decomposition

In addition to the conventional learning methods introduced in Section 3.3.1, `deeptime` also offers several deep learning methods for dimension reduction.

The deep learning components require PyTorch; however, PyTorch-dependent parts of the library are separate—i.e., a working installation of PyTorch is not required for the rest of the library. The estimators providing deep dimension reduction can be found in the `deeptime.decomposition.deep` subpackage.

Deep learning requires some additional flexibility and data handling compared to conventional learning. In particular, the user first must define a neural network architecture and an optimizer for adjusting the network's weights. There are some predefined architectures directly available in `deeptime` (such as multilayer perceptrons), but in principle these as well as the optimizer are defined with PyTorch (see Figure 3.3b). Once defined, one can construct a deep estimator that contains losses, validation metrics, and training procedures (see Figure 3.3a). Fitting deep learning components typically involves shuffling and dividing the data into batches. Since the optimal batch size and also the shuffling method are problem-dependent, these choices must be made by the user. PyTorch offers `DataLoaders` for this exact purpose. Therefore `estimator.fit` is performed on a data loader instance rather than arrays (see Figure 3.3a). Finally, deep learning estimators also produce models which encapsulate among other things a copy of the trained neural network. While PyTorch neural networks operate on `torch.Tensor` instances, `deeptime` models with deep learning components are designed so that they can also work with ordinary NumPy arrays, ensuring a seamless integration with other and in particular conventional models and methods. For more details about PyTorch, see the official documentation `https://pytorch.org/`.

3.3.2.0.1 VAMPNets. VAMPNets [49] are a deep learning approach that seek to find parametrizations of neural networks $\boldsymbol{\chi}_0$ and $\boldsymbol{\chi}_1$ (referred to as "lobes") so that the VAMP-E or one of the VAMP-r scores (3.10) under these transformations is maximized, leading to smaller model errors (recall paragraph about VAMP in Section 3.3.1). This is possible because there is a variational upper bound to the scores and their computation is differentiable—therefore any of the VAMP scores can be used directly as an objective function in a deep learning context.

Other deep learning methods which are not currently included in `deeptime` but also approximate the Koopman operator are, e.g., those found in Refs. 122, 125, 183, 184.

3.3.2.0.2 KVADNets. Analogously to VAMPNets, KVADNets optimize the variational KVAD score to find an optimal parametrization of feature functions $\mathbf{f}$ (3.15). As in the case of VAMPNets, the KVAD score is differentiable [94].

3.3.2.0.3 Time-lagged (variational) autoencoders (T(V)AEs). TAEs [121] are a type of neural network approach in which instantaneous data $\mathbf{x}_t \in \mathbb{R}^d$ is compressed / encoded through a parameterized function

$$E : \mathbb{R}^d \to \mathbb{R}^n, \mathbf{x}_t \mapsto E(\mathbf{x}_t) = \mathbf{z}_t$$

with $n \leq d$ and then reconstructed as time-lagged data $\mathbf{x}_{t+\tau}$, $\tau > 0$ via a decoder network

$$D : \mathbb{R}^n \to \mathbb{R}^d, \mathbf{z}_t \mapsto D(\mathbf{z}_t) \approx \mathbf{x}_{t+\tau}.$$

The optimization target is to reduce the mean-squared error between $\mathbf{x}_{t+\tau}$ and $(D \circ E)(\mathbf{x}_t)$, effectively training a latent and lower-dimensional representation $E(\mathbf{x}_t)$ of the process. In [121] it was shown that in the linear case TAEs perform time-lagged canonical correlation analysis, cf. the paragraph about VAMP in Section 3.3.1. An architecture that is akin to the one of TAEs was used in [185] to find collective variables in the context of molecular enhanced sampling.

A natural extension to TAEs is to exchange the neural network architecture of an autoencoder by the architecture of a variational autoencoder (VAE) [186, 187], yielding the generative TVAE that can also be found in the `deeptime` library. In [126] these architectures (there called "variational dynamics encoder" (VDE)) were used in conjunction with a loss term inspired by saliency maps [188] (known from computer vision) to produce interpretable dynamical models while still maintaining the high degree of nonlinearity that can be achieved by neural networks.

3.3.3 Numerical experiments

We compare some of the dimension reduction methods introduced in Section 3.3.1 and Section 3.3.2. The first example highlights differences in the approximation if used for dimension reduction in a time-homogeneous system. The second example uses data obtained from a time-inhomogeneous system with the objective to find coherent structures.

3.3.3.1 Dimension reduction

We consider a two-state hidden Markov model (see Section 3.4) with transition matrix

$$P = \begin{pmatrix} 0.95 & 0.05 \\ 0.05 & 0.95 \end{pmatrix} \tag{3.16}$$

and anisotropic but linearly separable two-dimensional Gaussian emission distributions. In a subsequent step the data is transformed via

$$(x, y) \mapsto (x, y + \sqrt{|x|}). \tag{3.17}$$

This leads to two wedge-shaped output distributions which are no longer linearly separable (see Figure 3.4). We simulate a trajectory of $T = 1000$ frames from this model and try to recover a separation into the two original hidden states using different dimension reduction methods by projecting onto the dominant slow process, which is the jump process between the two wedges.

Figure 3.4 (a,b,c,g,h,i) shows the sampled and transformed time series data $\mathbf{x}_t \in \mathbb{R}^2$ as a scatter plot. The estimated models yield two-dimensional decision landscapes $\chi : \mathbb{R}^2 \to \mathbb{R}$ which are shown in the background as a filled contour. Based on the decision landscape we obtain crisp assignments to one of the two states via k-means [189] clustering with $k = 2$ cluster centers in the projected space; these clusters determine the point colors in the scatter plot. In Figure 3.4 (a,b,c,g,h,i) we furthermore report the 10-fold cross-validated VAMP-2 score along with its standard deviation (see Section 3.3.1 for the score and Ref. 190 for the cross-validation scheme), which enables a quantitative assessment of the quality of the projection χ.

Figure 3.4 (d,e,f,j,k,l) shows the projection of the two-dimensional time series onto one dimension for the first 200 frames of the trajectory, where $\chi(x_t)$ is presented in transparent blue with corresponding crisp clustered assignments in opaque blue and the hidden reference state is presented in orange. We also report the assignment accuracy of the crisp state with respect to the hidden reference state over the entire dataset in the titles of subfigures (a-f).2. While the assignment accuracy can be used as another measure of the quality of the projection, it is only available if the ground truth is known. A VAMP score, on the other hand, can always be evaluated. Here, we chose the VAMP-2 score, because its maximization can be identified with the maximization of kinetic variance (see Ref. 191). Maximizing kinetic variance achieves an optimal separation of metastable sets, which corresponds to the separation of the two wedges in this example.

In the limit of infinite data (i.e., when faithfully representing the original data distribution) and optimal featurization, the score should approach $s_{\text{lim}} = 1.81$. This can be found from using the ground truth hidden transition matrix (3.16) and applying it to the VAMP score assuming that the distribution of data is given by the stationary distribution. In more detail, if $\boldsymbol{\mu} = (0.5, 0.5)^\top$ is the stationary distribution corresponding to (3.16), we assume that data distribution is given by the stationary distribution and set covariance matrices $C_{00} = C_{\tau\tau} = \text{diag}(\boldsymbol{\mu})$ and the cross-covariance matrix to $C_{0\tau} = P$ (in this example $\tau = 1$). From the covariance matrices we can obtain the Koopman matrix (cf. Section 3.3.1) which can be decomposed and used for scoring.

Below, we discuss and further describe each of the panels in Figure 3.4.

(a) *TICA*. TICA is a linear method in that it can only draw linear decision boundaries and the dataset is deliberately not linearly separable. Therefore the tip of the upper wedge and the outer areas of the lower wedge are misclassified. This is also reflected in the comparably low VAMP-2 score and accuracy.

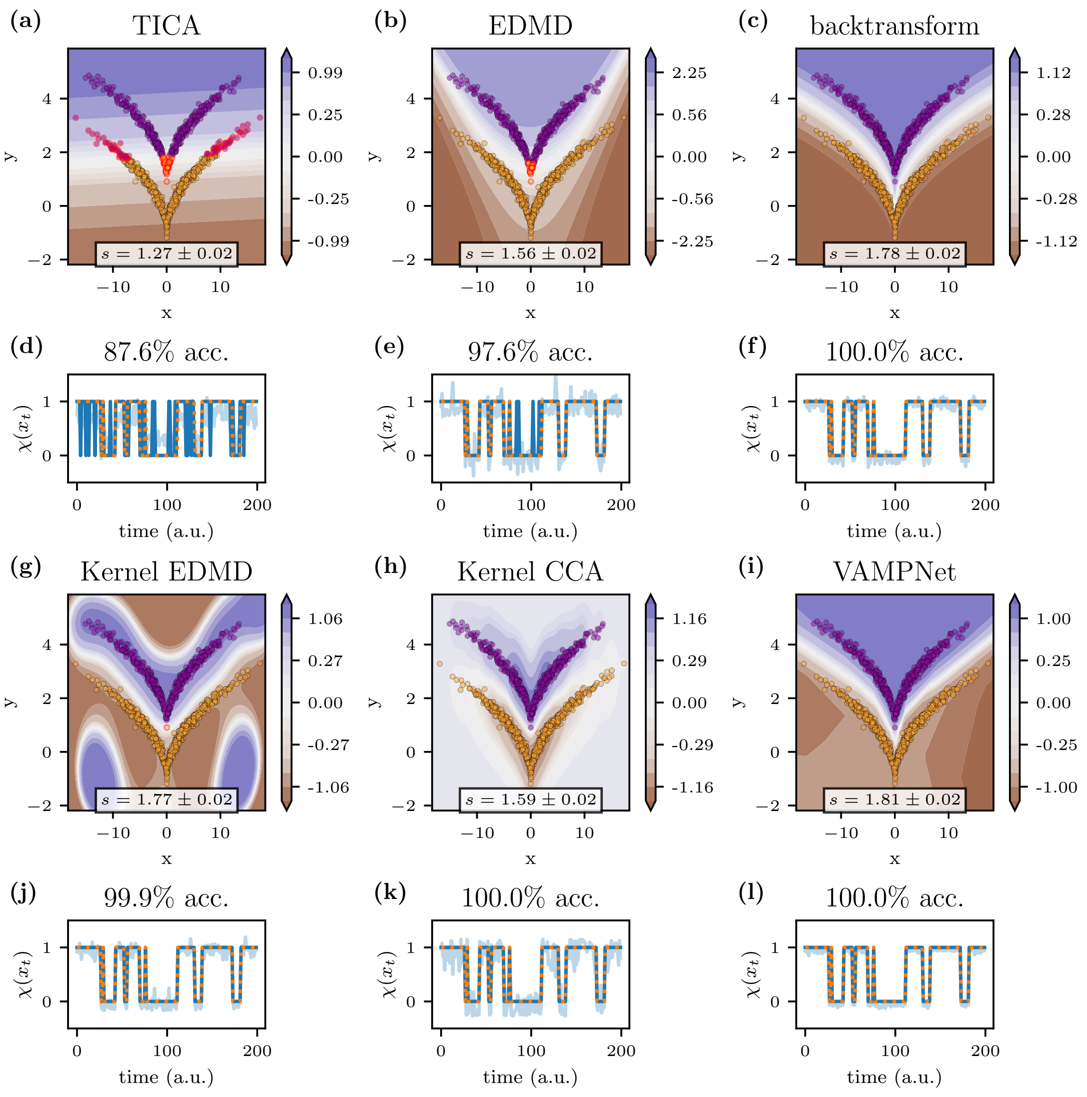


Figure 3.4: **Comparison of different Koopman operator methods.** The data is generated by a HMM with two hidden states and respective two-dimensional output probability distributions which are not linearly separable. The methods should approximately recover the underlying hidden process, where in **(a,b,c,g,h,i)** the background contour is the decision landscape and the scatter colors denote sharp assignments obtained by a two-state clustering in the projection. Incorrectly assigned states have a red edge. The scores reported in the bottom parts of the plots are cross-validated VAMP-2 scores. Plots **(d,e,f,j,k,l)** show a projection of the two-dimensional time series onto one dimension (transparent blue) with crisp assignments (opaque blue) and the ground truth (orange) as reference. The accuracy (acc.) refers to the amount of correctly assigned states after clustering.

(b) *EDMD.* We choose EDMD with an ansatz basis of monomials up to degree two in two-dimensional space; i.e.,

$$B = \{(x, y) \mapsto x^p y^q : p, q \in \mathbb{N}_{\geq 0},\ p + q \leq 2\}\,.$$

This leads to a decision landscape shaped like a rounded cone, able to separate most of the data into the two hidden states except for the tip of the upper wedge. Consequently, score and accuracy achieve a higher value than the one obtained from TICA.

(c) *Backtransform.* Here we use the hand-tailored transformation $(x, y) \mapsto (x, y - \sqrt{|x|})$, which makes the two states linearly separable again and apply VAMP. This featurization uses the ground truth as prior knowledge and therefore achieves perfect state separation. Consequently, the accuracy is at 100% and the VAMP-2 score reaches a high value. Due to finite data it does not quite reach the theoretical limit of $s_{\mathrm{lim}} = 1.81$.

(d) *Kernel EDMD.* We use kernel EDMD with a Gaussian kernel (3.11). The regularization parameter ε of the estimator as well as the bandwidth σ of the kernel are tuned to maximize the VAMP-2 score on a validation set using the SLSQP optimizer [192], yielding $\sigma \approx 1.42$ and $\varepsilon \approx 6.7 \times 10^{-4}$. The method finds a good separation between the two hidden states.

(e) *Kernel CCA.* As in the kernel EDMD case, we choose for kernel CCA a Gaussian kernel (3.11) with regularization parameter and bandwidth tuned to maximize the VAMP-2 score on a validation set using the SLSQP optimizer [192]. This leads to $\sigma \approx 0.85$ and $\varepsilon \approx 0.36$. Compared to the other methods, the support of the estimated singular functions is smaller and in particular does not extend far beyond the area spanned by the sample data. This means that according to kernel CCA, there is large uncertainty as to which state a point in space belongs to as soon as it is outside the densely populated areas of the wedges. On the other hand, the score is lower compared to kernel EDMD or VAMPNets. This means that the metastable sets are separated less clearly, which can also be observed in the fuzziness of the transparent blue trajectory in Figure 3.4k and therefore the slow dynamics of the system are not represented as well as they are represented with, e.g., kernel EDMD.

(f) *VAMPNets.* As an architecture for the lobe $\boldsymbol{\chi}$ we choose a multilayer perceptron of depth $d = 5$ with a rectified linear unit (ReLU) nonlinearities and 15, 10, 10, 5, and 1 neurons, respectively. The network is trained using the Adam optimizer [193] with a learning rate of 10^{-3}. We obtain a decision landscape that resembles the one of the backtransform with a perfect state separation. Also the idealized VAMP-2 score s_{lim} based on the hidden transition matrix is within the standard deviation of the VAMPNet VAMP-2 score. The hyperparameters were chosen heuristically so that training was stable and yielded high scores.

For the optimization of the parameters of kernel EDMD we found it crucial to first whiten the data by removing the empirical mean $\boldsymbol{\mu}$ and transforming it into the PCA basis via

$$\mathbf{x}_t \mapsto C^{-\frac{1}{2}}(\mathbf{x}_t - \boldsymbol{\mu}), \tag{3.18}$$

where C denotes the covariance matrix over the trajectory. The other methods were numerically more stable and applicable directly to the raw data. Whether whitening is required does not only depend on the method but in particular also on the chosen ansatz.

3.3.3.2 Coherent set detection

Here, we illustrate how the introduced decomposition methods can be used to detect coherent sets; i.e., sets of particles which are geometrically consistent under a forward-backward dynamic and small perturbations [194, 195]. Following Ref. 195, one can quantitatively describe coherent sets $A \subset \Omega$ under the transfer operator $\mathcal{T}$ (see, e.g., (2.16)) as a set which is difficult to leave, i.e.,

$$\left\langle \mathcal{T}^*\mathcal{T}\frac{1_A}{\mu_s(A)}, 1_A \right\rangle_{\mu_s} \approx 1, \tag{3.19}$$

the probability of staying within A under the forward-backward dynamic [195] should be close to 1. Here, $\mu_s(A)$ refers to the evaluation of the measure induced by the initial distribution.

While dominant eigenfunctions of methods assuming time-homogeneous dynamics (cf. Figure 3.1) can be related to metastable sets, methods that may also be applied to time-inhomogeneous dynamics yield coherent sets [88]. In particular, metastable sets can be understood as a special case of coherent sets.

Practically, these sets can be obtained by projecting Lagrangian data into the dominant (with respect to magnitude of singular values) left singular function space of an approximated Perron–Frobenius or Koopman operator [161, 194], as these singular functions correspond to eigenfunctions of backward-forward dynamic $\mathcal{T}\mathcal{T}^*$ or forward-backward dynamic $\mathcal{T}^*\mathcal{T}$ [90, 161, 194] and therefore are an important ingredient for characterizing coherence (3.19). Spatial proximity in the singular function space indicates membership of the same coherent set.

As an application example we choose the Bickley jet, an idealized and periodically perturbed approximation of stratospheric flow which is described by a deterministic but non-autonomous system of ODEs [52, 53]. The ODEs act on particles $\mathbf{x} = (x, y) \in \Omega = [0, 20] \times [-4, 4]$ and are given by

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} -\frac{\partial \Psi}{\partial y} \\ \frac{\partial \Psi}{\partial x} \end{pmatrix} \tag{3.20}$$

with stream function

$$\begin{aligned} \Psi(x, y, t) = c_3 y - U_0 L \tanh(y/L) + A_3 U_0 L \operatorname{sech}^2(y/L) \cos(k_1 x) \\ + A_2 U_0 L \operatorname{sech}^2(y/L) \cos(k_2 x - \sigma_2 t) \\ + A_1 U_0 L \operatorname{sech}^2(y/L) \cos(k_1 x - \sigma_1 t) \end{aligned}$$

and parameters chosen as in [195]. The domain Ω is quasi-periodic in x-direction. The Bickley jet is widely used as a benchmark problem in the coherent set literature, e.g., in Refs. 161, 194–198.

We expect to find a separation into nine coherent sets, where the domain Ω is separated into an upper Ω_{up} and lower Ω_{low} part with three circular coherent sets each, a coherent layer that is between Ω_{up} and Ω_{down} and the remainder of Ω_{up} and Ω_{low} sans the circular coherent sets, as illustrated in any of the panels of Fig. 3.5 column 4.

Because the ODE is not autonomous (meaning $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x})$ depends on time t), we restrict ourselves to methods that support time-inhomogeneous dynamics, in particular kernel CCA, VAMP, VAMPNets, KVAD, and KVADNets. In order to fit the respective Koopman models, we first integrate $N = 3000$ particles whose positions are drawn uniformly in Ω from $t_0 = 0$ to $t_1 = 40$. From the resulting trajectories we use the initial time particle position matrix $X \in \Omega^N$ and final time particle position matrix $Y \in \Omega^N$ to find an embedding with corresponding Koopman or Perron–Frobenius operator that describes transport from $\mathbf{x}_i$ to $\mathbf{y}_i$.

The visualization of the first three estimated dominant singular functions already reveals some of the coherent structure of the underlying process (see Figure 3.5 columns 1–3). All of

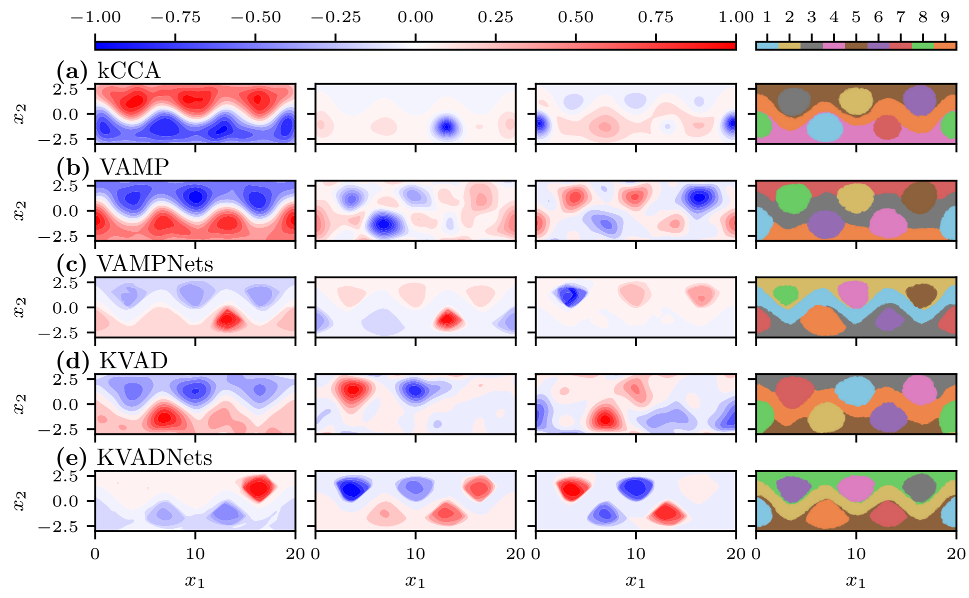


Figure 3.5: **Comparison of different Koopman operator methods for coherent set detection.** Columns 1–3 show the first, second, and third dominant singular function of the respective estimated Koopman operators. Column 4 shows a k-means clustering with $k = 9$ on the initial data after it has been transformed by the first nine respective singular functions. **(a)** Kernel CCA with a Gaussian kernel with bandwidth and regularization parameter optimized to maximize the VAMP-2 score. **(b)** Variational approach for Markov processes (VAMP)-estimated model, where the ansatz featurization consists of a set of randomly shifted and distorted Gaussian functions $f(x) = \exp(-x^2)$. **(c)** VAMPNets with multilayer perceptron lobes. **(d)** KVAD with the same ansatz as VAMP and a Gaussian kernel with bandwidth $\sigma = 1$. **(e)** KVADNets with a Gaussian kernel with bandwidth $\sigma = 0.5$ and a feature transformation given by multilayer perceptrons.

the methods yield similar results and, with different degrees of sharpness, each show the three vortices in the upper part and lower part of the domain.

To obtain crisp assignments to for a predefined number of coherent sets we perform k-means clustering using kmeans++ initialization with $k = 9$ cluster centers with one cluster center belonging to exactly one coherent set. The clustering is repeated 500 times and we select the cluster centers which yield the smallest cumulative squared distance between sample points and assigned cluster center (sometimes referred to as "inertia"). In the last column of Figure 3.5, the particle positions at $t = 0$ are color-coded according to their cluster membership.

For both VAMP and KVAD we set up the feature functions in the following way: Weight matrices $W_1 \in \mathbb{R}^{100\times 3}$, $W_2 \in \mathbb{R}^{50,100}$ are generated by drawing i.i.d. samples from the normal distribution $\mathcal{N}(0,1)$ and bias vectors $b_1 \in \mathbb{R}^{100}$, $b_1 \in \mathbb{R}^{50}$ are generated by drawing i.i.d. samples from the uniform distribution $\mathcal{U}(-1,1)$. The vector-valued feature function is then given by

$$F : \mathbb{R}^2 \to \mathbb{R}^{50}, \quad \mathbf{x} \mapsto W_2\sigma(W_1T(\mathbf{x}) + b_1) + b_2,$$

where $\sigma(x) = \exp(-x^2)$ acts component-wise and T is a transformation that embeds the two-dimensional data into three dimensions by mapping it onto a cylinder

$$T : \mathbb{R}^2 \to \mathbb{R}^3, \quad \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} \cos(2\pi x/20) \\ \sin(2\pi x/20) \\ y/3 \end{pmatrix}, \tag{3.21}$$

accounting for the quasi-periodicity of the domain Ω. KVAD is equipped with a Gaussian kernel with bandwidth $\sigma = 1$.

For VAMPNets, the instantaneous and time-lagged lobes are each a multilayer perceptron (MLP) with shared weights. The two-dimensional data is first transformed into three dimensions to account for quasi-periodicity in x direction via (3.21) and subsequently transformed through a batch normalization layer. The MLPs possess layers with 256, 512, 128, 128, and 9 neurons, respectively, separated using ELU nonlinearities and dropout ($p = 50\%$).

In the case of KVADNets there is per definition just one lobe. Its architecture is the same as for VAMPNets.

All models project onto the dominant nine singular functions.

(a) Kernel CCA. We use a Gaussian kernel where the bandwidth and regularization parameter maximize the VAMP-2 score ($\sigma \approx 0.58$, $\varepsilon \approx 5.6 \cdot 10^{-3}$). Optimization was performed with SLSQP [192]. The first singular function shows a clear separation between upper and lower part of the domain as the dominant process. The vortices are circular in shape and can be observed in the evaluation of the singular functions as well as the clustering.

(b) VAMP. The results are qualitatively comparable to the ones of kernel CCA, however the clustering is less pronounced.

(c) *VAMPNets.* Some of the coherent structures are clearly visible in the first three singular functions. The clustering is pronounced; however, it yields vortices of varying sizes.

(d) KVAD. We use a Gaussian kernel with bandwidth $\sigma = 1$. The results are qualitatively comparable to VAMP.

(e) *KVADNets.* Here the vortices can easily be detected in the singular functions; however, the shape is less circular compared to the other methods. Furthermore, the first singular function has a less pronounced decision surface between upper and lower part of the domain; rather, it almost exclusively describes the exchange of mass between two individual vortices. The clustering, however, is sharp and is comparable to the other methods in terms of the detected sets.

While differences in estimated coherent sets can be evaluated qualitatively by visual inspection, we now seek to compare the methods in a quantitative fashion and try to determine a "best" subdivision into coherent sets according to some criterium.

To this end, we define a "coherence score". Let $\mathbf{x}_t = \mathbf{\Phi}_{t_0,t}(\mathbf{x}_0)$ be the flow describing the solution of the governing equations given an initial position $\mathbf{x}_0$ at initial time t_0. Since in this example the ground truth dynamics are known, we can take our definition of a coherent set (3.19) as template. For a subdivision of Ω into disjoint coherent sets $\bigcup_i A_i = \Omega$, the score restricted to one set A_i is defined as

$$s_{\mathrm{coh}}^{(i)} \coloneqq \mathbb{P}\left[(\mathbf{\Phi}_{t_0,t_1}^{-1} \circ \mathbf{N}_\sigma \circ \mathbf{\Phi}_{t_0,t_1})(\mathbf{x}_{t_0}) \in A_i \mid \mathbf{x}_{t_0} \in A_i\right], \tag{3.22}$$

where $\mathbf{N}_\sigma(\mathbf{x}) \coloneqq \mathbf{x} + \sigma\boldsymbol{\eta}$ distorts the forward-mapped $\mathbf{x}_0$ by white noise $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_d)^\top$, $\eta_i \sim \mathcal{N}(0,1)$ with standard deviation σ. In this example, we chose $\sigma = 10^{-1}$. In other words, Equation (3.22) describes the probability of a particle staying inside set $A_i \subset \Omega$ under propagation forward in time, addition of noise, and subsequent back-propagation to its initial time. This concept is illustrated in Figure 3.6a. Following the arrows in the figure, it depicts a subset A_i at $t_0 = 0$ in blue and red, the remainder is colored in light gray. The particles are then propagated according to the flow $\mathbf{\Phi}_{t_0,t_1}$ to the final time $t_1 = 40$ (upper right panel). The map N_σ is applied, yielding slightly different particle positions at t_1 (lower right panel). Subsequently, the distorted particles are mapped back to $t_0 = 0$. As one might expect, the particles leaving A_i aggregate at the set's boundary. The figure uses the subdivision of the domain that is yielded by the KVADNets trained transfer operator (see Figure 3.5e).

Finally, Figure 3.6b shows the forward-backward mapping for each of the coherent sets. For clarity, we do not distinguish leaked particles from the respective sets but only show them as generally leaked from any of the sets. It can be observed that most of the interior area of the detected vortices remains vacant of leaked particles.

In order to arrive at one value for all estimated coherent sets, we consider the expectation

$$s_{\mathrm{coh}} \coloneqq \mathbb{E}_{\mu_{t_0}}\left[s_{\mathrm{coh}}^{(i)}\right] = \sum_i \frac{\mu_{t_0}(A_i)}{\mu_{t_0}(\Omega)} s_{\mathrm{coh}}^{(i)}. \tag{3.23}$$

For practical evaluation of the score, we estimate a MSM without a reversibility constraint (see Section 3.4) on $n = 2500$ discrete trajectories with nine states corresponding to the coherent sets, each trajectory corresponding to one individual particle and containing exactly two entries; namely, the coherent set before and after application of the forward-backward dynamic as given in (3.22). Then, the ith diagonal entry of the transition matrix is exactly the coherence score (3.22) corresponding to the ith coherent set A_i. The distributions $\mu_{t_0}(A_i)$ and $\mu_{t_0}(\Omega)$ are the empirical distributions; i.e., the number of particles initially inside set A_i and the total number of particles N, respectively.

A drawback of this score is that it does not indicate how faithfully the discovered coherent sets represent the system's dynamics. If, for example, the entire domain is its own coherent set, the score is maximized. Therefore, the score (3.23) should be considered in conjunction with other indicators such as the VAMP or KVAD score.

We compare these metrics in Table 3.1. For all used methods of this example it shows the coherence score, the VAMP-2 score, and the KVAD score calculated with a Gaussian kernel with bandwidth $\sigma = 0.5$.

The table also reports standard deviations, which are obtained by repeating the scoring for fifteen rounds of $n = 2500$ independently and over the domain uniformly sampled initial particle positions. According to the coherence score, KVADNets deliver the best subdivision into coherent sets, with kernel CCA as a close second. This is also reflected in their respective

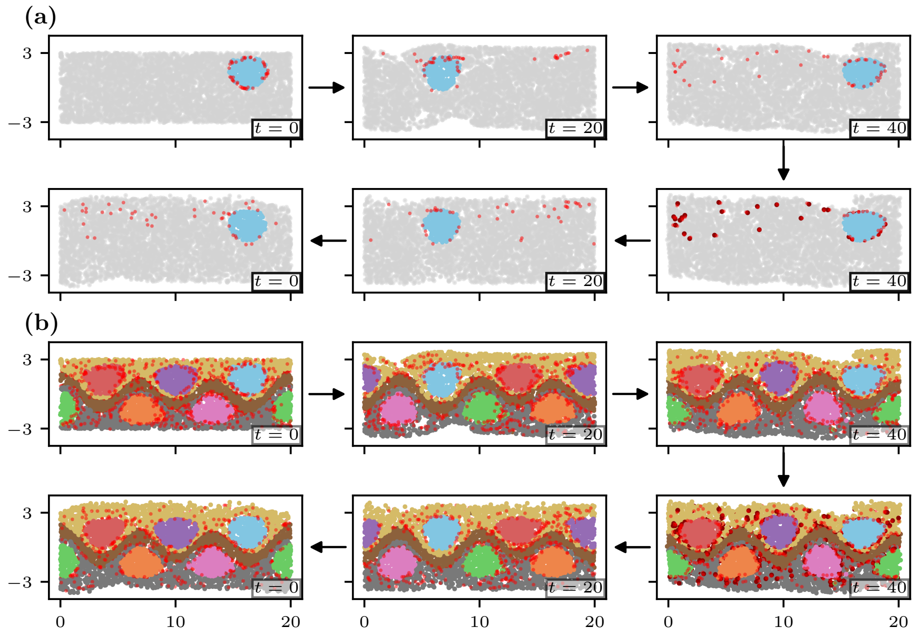


Figure 3.6: **Coherence in the Bickley jet.** This figure demonstrates the concept of "leaked" particles used for defining a coherence score. Particles are colored according to their coherent set membership or marked as leaked (red dots). **(a)** Particles that stay assigned to a particular coherent set (blue dots) and particles that change their set membership under the forward-backward transformation with noise (red dots) as detected by KVADNets. The top row depicts application of forward dynamics, bottom row application of noise and backward dynamics, respectively. Particles initially not belonging to the considered coherent set are shown in grey. **(b)** Generalization of (a) depicting all coherent sets; red dots now depict particles leaking from *any* coherent set.

Table 3.1: **Coherence scores on Bickley jet.** Table showing different kinds of scores with standard deviations for different methods used for coherent set detection using the Bickley jet example (Section 3.3.3.2). For the evaluation of the KVAD score, a Gaussian kernel with bandwidth $\sigma = 0.5$ was chosen. The methods are in ascending order from left to right according to their coherence score.

	KVAD	VAMP	VAMPNets	Kernel CCA	KVADNets
Coherence score	0.74 ± 0.01	0.77 ± 0.01	0.79 ± 0.01	0.85 ± 0.01	0.87 ± 0.01
VAMP-2 score	4.63 ± 0.06	5.18 ± 0.08	7.28 ± 0.06	5.77 ± 0.08	6.03 ± 0.09
KVAD score	$0.070 \pm 1.2 \cdot 10^{-3}$	$0.073 \pm 1.1 \cdot 10^{-3}$	$0.078 \pm 1.1 \cdot 10^{-3}$	$0.080 \pm 1.4 \cdot 10^{-3}$	$0.087 \pm 1.2 \cdot 10^{-3}$

VAMP-2 and KVAD scores. In contrast to KVAD, which yields the same sequence of methods (if ordered ascendingly) as the coherence score, the VAMP-2 score for VAMPNets is an outlier. The VAMP-2 score for VAMPNets is significantly higher compared to any of the other methods. The Bickley jet is a deterministic system and therefore the Koopman operator associated to it is not Hilbert–Schmidt—a violation of the assumptions that are made to define the VAMP scores. Up to noise effects caused by numerical integration, this might be the cause of the high score of VAMPNets.

It should be noted that the coherence score can still be approximated if the ground truth is not known or too expensive to compute by using a propagation model of the form (3.1). Assuming a good representation of the slow dynamics (which is indicated by a high VAMP or KVAD score), the error of integrating backwards in time with (3.1) is small.

3.4 Markov state models

Markov state models (MSMs) are stochastic models describing the time evolution of a random process $\{\mathbf{x}_t\}_{t\geq 0}$, $\mathbf{x}_t \in \Omega$ (see Refs. 127–135) and describe Markov chains with memory depth of 1. In other words, given a sequence $(..., \mathbf{x}_{t-2\tau}, \mathbf{x}_{t-\tau}, \mathbf{x}_t)$ with a set of possible states Ω, the conditional probability of encountering a particular state $\mathbf{x}_{t+\tau} \in \Omega$ is only conditional on $\mathbf{x}_t \in S$; i.e. $\mathbb{P}(\mathbf{x}_{t+\tau} | \mathbf{x}_t, \mathbf{x}_{t-\tau}, \mathbf{x}_{t-2\tau}, ...) = \mathbb{P}(\mathbf{x}_{t+\tau} | \mathbf{x}_t)$. In contrast to the methods presented in Section 3.3, we assume that we have a finite number of discrete states. Therefore we consider

$$S \coloneqq \{1, \dots, n\} \cong \Omega \tag{3.24}$$

as state space for the remainder of this section. Often we are presented with data that does not live in a countable or even finite state space. In these cases, the state space is tessellated using a finite amount of indicator functions. Typically, the tesselation is a Voronoi decomposition.

As shown at the example of the Prinz potential (Figure 3.7a), the fineness of the chosen discretization affects MSM approximation quality [133]. The estimated transition matrix can approximate the dynamics with higher spatial resolution in a finer discretized space (Figure 3.7b). Furthermore, the discretization has implications on the estimated eigenfunctions and, in particular, the estimated stationary distribution (Figure 3.7c). Evaluating the eigenvalues for a given discretization yields a comprehensive picture of the model's quality (Figure 3.7d) as the true eigenvalues present an upper bound to the estimated ones (variational principle [90]): the sum of eigenvalues reflects the VAMP-1 score.

MSMs fit into the framework of transfer operators as introduced in Section 3.3 (see Figure 3.1). In particular, an indicator function ansatz used with VAC and/or EDMD yields an MSM. When indicator functions are used with VAMP, one obtains generalized MSMs (GMSMs), which are capable of representing time-inhomogeneous dynamics. We suggest Refs. 133–135 for thorough reviews.

The conditional probabilities in the MSM framework are described by a transition matrix $P \in \mathbb{R}^{n\times n}$, where $n = |S|$ is the number of states. The transition matrix is given by

$$P_{ij} = \mathbb{P}(x_{t+\tau} = j | x_t = i) \qquad \forall t \geq 0, \tag{3.25}$$

i.e., the time-stationary probability of transitioning from state $i \in S$ to state $j \in S$ within time τ. This also means that P is a row-stochastic matrix. Note that the MSM transition matrix is a special case of a transfer operator approximation (see Section 3.3), where the ansatz consists of indicator functions.

Dynamical quantities of interest can be computed from an MSM's transition matrix, e.g., mean first passage times and fluxes among (sets of) states [199], implied timescales [133], or metastable decompositions of Markov states [200].

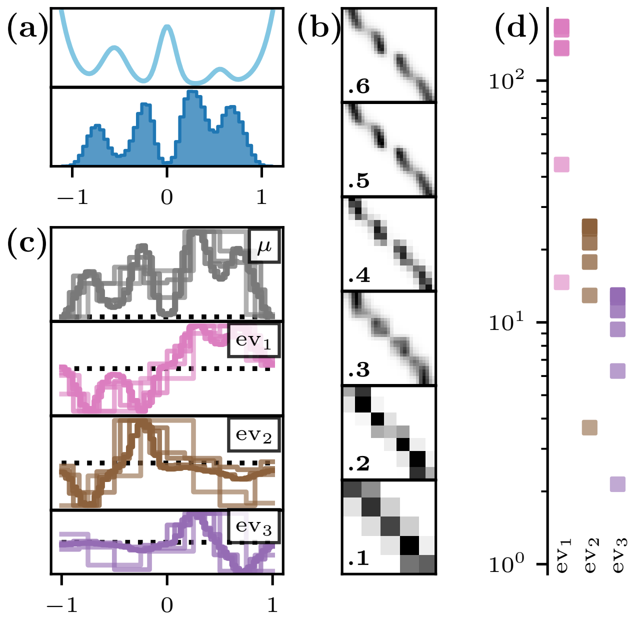


Figure 3.7: **Markov state models (MSMs) on Prinz potential.** MSM spectral decomposition for a random walk in a asymmetric 1D 4-well potential; the corresponding potential function (upper part) with histogram of simulated data (lower part) is depicted in **(a)**. **(b)** shows transition matrix estimates for various discretizations, from very coarse **(b)**.1 to very fine **(b)**.6. The discrete states are sorted in ascending order with respect to their corresponding x-coordinate. **(c)** depicts the four dominant left eigenvectors; discretizations are color coded from faint (coarse) to strong colors (fine discretization). The eigenvalues corresponding to the nonstationary eigenvectors are depicted in **(d)** in the same color.

3.4.0.0.1 MSM estimation with deeptime The goal of the `deeptime.markov` module is to provide tools to estimate and analyze MSMs from discrete-state time series data. If the data's domain is not discrete, classical discretization algorithms (such as the ones implemented in `deeptime.clustering`) can be employed to assign each frame to a state.

In what follows, we introduce the core object, the `MarkovStateModel`, as well as a variety of estimators. An overview of the main models contained in the `markov` module is depicted in Figure 3.8.

`deeptime` implements maximum-likelihood estimators for Markov state models as well as Bayesian sampling routines [201], leading to `MarkovStateModel` and `BayesianPosterior` model instances, respectively. An integral component of MSM estimation and sampling based on time series data is collecting statistics over the encountered state transitions (transition counting), which leads to a `TransitionCountModel`.

Bayesian sampling of MSMs leads to a `BayesianPosterior` that consists out of one `Ma⌋ rkovStateModel` instance representing the prior as well as the sampled `MarkovStateModels` instances (see Figure 3.8a). Each `MarkovStateModel` possesses a transition matrix (3.25) and, if available, statistical information about the data in the form of a transition count matrix. Furthermore, `deeptime` provides augmented Markov models (AMMs) [202] which can be used when experimental data is available, as well as observable operator model MSMs (OOMs) [138]. OOMs are unbiased estimators for the MSM transition matrix that correct for the effect of being presented with out of equilibrium data even when short lag-times are used. Both AMMs and OOMs inherit from the `MarkovStateModel` class definition.

While MSMs are a special case of the transfer operator model (cf. Section 3.3.1 and Figures 3.1 and 3.2), they can be converted to `CovarianceKoopmanModel`s of two different types. In one case, one can define the Koopman operator solely based on the transition matrix and corresponding stationary distribution; i.e., without respect to any statistical information. In the other case, when statistical information is present in the form of a `TransitionCountModel`, the statistics over transition counts may be used to estimate an empirical distribution according to which the Koopman operator is defined. The choice of Koopman model is up to the user; therefore, in `deeptime` MSMs do not inherit from `CovarianceKoopmanModel` but rather offer properties yielding respective instances of `CovarianceKoopmanModel`.

When estimating MSMs from data, `deeptime` assumes that the data is in the form of $k \geq 1$ trajectories $T_1, T_2, \ldots, T_k$ which comprise sequences of discrete states, i.e.,

$$T_i = (s_1, s_2, \ldots, s_{n_i}),\ \forall j = 1, \ldots, n_i : s_j \in S, \tag{3.26}$$

where n_i is the length of the i-th trajectory and $S = \{0, 1, \ldots, N_S - 1\}$ is the set of discrete states. In terms of further analysis it can be desirable to restrict the discrete state space onto a subset of $S' \subset S$, e.g., when certain state transitions are not populated and/or to select an ergodic subset. This task is best performed using a `TransitionCountModel` instance prior to estimating an MSM, as it possesses methods to produce new instances of the transition count model but restricted onto S'.

3.4.0.0.2 Hidden Markov models In many applications, the observed processes are only approximately Markovian in discrete state space; i.e., MSMs are only approximately valid [133]. The Markovianity assumption for the observed dynamics is discarded for hidden Markov models (HMMs) which assume that the modeled stochastic process is hidden (not directly observable). Therefore, the central object of the HMM is the transition matrix $\tilde{P}$ among hidden states $s_i \in S$. The transition matrix $\tilde{P}$ can be estimated from the time series of observable states O with the Baum–Welch algorithm [136, 203–205]. Briefly: alongside the transition matrix $\tilde{P}$, for each hidden state $s_t \in S$ the algorithm estimates an emission probability for a given observable state $\mathbf{o}_t \in O$.

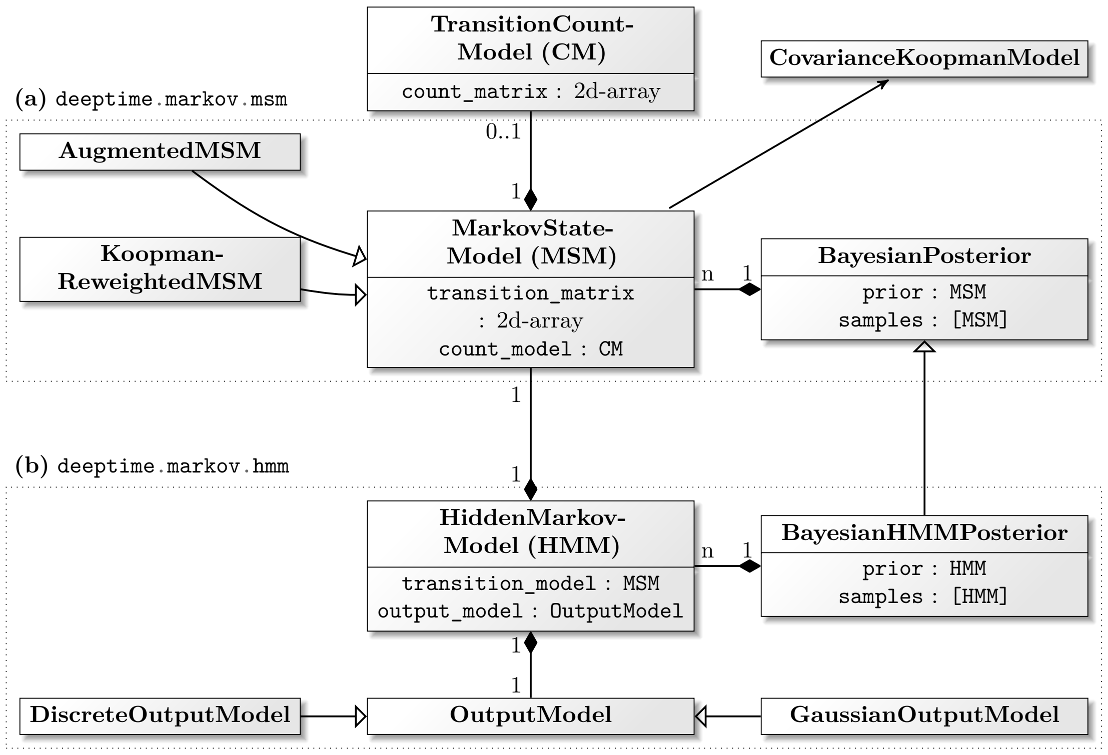


Figure 3.8: **Class diagram showing relationships of main models contained in the `markov` package.** We show two of the sub-packages of the `markov` module. Classes may be related via inheritance (—▷), composition (◆), or produce objects of another kind (→). In case of composition, we further denote the cardinality of the relationship next to the arrow. **(a)** `msm`. The `markov.msm` package has the Markov state model (MSM) at its core. It is completely determined by a transition matrix, but may also contain information about the statistics of data, in which case it possesses a `TransitionCountModel`. Furthermore there are `AugmentedMSM` and `KoopmanReweightedMSM` subclasses. An MSM is also a Koopman model using indicator basis functions; therefore, it can generate corresponding `CovarianceKoopmanModel` objects (see Section 3.3 and Figure 3.2). Bayesian sampling around an MSM leads to `BayesianPosteriors`, consisting of the prior and drawn samples. **(b)** `hmm`. This package contains estimators and models corresponding to hidden Markov model estimation. A `HiddenMarkovModel` consists out of a `transition_model` which describes evolution of a hidden state and an `OutputModel` which assigns a distribution over observable states to each hidden state. The discrete output model assigns each hidden state a discrete probability distribution over states, the Gaussian output model samples from one-dimensional Gaussians with means and variances conditioned on the hidden state. Same as MSMs, Bayesian sampling is available for hidden Markov models (HMMs), leading to a `BayesianHMMPosterior`.

HMMs therefore provide a (time-dependent) mapping between observable and hidden states along with the transition matrix $\tilde{P}$ [137]. This further allows us to estimate a maximum likelihood pathway of the trajectories in the hidden state space (Viterbi algorithm [206]).

Because the Baum–Welch algorithm converges to a local likelihood maximum [137], it is crucial to provide a reasonable initial guess of the emission probabilities and initial state distribution. `deeptime` offers multiple possibilities to initialize the HMM estimation procedure (contained in the `deeptime.markov.hmm.init` package), with a fallback option to a classical MSM or MSM-derived (e.g., PCCA [200]) estimate of the metastable dynamics.

The initial guess is an object of type `HiddenMarkovModel` (see Figure 3.8b). HMMs are composed of an MSM which describes the hidden state transitions and an output model. The output model is responsible for mapping a hidden state s_t to an observable state $\mathbf{o}_t = \mathbf{o}(s_t) \in O$. `deeptime` offers `DiscreteOutputModel`s which map each hidden state to a sample of a discrete probability distribution over observable states as well as `GaussianOutputModel`s which map a hidden state to a sample of a one-dimensional Gaussian distribution with mean and variance depending on the hidden state.

As with MSMs, HMMs in `deeptime` also support Bayesian sampling following a Gibbs sampling scheme detailed in Ref. 158. This produces a `BayesianHMMPosterior` which inherits from the `BayesianPosterior`, (cf. Figure. 3.8), which allows samples of quantities of interest which can be derived from an HMM instance to be collected.

3.5 Sparse identification of nonlinear dynamics

The sparse identification of nonlinear dynamics (SINDy) algorithm [54] is a data-driven method for discovering nonlinear dynamical systems models from measurement data using sparse regression. The method also fits into the Koopman operator framework presented in Sections 2.1.1–2.1.2 and 3.3 since it is related to an approximation of the Koopman generator, defined by

$$\mathcal{L}f := \lim_{\tau \to 0} \frac{1}{\tau}(\mathcal{K}_\tau f - f),$$

see Ref. 207 for details. The goal of SINDy is to approximate a nonlinear dynamical system

$$\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}) \tag{3.27}$$

as a sparse linear combination of candidate functions $\theta_k(\mathbf{x})$:

$$\frac{d}{dt}x_j \approx \sum_{j=1}^{\ell} \xi_{jk}\theta_k(\mathbf{x}) \quad \Longrightarrow \quad \frac{d}{dt}\mathbf{x} \approx \Xi\mathbf{\Theta}(\mathbf{x}). \tag{3.28}$$

The matrix Ξ is assumed to be sparse, with the nonzero elements determining which terms in the library $\mathbf{\Theta}$ are active in the dynamics. In practice, the library $\mathbf{\Theta}$ is defined either to contain a generic set of terms, such as monomials, or terms guided by partial knowledge of the physical system. For example, metabolic regulatory networks often include rational function nonlinearities [208]. However, monomomials often suffice, either because the governing physics is polynomial (e.g., the Navier–Stokes equations for fluid dynamics), or because polynomials provide a reasonable Taylor expansion of the dynamics into a normal form [209].

The sparse matrix Ξ is typically identified via sparse regression based on a time series of data $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ collected at times $t_1, t_2, \ldots, t_m$. This data is organized into a matrix $X \in \mathbb{R}^{n\times m}$, and a matrix of derivatives $\dot{X}$ is formed either by measuring the derivatives directly or numerically approximating them from the data in X. The library $\mathbf{\Theta}$ may now be evaluated on the data matrix X, resulting in the following matrix system of equations

$$\dot{X} \approx \Xi\mathbf{\Theta}(X). \tag{3.29}$$

The matrix Ξ is then solved for in the following optimization

$$\operatorname{argmin}_{\Xi} \|\dot{X} - \Xi\mathbf{\Theta}(X)\|_F + \lambda\|\Xi\|_0. \tag{3.30}$$

The first term measures the model error, while the $\|\cdot\|_0$ term counts the number of nonzero elements in Ξ, promoting sparsity. This zero norm is non-convex, and several relaxations are available that yield sparse solutions [54, 210].

There are several extensions to SINDy, e.g., incorporating the effect of actuation and control [211, 212] and to enforce partially known physics, such as symmetries and conservation laws [213]. It is also possible to combine SINDy with deep autoencoders to identify a coordinate system in which the dynamics are approximately sparse [209]. Other extensions include the discovery partial differential equations [214, 215], the modeling of stochastic dynamics [207, 216, 217], updating already existing models [218], and weak formulations of the problem [219–221], among others [215, 222–224]. SINDy has also been extended to accommodate tensor libraries, which dramatically increases its ability to handle systems with high state dimension [225]. This sparse modeling procedure has been applied to discover new physical models, for example in fluid dynamics [213, 226], including for turbulence closure modeling [227].

It is important to note that SINDy also applies equally well to discrete time systems

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k) \tag{3.31}$$

in which case derivatives need not be estimated. If SINDy is formulated in discrete time with no sparsity promoting term (i.e., $\lambda = 0$) and with a library $\mathbf{\Theta}(\mathbf{x}) = \mathbf{x}$, then the DMD approximation is recovered.

To demonstrate SINDy, we consider the Rössler attractor [228], a system of ODEs exhibiting chaotic behavior. Fig. 3.9 shows the reconstruction of the dynamic attractor for the Rössler system of equations:

$$\begin{aligned}\dot{x}_1 &= -x_2 - x_3\\ \dot{x}_2 &= x_1 + ax_2\\ \dot{x}_3 &= b + x_3(x_1 - c)\end{aligned}$$

with constants $a = 0.1$, $b = 0.1$, and $c = 14$.

`deeptime` has two SINDy objects. The `SINDy` estimator is used to solve the optimization problem (3.29) given $\mathbf{\Theta}$, X, and optionally $\dot{X}$. By default the sequentially-thresholded least-squares algorithm [54] is used to solve the optimization problem. If $\dot{X}$ is not user-provided, it is estimated from X with a first order finite difference method.

The estimator produces a `SINDyModel`, representing the learned dynamical system. The model can be used to predict derivatives given state variables, to simulate forward in time from novel initial conditions, and to score itself against ground truth data.

The implementation is API-compatible to the Python package PySINDy [149], which in particular enables users to make use of a wider range of optimizers defined in PySINDy.

3.6 Datasets

`deeptime` offers a range of datasets to which its methods can be applied. The datasets and methods were purposefully designed to be non-domain-specific and to deliver data generators

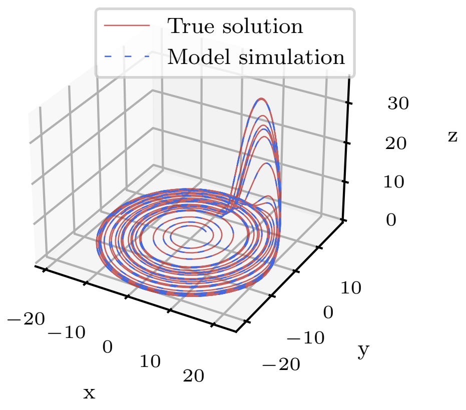


Figure 3.9: Reconstruction of the Rössler attractor using the sparse identification of nonlinear dynamics (SINDy) method.

rather than fixed datasets. As a result, the repository as well as package size are remain small and generation parameters can be varied to study their effects on the algorithms. The data simulators are structured so that performance-critical parts are implemented in C++ and the generation procedure is not very time consuming.

In particular, a range of example SDEs of the form

$$\mathrm{d}\mathbf{x}_t = \mathbf{F}(t, \mathbf{x}_t)\mathrm{d}t + \sigma \mathrm{d}W_t,$$

where $\mathbf{F} : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$, W_t a d-dimensional Wiener process, and $\sigma \in \mathbb{R}^{d\times d}$, are implemented. All these SDEs are integrated using an Euler–Maruyama integrator. While the definition of these examples happens in C++, it is set up in such a way that also C++-inexperienced users can natively define their own. In Listing C we show an example of such a definition.

For example, the definition of a double well system

$$\mathrm{d}\mathbf{x}_t = -\nabla V(\mathbf{x}_t)\mathrm{d}t + \sigma \mathrm{d}W_t, \quad V(\mathbf{x}) = (\mathbf{x}_1^2 - 1)^2 + \mathbf{x}_2^2,$$

with $\mathbf{x}_t \in \mathbb{R}^2$ and $\sigma = \mathrm{diag}(0.7, 0.7)$ can be achieved by a struct definition detailing the evaluation of the right-hand side. Many of the parameters of the system can be made available at compile-time, enabling further optimizations by the compiler. An example trajectory as well as a contour plot of the potential landscape can be found in Figure 3.10a. By making information such as the data type (e.g., `float` or `double`), the dimension of the state space, the integrator, and σ available at compile time, the compiler can perform further optimizations and potentially vectorizations that it otherwise could not, reducing the time it needs for evaluation.

Users also have the option to define the right-hand side $\mathbf{F}(\mathbf{x}_t)$ as well as the diffusion matrix σ in Python at some performance penalty (see Figure 3.10b. Three different implementations are compared: one native C++ implementation, one implementation where just σ and the right-hand side are defined in Python, and one native Python implementation. One can see that roughly one order of magnitude in terms of evaluation performance is gained from native Python to a mixed Python/C++ implementation and from the mixed implementation to a native C++ implementation.

A drawback of making this information known at compile time is that for the mixed Python/C++ implementations, the dimension needs to be predefined; i.e., it must be explicitly

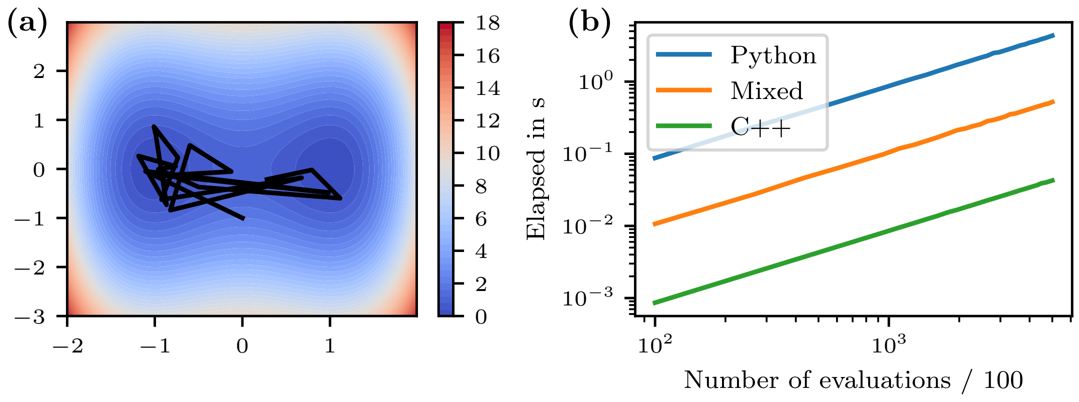


Figure 3.10: **Two-dimensional double-well example system.** We show a performance comparison between three different implementations of a two-dimensional double well system. **(a)** Potential landscape $V(\mathbf{x}) = V(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1^2 - 1)^2 + \mathbf{x}_2^2$ with example trajectory under diffusion matrix $\sigma = \operatorname{diag}(0.7, 0.7)$ and state snapshots taken every 10^4 steps. **(b)** Time elapsed over number of evaluations, while one evaluation corresponds to evolving the state by 100 steps under a integration step size of $h = 10^{-3}$. "Python" refers to a native Python implementation, "C++" refers to a native C++ implementation, and "mixed" refers to a C++ implementation, where the diffusion matrix as well as the gradient of the potential are defined in Python.

exported when generating the Python bindings. On the other hand it improves performance and one can first prototype a system using the Python-defined diffusion matrix and right-hand side, and then eventually move the implementation to native C++ with relative ease.

3.7 Discussion and outlook

We have outlined the key components of `deeptime`'s API and discussed the corresponding theory and methods as well as their relationships, in particular transfer operator based methods which can be used for dimension reduction, coherent set detection, analysis of kinetic quantities, and discovery of governing dynamics. These applications were each demonstrated with respective examples.

For future development we are actively looking for contributors and want to extend the currently available library of methods and datasets. For example there is a version of VAMPNets which allows the inclusion of experimental data. The SINDy module can be extended to include neural network based estimation of dynamics. Also the HMM module can be extended to support a richer set of output models. Furthermore the inclusion of more example datasets is desirable as this enables users to test and analyze existing or new methods and draw comparisons.

Finally we are planning to integrate time series specific chunking and streaming capabilities so that methods which support online learning can more easily be used with data streams.

Chapter 4

ReaDDy2: Fast and flexible software framework for interacting-particle reaction dynamics

The results of this chapter were originally presented in Ref. P1:

> M. Hoffmann, C. Fröhner, F. Noé. "ReaDDy 2: Fast and flexible software framework for interacting-particle reaction dynamics". In: *PLoS Comput. Biol.* (2019). URL: `https://doi.org/10.1371/journal.pcbi.1006830`.

Moritz Hoffmann (MH) was lead author and sole first author in this project. The work was developed in collaboration with Christoph Fröhner (CF) and is therefore also part of the dissertation of CF. The author contributions were as follows: MH, CF, and Frank Noé (FN) conceived the project. MH implemented the majority of `ReaDDy2`. CF implemented parts of `ReaDDy2`. MH laid out and implemented the concept of topology reactions as detailed in Section 4.2.3. MH carried out the performance analysis of Section 4.3.2. CF laid out the validation scenarios presented in Sections 4.4.1–4.4.3 and 4.4.5. MH and CF laid out the scenarios presented in Section 4.4.4. MH and CF analyzed and visualized the data. All contributors wrote the paper.

Interacting-particle reaction dynamics (iPRD) combine the simulation of dynamical trajectories of interacting particles as in MD simulations with reaction kinetics, in which particles appear, disappear, or change their type and interactions based on a set of reaction rules. We introduced iPRD as a combination of overdamped Langevin dynamics and reactive particles in Section 2.4. This combination facilitates the simulation of reaction kinetics in crowded environments, involving complex molecular geometries such as polymers, and employing complex reaction mechanisms such as breaking and fusion of polymers. iPRD simulations are ideal to simulate the detailed spatiotemporal reaction mechanism in complex and dense environments, such as in signaling processes at cellular membranes, or in nano- to microscale chemical reactors.

The complexity of the simulated systems often exceeds the complexity of systems typically analyzed with the tools introduced in Chapter 3. Also computational costs play a big role, as systems can contain many thousand particles and potentially very long trajectories are required to acquire sufficient sampling. Here, we introduce the iPRD implementation `ReaDDy2`, which aims to be both computationally efficient but also user-friendly. It is validated on several model systems with comparisons against—if available—analytically obtained quantities and results from the literature.

4.1 Introduction

Other available simulation tools that are capable of special cases of iPRD simulations are, e.g., the MD packages LAMMPS [41] which is capable of forming and breaking bonds dynamically and ESPResSo [229, 230] which additionally has an implementation of catalytic reactions. In comparison to the iPRD simulator `ReaDDy` [68], these do not support full iPRD and are built and optimized for particle numbers that stay roughly constant. Comparing iPRD and classical PBRD without interactions, the interaction potentials can be used to induce structure on mesoscopic length scales, e.g., volume-exclusion in crowded systems [68, 231], clustering of weakly interacting macromolecules [232], restriction of diffusing particles to arbitrarily-shaped membranes [6, 22, 68]. Furthermore it allows to study the large-scale structure of oligomers [233], polymers and membranes [234]. When not only considering interactions but also reactions, a wide range of reactive biochemical systems are in the scope of the model. For example, the reaction dynamics of photoreceptor proteins in crowded membranes [22] including cooperative effects of transmembrane protein oligomers [6] have been investigated. Another example is endocytosis, in which different proteins interact in very specific geometries [235, 236]. The simulation tool Cytosim [237] is another software package that can be used to investigate mesoscopic biochemical systems, specifically geared towards the simulation of the cytoskeleton.

The price of resolving these details is that the computation is dominated by computing particle-particle interaction forces and—depending on the system's density and reactivity—evaluating reactions. Although non-interacting particles can be propagated quickly by exploiting solutions of the diffusion equation [57, 69, 238, 239], interacting particles are propagated with small time-steps [240, 241], restricting the accessible simulation timescales whenever parts of the system are dense. As this computational expense is not entirely avoidable when the particle interactions present in iPRD are needed to model the process of interest realistically, it is important to have a simulation package that can fully exploit the computational resources.

`ReaDDy2` provides a Python interface in which the simulation environment, particle interactions and reaction rules can be conveniently defined and the simulation can be run, stored and analyzed. A C++ interface is available to enable deeper and more flexible interactions with the framework. The main computational work of `ReaDDy2` is done in hardware-specific simulation kernels. While the version introduced here provides single- and multi-threading CPU kernels, the architecture is ready to implement GPU and multi-node kernels. We demonstrate the efficiency and validity of `ReaDDy2` using several benchmark examples. `ReaDDy2` is available at the https://readdy.github.io/ website.

The library is significantly faster, more flexible, and more conveniently usable than its predecessor `ReaDDy` [68, 242]. Specifically, `ReaDDy2` includes the following new features:

- **Computational efficiency and flexibility**: `ReaDDy2` defines computing kernels which perform the computationally most costly operations and are optimized for a given computing environment. The current version provides a single-CPU kernel that is four to ten times (depending on system size) faster than ReaDDy, and a multi-CPU kernel that scales with 80% efficiency to number of physical CPU cores for large particle systems (Section 4.3.2). Kernels for GPUs or parallel multi-node kernels can be readily implemented with relatively little additional programming work (Section 4.3).

- **Python user interface**: ReaDDy2 can be installed via the conda package manager and used as a regular python package. The python interface provides the user with functionality to compose the simulation system, define particle interactions, reactions and parameters, as well as run, store and analyze simulations.
- **C++ user interface**: ReaDDy2 is mainly implemented in C++. Developers interested in extending the functionality of ReaDDy2 in a way that interferes with the compute kernels, e.g., by adding new particle dynamics or reaction schemes, can do that via the C++ user interface.
- **Reversible reaction dynamics**: ReaDDy2 can treat reversible iPRD reactions by using steps that obey detailed balance, as described in [243] (iPRD–DB), and thus ensure correct thermodynamic behavior for such reactions (Section 4.4.1).
- **Topologies**: We enable building complex multi-particle structures, such as polymers, by defining topology graphs (briefly: topologies, see Section 4.2.3). As in MD simulations, topologies are an efficient way to encode which bonded interactions (bond, angle and torsion terms) should act between groups of particles in the same topology. Note that particles in topologies can still be reactive. For example, it is possible to define reactions that involve breaking or fusing polymers (Section 4.4.4).
- **Potentials and boundaries**: Furthermore, the range of by default supported interaction potentials has been broadened, now including harmonic repulsion, a harmonic interaction potential with a potential well, Lennard–Jones interaction, and screened electrostatics. The simulation volume can also be equipped with partially or fully periodic boundary conditions.

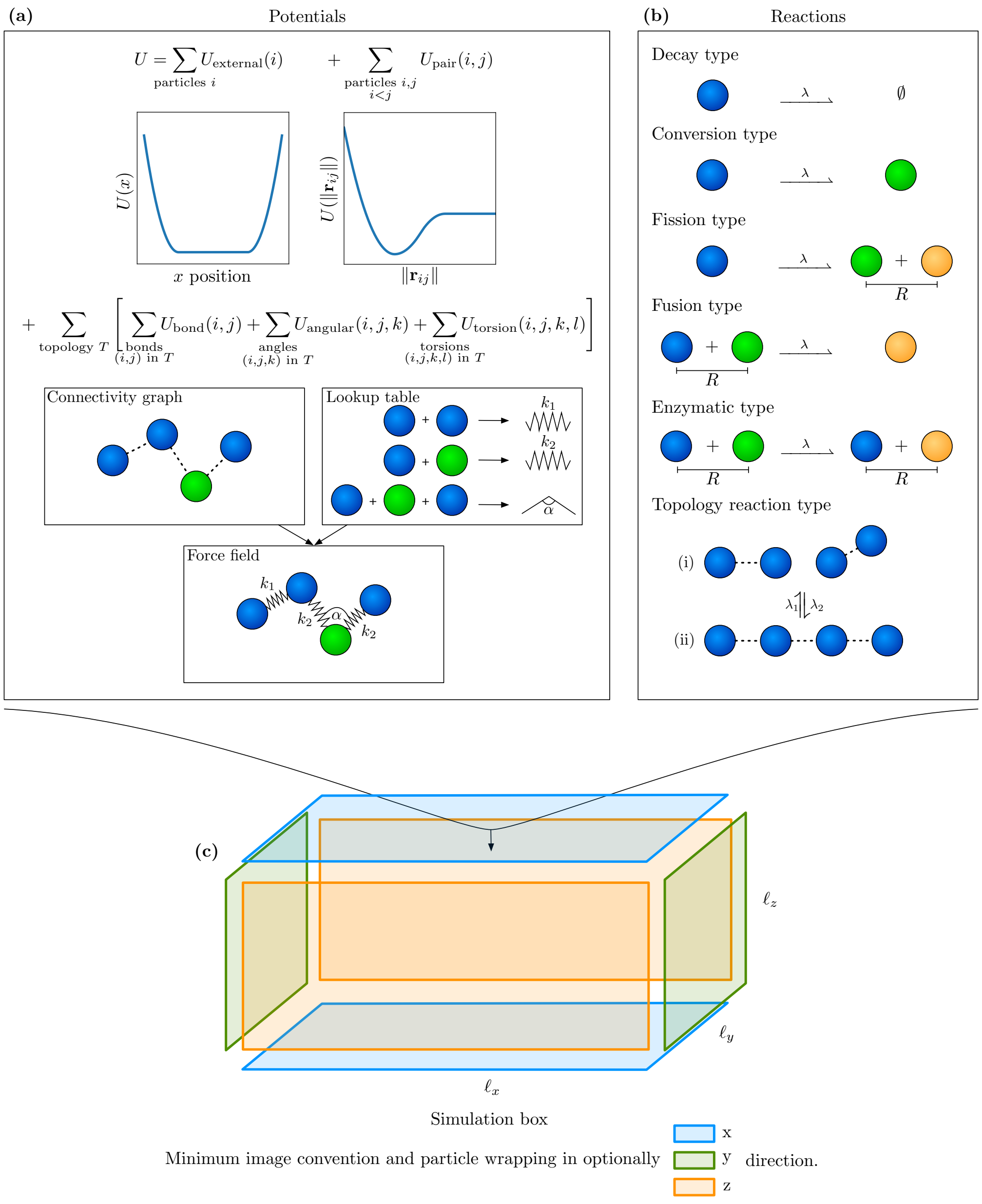
This chapter summarizes the features of ReaDDy2 and the demonstrates its efficiency and validity of ReaDDy2 using several benchmarks and reactive particle systems. With few exceptions, we limit our description to the general features that are not likely to become outdated in future versions.

4.2 Interacting-particle reaction dynamics in ReaDDy2

The ReaDDy2 simulation system consists of particles interacting by potentials and reactions (Figure 4.1) at a temperature T. Such a simulation system is confined to a box with either repulsive or periodic boundaries. A boundary always has to be either periodic or be equipped with repulsive walls so that particles cannot diffuse away arbitrarily. To simulate iPRD in complex architectures, such as cellular membrane environments with specific shapes, additional potentials can be defined that confine the particle to a sub-volume of the simulation box (see Section 4.2.1).

ReaDDy2 provides a developer interface to flexibly design models of how particle dynamics are propagated in time. The default model, however, is overdamped Langevin dynamics (cf. Section 2.2) with isotropic diffusion as this is the most commonly used PBRD and iPRD model. In these dynamics a particle i moves according to the SDE given in Equation (2.18).

In ReaDDy2 the default assumption is that the diffusion coefficients $D^{(i)}(T)$ are given for the simulation temperature T. Additionally, we offer the option to define diffusion coefficients for a reference temperature $T_0 = 293K$ and then generate the diffusion coefficients at the simulation



(a)
Potentials
$U = \sum_{\text{particles } i} U_{\text{external}}(i) + \sum_{\substack{\text{particles } i,j \\ i<j}} U_{\text{pair}}(i,j)$
$U(x)$
x position
$U(\|\mathbf{r}_{ij}\|)$
$\|\mathbf{r}_{ij}\|$
$+ \sum_{\text{topology } T} \left[\sum_{\substack{\text{bonds} \\ (i,j) \text{ in } T}} U_{\text{bond}}(i,j) + \sum_{\substack{\text{angles} \\ (i,j,k) \text{ in } T}} U_{\text{angular}}(i,j,k) + \sum_{\substack{\text{torsions} \\ (i,j,k,l) \text{ in } T}} U_{\text{torsion}}(i,j,k,l) \right]$
Connectivity graph
Lookup table
k_1
k_2
α
Force field
(b)
Reactions
Decay type
λ
$\emptyset$
Conversion type
Fission type
R
Fusion type
Enzymatic type
Topology reaction type
(i)
(ii)
λ_1
λ_2
(c)
ℓ_z
ℓ_y
ℓ_x
Simulation box
Minimum image convention and particle wrapping in optionally
x
y
z
direction.

Figure 4.1: **The simulation model.** **(a)** Potentials: Particles are subject to position-dependent external potentials, such as boundary potentials or external fields and interaction potentials involving two, three or four particles. As in molecular dynamics force fields, bonded potentials are defined within particle groups called "topologies" whose bonding structure is defined by a connectivity graph. **(b)** Reactions: Most reactions are unimolecular or bimolecular particle reactions. Topology reactions act on the connectivity graphs and particle types and therefore change the particle bonding structure. **(c)** Simulation box: The simulation box with edge lengths ℓ_x, ℓ_y, and ℓ_z. It can optionally be periodic in a combination of x, y, and z directions, applying the minimum image convention.

temperature T by employing the Einstein–Smoluchowski model for particle diffusion in liquids [244, 245]:

$$D^{(i)}(T) = D^{(i)}(T_0)\frac{T}{T_0}.$$

This way, simulations at different temperatures are convenient while only having to specify one diffusion constant. Using this model, the dynamics are given by

$$\mathrm{d}\mathbf{x}_t^{(i)} = -\frac{D^{(i)}(T_0)}{k_B T_0}\mathbf{f}_t^{(i)}\mathrm{d}t + \sqrt{2D^{(i)}(T_0)\frac{T}{T_0}}\mathrm{d}\mathbf{W}_t^{(i)}. \tag{4.1}$$

This means that the mobility is preserved if the temperature changes and (2.18) is recovered for $T = T_0$.

A simple integration scheme for (2.18) is Euler–Maruyama, as detailed in Equation (2.19). Using the modified dynamics (4.1), the integration scheme reads

$$\mathbf{x}_{t+\tau}^{(i)} = \mathbf{x}_t^{(i)} - \tau\frac{D^{(i)}(T_0)}{k_B T_0}\mathbf{f}_t^{(i)} + \sqrt{2D^{(i)}(T_0)\frac{T}{T_0}\tau}\boldsymbol{\eta}_t, \tag{4.2}$$

where—same as in (2.19)—$\tau > 0$ is a finite time step size and $\boldsymbol{\eta}_t \sim \bigotimes_{i=1}^d \mathcal{N}(0,1)$.

The particles' positions are loosely bound to a cuboid simulation box with edge lengths ℓ_x, ℓ_y, ℓ_z (cf. Figure 4.1). If a boundary is non-periodic it is equipped with a repulsive wall given by the potential

$$V_\text{wall} : \mathbb{R}^3 \to \mathbb{R},\ \mathbf{x} \mapsto \sum_{i=1}^3 \frac{1}{2}k d(\mathbf{x}_i, W_i)^2 \tag{4.3}$$

acting on every component i of the single particle position $\mathbf{x}$, where k is the force constant,

$$W = \prod_{i=1}^3 W_i = \prod_{i=1}^3 \left[x_\text{origin}^{(i)}, x_\text{origin}^{(i)} + x_\text{extent}^{(i)}\right]$$

the cuboid in which there is no repulsion contribution of the potential, and

$$d(\cdot, W_i) \coloneqq \inf\left\{d(\cdot, w) : w \in W_i\right\}$$

the shortest distance to the interval W_i. The cuboid can be larger than the simulation box in the periodic directions. In non-periodic directions there must be at least one repulsive wall for which this is not the case.

Due to the soft nature of the walls particles still can leave the simulation box in non-periodic directions. In that case they are no longer subject to pairwise interactions and bimolecular reactions however still are subject to the force of the wall pulling them back into the box.

Other types of dynamical models and other integration schemes can be implemented in `ReaDDy2` via its C++ interface. For example, non-overdamped dynamics, anisotropic diffusion [241, 246], hydrodynamic interactions [247], or employing the MD-GFRD scheme to make large steps for noninteracting particles will all affect the dynamical model and can be realized by writing suitable plugins.

4.2.1 Potentials

The deterministic forces are given by the gradient of a many-body potential energy U (see Figure 4.1a)

$$\mathbf{f}_t^{(i)} = \boldsymbol{\nabla}_i \left(\sum_i U_{\mathrm{ext}}(\mathbf{x}_i) + \sum_{i \neq j} U_{\mathrm{pair}}(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i \neq j \neq k} U_{\mathrm{triple}}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) + \dots \right).$$

The potentials are defined by the user. `ReaDDy2` provides a selection of standard potential terms, additional custom potentials can be defined via the C++ interface and then included into a Python simulation script.

External potentials only depend on the absolute position of each particle. They can be used, e.g., to form softly repulsive walls (4.3) and spheres, or to attach particles to a surface, for example to model membrane proteins. Furthermore the standard potential terms enable the user to simulate particles inside spheres and exclude particles from a spherical volume. The mentioned potential terms can also be combined to achieve more complex geometrical structures. Under the hood, `ReaDDy2` makes it easy to define harmonic inclusion and/or exclusion potentials: if a function is provided which gives the shortest difference vector between a position and the surface of the geometry, one can automatically use it as harmonic potential in a simulation as custom geometry.

Pair potentials generally depend on the particle distance and can be used, e.g., to model space exclusion at short distances.

A fundamental restriction of `ReaDDy2` interaction potentials is that they have a finite range and can therefore be cut off. This means that, e.g., full electrostatics is not supported but screened electrostatic interactions are implemented (see Section 4.4.5). Additionally a harmonic repulsion potential, a weak interaction potential made out of three harmonic terms, and Lennard–Jones interaction are incorporated.

`ReaDDy2` has a special way of treating interaction potentials between bonded particles. Topologies define graphs of particles that are bonded and imply which particle pairs interact via bond constraints, which triples interact via angle constraints, and which quadruplets interact via a torsions potential. See Section 4.2.3 for details.

4.2.2 Reactions

Reactions are discrete events, that can change particle types, add, and remove particles (see Figure 4.1b). In `ReaDDy2`, the Doi reaction model (cf. Section 2.3.3) is used. That means, that each reaction is associated with a microscopic rate constant $\lambda > 0$ which has units of inverse time and represents the probability per unit time of the reaction occurring. The integration time-steps used in `ReaDDy2` should be significantly smaller than the inverse of the largest reaction rate. We compute reaction probabilities by Equation (2.30) under a integration time step τ.

In the software it is checked whether the time step τ is smaller than the inverse reaction rate up to a threshold factor of 10, otherwise a warning is displayed as discretization errors might become too large. In general, `ReaDDy2` reactions involve either one or two reactants. At any time step, a particle that is subject to an unary reaction will react with probability $p(\lambda; \tau)$. If there are two products, they are placed within a sphere of specified radius R_{u} around the educt's position $\mathbf{x}_0$. This is achieved by randomly selecting an orientation $\mathbf{n} \in \mathbb{R}^3$, distance $d \leq R_{\mathrm{u}}$, and weights $w_1 \geq 0$,$w_2 \geq 0$, s.t. $w_1 + w_2 = 1$. The products are placed at $\mathbf{x}_1 = \mathbf{x}_0 + dw_1\mathbf{n}$ and $\mathbf{x}_2 = \mathbf{x}_0 - dw_2\mathbf{n}$. Per default, $w_1 = w_2 = 0.5$ and the distances d are drawn such that the distribution is uniform with respect to the volume of the sphere. When it is necessary to produce

```
pairs = []
triples = []
quadruples = []

$G = G(V, E)$ # a graph with vertices $V$ and edges $E$
for $v_i \in V$:

  for $v_j \in N(v_i)$:  # all neighbors of $v_i$
    if i < j:  # prevent finding both $(v_i, v_j)$ and $(v_j, v_i)$
      pairs.append($(v_i, v_j)$)

      for $v_k \in N(v_i) \setminus \{v_j\}$:  # all neighbors of $v_i$ except $v_j$
        for $v_l \in N(v_j) \setminus \{v_i, v_k\}$:  # all neighbors of $v_j$ except $v_i$ and $v_k$
          quadruples.append($(v_k, v_i, v_j, v_l)$)

  for $v_k \in N(v_i) \setminus \{v_j\}$:
    if k < j:  # prevent finding both $(v_k, v_i, v_j)$ and $(v_j, v_i, v_k)$
      triples.append($(v_k, v_i, v_j)$)
```

Listing 1: Algorithm to find all edges (v_i, v_j), triples (v_k, v_i, v_j), and quadruples (v_k, v_i, v_j, v_l) in a connected graph G up to reversing the order, i.e., if the triple (v_j, v_i, v_k)—corresponding to an angle—is detected, the same triplet (v_k, v_i, v_j) is not recorded. Same for pairs corresponding to bonds and quadruples corresponding to torsion potentials.

new particles, we suggest to define a producing particle A and use the unary reaction $\mathrm{A} \rightharpoonup \mathrm{A} + \mathrm{B}$ with corresponding placement weights $w_1 = 0, w_2 = 1$ so that the A particle stays at its position.

A complex can be formed if there are two educts within a reactive distance of R_b or less, where R_b is a parameter, e.g., see Figure 4.1b Fusion or Enzymatic reaction. The formation probability is given in accordance to the Doi reaction model by $p(\lambda; \tau)$.

Optionally `ReaDDy2` can simulate reversible reactions using the reversible iPRD–DB scheme developed in [243]. This scheme employs a Metropolis–Hastings algorithm that ensures the reversible reaction steps to be made according to thermodynamic equilibrium by accounting for the system's energy in the educt and product states.

4.2.3 Topologies

Topologies are a way to group particles into superstructures. For example, large-scale molecules can be represented by a set of particles corresponding to molecular domains assembled into a topology. A topology also has a set of potential energy terms such as bond, angle, and torsion terms associated. The specific potential terms are implied by finding all paths of length two, three, and four in the topology connectivity graph. The used algorithm is sketched in Listing 1. The sequence of particle types associated to these paths then is used to gather the potential term specifics, e.g., force constant, equilibrium length or angle, from a lookup table (Figure 4.1a).

Reactions are not only possible between particles, but also between a topology and a particle (Figure 4.1b) or two topologies. In order to define such reactions, one can register topology types and then specify the consequences of the reaction on the topology's connectivity graph. We distinguish between global (or: structural) and local (or: spatial) topology reactions.

Global topology reactions are triggered analogously to unary reactions, i.e., they can occur at any time with a fixed rate and probability as given in (2.30). Any edge in the graph can be removed and added. Moreover, any particle type as well as the topology type can be changed, which may result in significant changes in the potential energy. If the reaction causes the graph to split into two or more components, these components are subsequently treated as separate topologies that inherit the educt's topology type and therefore also the topology reactions associated with it. Such a reaction is the topology analogue of a particle fission reaction.

A local topology reaction is triggered analogously to binary reactions with probability $p(\lambda;\tau)$ if the distance between two particles is smaller than the reaction radius. At least one of the two particles needs to be part of a topology with a specific type. The product of the reaction is then either yielded by the formation of an edge and/or a change of particle and topology types. In contrast to global reactions only certain changes to particle types and graphs can occur:

- Two topologies can fuse, i.e., an additional edge is introduced between the vertices corresponding to the two particles that triggered the reaction.
- A topology and a free particle can fuse by formation of an edge between the vertex of the topology's particle and a newly introduced vertex for the free particle.
- Two topologies can react in an enzymatic fashion, i.e., particle types of the triggering particles and topology types can be changed.
- Two topologies and a free particle can react in an enzymatic fashion analogously.

In all of these cases the involved triggering particles' types and topology types can be changed.

4.2.4 Simulation setup and boundary conditions

Once the potentials, the reactions (see Figure 4.1 (a,b)), and a temperature T have been defined, a corresponding simulation can be set up. A simulation box can be periodic or partially periodic, see Figure 4.1c. Periodicity in a certain direction means that with respect to that direction particle wrapping and the minimum image convention are applied. Non-periodic directions require a harmonically repelling wall as given in (4.3).

In order to define the initial condition, particles and particle complexes are added explicitly by specifying their 3D position and type. A simulation can now be started by providing a time step size τ and a number of integration steps.

4.3 Design and Implementation

`ReaDDy2` is mainly written in C++ and has Python bindings making usage, configuration, and extension easy while still being able to provide high performance. To encourage usage and extension of the software, it is Open Source and licensed under the BSD-3 license. It therefore can not only be used in other Open Source projects without them requiring to have a similar license, but also in a commercial context.

4.3.1 Design

The software consists of three parts. The user-visible toplevel part is the Python user interface, see Figure 4.2a. It is a language binding of the C++ user interface (Figure 4.2b) and has additional convenience functionality. The workflow consists out of three steps:

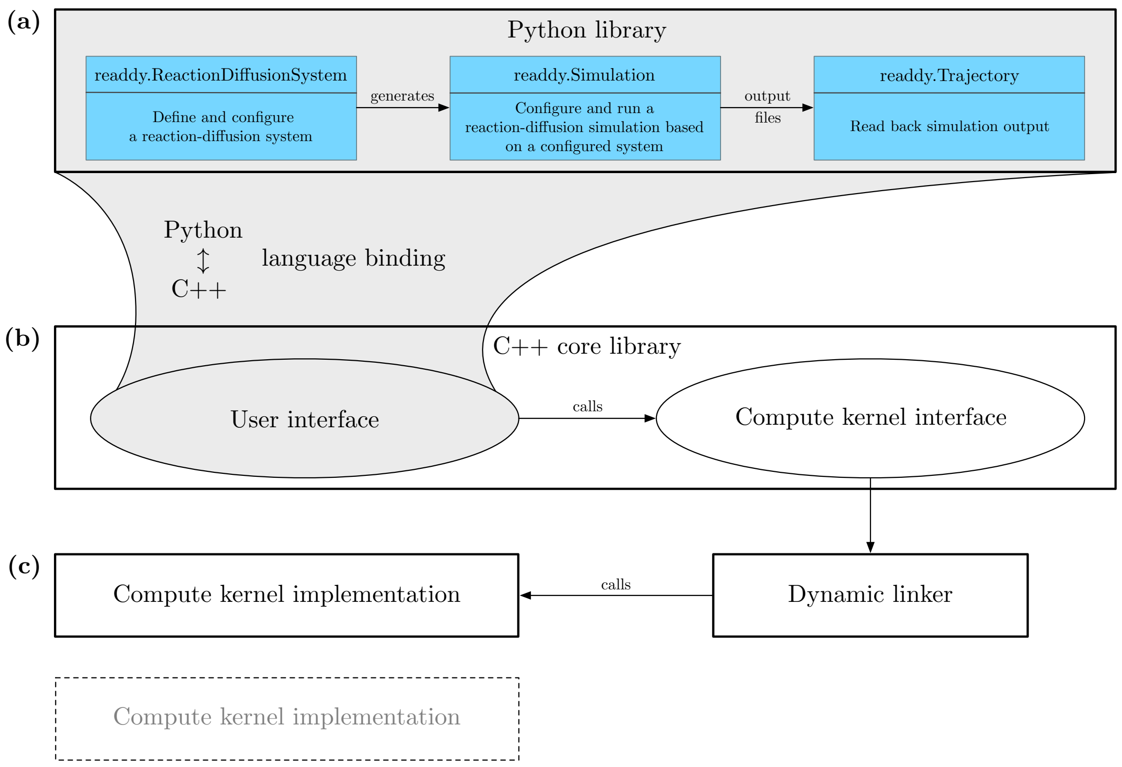

(a)
Python library
readdy.ReactionDiffusionSystem
Define and configure a reaction-diffusion system
generates
readdy.Simulation
Configure and run a reaction-diffusion simulation based on a configured system
output files
readdy.Trajectory
Read back simulation output
Python ↕ C++
language binding
(b)
C++ core library
User interface
calls
Compute kernel interface
(c)
Compute kernel implementation
calls
Dynamic linker
Compute kernel implementation

Figure 4.2: **The software structure. (a)** Python user interface: Provides a Python binding to the "C++ user interface" with some additional convenience functionality. The user creates a `readdy.ReactionDiffusionSystem` and defines particle species, reactions, and potentials. From a configured system, a `readdy.Simulation` object is generated, which can be used to run a simulation of the system given an initial placement of particles. **(b)** C++ core library: The core library serves as an adapter between the actual implementation of the algorithms in a compute kernel and the user interface. **(c)** Compute kernel implementation: Implements the compute kernel interface and contains the core simulation algorithms. Different compute kernel implementations support different hard- or software environments, such as serial and parallel CPU implementations. The compute kernel is chosen when the `readdy.Simulation` object is generated and then linked dynamically in order to provide optimal implementations for different computing environments under the same user interface.

```
initialize_compute_kernel()
if has_output_file:
    write_simulation_setup()

set_up_neighbor_list()
compute_forces()
evaluate_observables()
while continue_simulating():
    call_integrator()
    update_neighbor_list()
    perform_reactions()
    perform_topology_reactions()
    update_neighbor_list()
    calculate_forces()
    evaluate_observables()

tear_down_compute_kernel()
```

Listing 2: `ReaDDy2` default simulation loop. Each of the calls are dispatched to the compute kernel, see Figure 4.2. Furthermore, the user can decide to switch off certain calls in the simulation loop while configuring the simulation.

1. The user is creating a `readdy.ReactionDiffusionSystem`, including information about temperature, simulation box size, periodicity, particle species, reactions, topologies, and physical units. Per default the configuration parameters are interpreted in a unit set well suited for cytosolic environments (lengths in nm, time in ns, and energy in kJ/mol), e.g., particles representing proteins in solution. The initial condition, i.e., the positions of particles, is not yet specified.

2. The system can generate one or many instances of `readdy.Simulation`, in which particles and particle complexes can be added at certain positions. When instantiating the simulation object, a compute kernel needs to be selected, in order to specify how the simulation will be run (e.g., single-core or multi-core implementation). Additionally, observables to be monitored during the simulation are registered, e.g., particle positions, forces, or the total energy. A simulation is started by entering a time step size $\tau > 0$ in units of time and a number of integration steps that the system should be propagated.

3. When a simulation has been performed, the observables' outputs have been recorded into a file. The file's contents can be loaded again into a `readdy.Trajectory` object that can be used to produce trajectories compatible with the VMD molecular viewer [248].

Running a simulation based on the `readdy.Simulation` object invokes a simulation loop. The default simulation loop is given in Algorithm 2. Individual steps of the loop can be omitted. This enables the user to, e.g., perform pure PBRD simulations by skipping the calculation of forces. Performing a step in the algorithm leads to a call to the compute kernel interface, see Figure 4.2b. Depending on the selected compute kernel the call is then dispatched to the actual implementation. Compute kernel implementations (Figure 4.2c) are dynamically loaded at runtime from a plugin directory. This modularity allows `ReaDDy2` to run across many platforms although not every computing kernel may run on a given platform, such as a CUDA-enabled computing kernel. ReaDDy2 version 2.0.11 includes two iPRD computing kernels: a single threaded default computing kernel, and a dynamically-loaded shared-memory parallel kernel.

The computing kernels contain implementations for the single steps of the simulation loop. Currently, integrator and reaction handler are exchangeable by user-written C++ extensions. Hence, there is flexibility considering what is actually performed during one step of the algorithm or even what kind of underlying model is applied.

In comparison to the predecessor ReaDDy, the software is a complete rewrite and extension. The functionality of the Brownian dynamics integrator has been preserved, however the reaction handlers can behave slightly differently. In particular, if during an integration step a reaction conflict occurs, i.e., there are at least two reaction events which involve the same educt particles, only one of these events can be processed. One possibility of choosing the to-be processed event is the so-called UncontrolledApproximation, which draws the next reaction event uniformly from all events and prunes conflicting events. Another possibility is drawing the next reaction event from all events weighted by their respective reaction probability. Since this approach is loosely based on the reaction order in the Gillespie SSA (cf. Section 2.3.2), this reaction handler is named Gillespie in ReaDDy2.

With respect to the microscopic evaluation of a reaction event, the ReaDDy implementation places product particles of fission type reactions at a fixed distance, which is handled more flexibly in the current implementation, see Section 4.2.2.

4.3.2 Performance

To benchmark ReaDDy2, we use a reactive system with three particle species A, B, and C introduced in [68] with periodic boundaries instead of softly repelling ones. The simulation temperature is set to $T = 293\,\mathrm{K}$ and the diffusion coefficients are given by $D_\mathrm{A} = 143.1\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, $D_\mathrm{B} = 71.6\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, and $D_\mathrm{C} = 68.82\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, respectively.

Particles of these types are subject to the two reactions $\mathrm{A}+\mathrm{B} \rightharpoonup \mathrm{C}$ with microscopic association rate constant $\lambda_\mathrm{on} = 10^{-3}\,\mathrm{ns}^{-1}$ and reaction radius $R_1 = 4.5\,\mathrm{nm}$, and $\mathrm{C} \rightharpoonup \mathrm{A}+\mathrm{B}$ with microscopic dissociation rate constant $\lambda_\mathrm{off} = 5\times 10^{-5}\,\mathrm{ns}^{-1}$ and dissociation radius $R_2 = R_1$. Particles are subject to an harmonic repulsion interaction potential which reads

$$U(r) = \begin{cases} \frac{\kappa}{2}(r-\sigma)^2 & \text{, for } r \leq \sigma, \\ 0 & \text{, otherwise,} \end{cases} \tag{4.4}$$

where $r = \|\mathbf{x}_j - \mathbf{x}_i\|_2$ is the inter-particle distance, σ is the distance at which particles start to interact and $\kappa = 10\,\mathrm{kJ\,mol}^{-1}\,\mathrm{nm}^{-2}$ is the force constant. The interaction distance σ is defined as sum of radii associated to the particles' types, in this case $r_\mathrm{A} = 1.5\,\mathrm{nm}$, $r_\mathrm{B} = 3\,\mathrm{nm}$, and $r_\mathrm{C} = 3.12\,\mathrm{nm}$.

All particles are contained in a cubic box with periodic boundaries. The edge length is chosen such that the initial number density of all particles is $\rho_\mathrm{tot} = 3141\,\mathrm{nm}^{-3}$. This total density is distributed over the species, such that the initial density of A is $\rho_\mathrm{A} = \rho_\mathrm{tot}/4$, the initial density of B is $\rho_\mathrm{B} = \rho_\mathrm{tot}/4$, and the initial density of C is $\rho_\mathrm{C} = \rho_\mathrm{tot}/2$.

For the chosen microscopic rates these densities roughly resemble the steady-state of the system. The performance is measured over a simulation time span of 300 ns which is much shorter than the equilibration time of this system. Thus the overall number of particles does not vary significantly during measurement and we obtain the computation time at constant density.

In the following the benchmark results are presented. A comparison between the sequential reference compute kernel, the parallel implementation, and the previous Java-based ReaDDy [68] is made with respect to their performance when varying the number of particles in the system

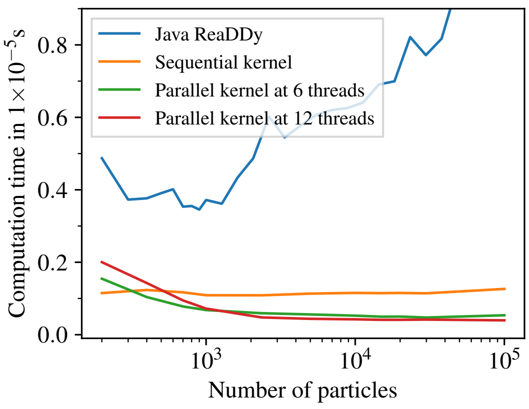


Figure 4.3: **Performance comparison.** Average computation time per particle and integration step for the benchmark system of Section 4.3.2 using a machine with an Intel Core i7 6850K processor, i.e., six physical cores at 3.8 GHz, and 32 GB DDR4 RAM at 2.4 GHz (dual channel). The number of particles is varied, but the particle density is kept constant. The sequential kernel (orange) has a constant per-particle CPU cost independent of the particle number. For large particle numbers, the parallel kernels are a certain factor faster (see scaling plot Figure 4.4). For small particle numbers of a few hundred the sequential kernel is more efficient. `ReaDDy2` is significantly faster and scales much better than the previous Java-based `ReaDDy` [68].

keeping the density constant. Since the particle numbers fluctuate the comparison is based on the average computation time per particle and per integration step (Figure 4.3). The sequential kernel scales linearly with the number of particles, whereas the parallelized implementation comes with an overhead that depends on the number of threads. The previous Java-based implementation does not scale linearly for large particle numbers, probably owing to Java's garbage collection. The parallel implementation starts to be more efficient than the sequential kernel given sufficiently many particles.

Figure 4.4 shows the strong scaling behavior of the parallel kernel, i.e. the speedup and efficiency for a fixed number of particles as a function of the used number of threads. For sufficiently large particle numbers, the kernel scales linear with the number of physical cores and an efficiency of around 80%. In hyperthreading mode, it then continues to scale linear with the number of virtual cores with an efficiency of about 55–60%.

The number of steps per day for a selection of particle numbers and kernel implementation is displayed in Table 4.1. For a system with 13, 000 particles and a time step size of $\tau = 1\,\mathrm{ns}$ (e.g., membrane proteins [68]), a total of 17 ms simulation time per day can be collected on a six-core machine (see Figure 4.3 for details). The current `ReaDDy2` kernels are thus suited for the detailed simulation of processes in the millisecond- to second timescale, which include many processes in sensory signaling and signal transduction at cellular membranes.

4.4 Results

In the following, several aspects of the model applied in `ReaDDy2` are validated and demonstrated by considering different application scenarios and comparing the results to analytically obtained

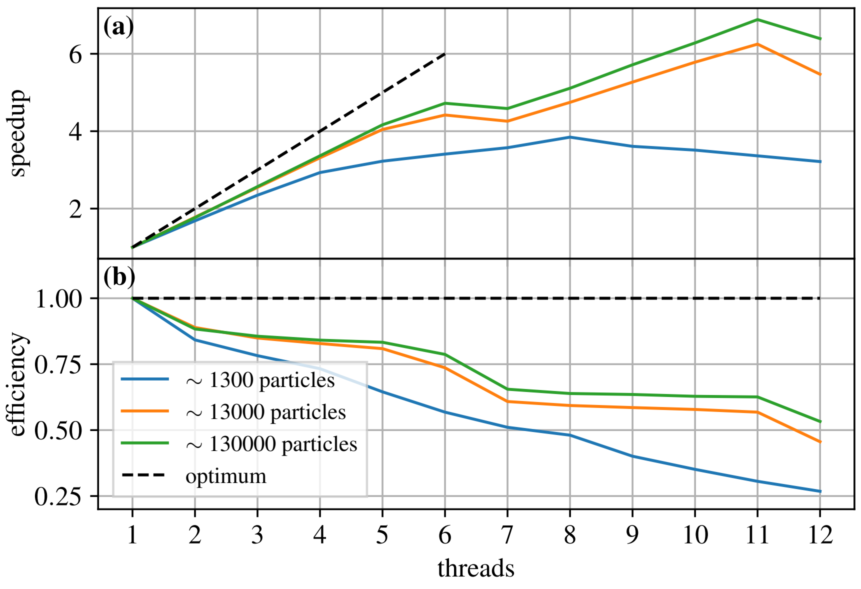


Figure 4.4: **Speedup and efficiency.** Parallel speedup and efficiency of the benchmark system of Section 4.3.2 as a function of the number of cores using the machine described in Figure 4.3. **(a)** Speedup with different numbers of cores compared to one core. Optimally one would like to have a speedup that behaves like the identity (black dashed line). **(b)** Efficiency is the speedup divided by the number of threads, i.e., how efficiently the available cores were used.

Table 4.1: **Number of steps per day for the benchmark system.** Number of time steps per day for benchmark system of Section 4.3.2 using the machine described in Figure 4.3. In case of the parallelized implementation the peak performance with respect to the number of threads is shown.

Approximate number of particles	Steps per day sequential kernel	Peak performance steps per day parallel kernel	Number of threads
250	2.8×10^8	2.6×10^8	4
1000	7.9×10^7	1.2×10^8	7
13000	5.6×10^6	1.7×10^7	11
40000	1.8×10^6	6.3×10^6	11

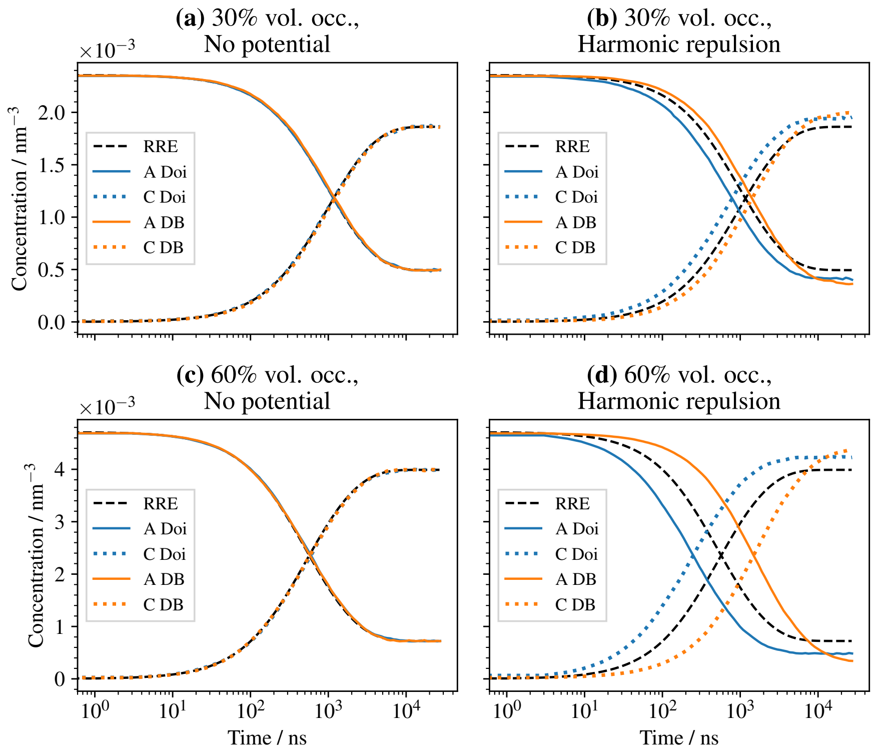


Figure 4.5: **Reaction kinetics and detailed balance.** Concentration time series of a the reaction-diffusion system introduced in Section 4.3.2 with the reversible reaction $A + B \leftrightharpoons C$. Compared are cases with and without harmonic repulsion (4.4). Additionally we compare two different reaction mechanisms, the Doi reaction scheme and the detailed balance (iPRD–DB) method for reversible reactions. **(a)** 30% volume occupation and no interaction potentials. **(b)** 30% volume occupation with harmonic repulsion between all particles. **(c)** 60% volume occupation and no interaction potentials. **(d)** 60% volume occupation with harmonic repulsion between all particles.

results, simulations from other packages, or literature data.

4.4.1 Reaction kinetics and detailed balance

We simulate the time evolution of particle concentrations of the benchmark system described in Section 4.3.2. In contrast to the benchmarks, the considered system initially only contains A and B particles at equal numbers. It then relaxes to its equilibrium mixture of A, B, and C particles (see Figure 4.5). Since the number of A and B molecules remain equal by construction, only the concentrations of A and C are shown.

In addition we compare the solutions with and without harmonic repulsion potentials (4.4) between all particles, as well as two different methods for executing the reactions: the Doi reaction scheme as described in Sections 2.3.3 and 4.2.2 and the detailed-balance reaction scheme iPRD–DB described in [243].

In contrast to Section 4.3.2, we construct a macroscopic reference system with rate constants

$k_\text{on} = 3.82 \times 10^{-1}\,\text{nm}^3\text{s}^{-1}$ and $k_\text{off} = 5 \times 10^{-5}\,\text{s}^{-1}$ resembling a cellular system. The microscopic reaction rate constants λ_on and λ_off are then chosen with respect to the reference system taking interaction potentials between A and B into account. In particular,

$$\lambda_\text{off} = k_\text{off}, \tag{4.5}$$

$$\lambda_\text{on} = \frac{k_\text{on}}{V_\text{eff}}, \tag{4.6}$$

where $V_\text{eff} = \int_0^R \exp(-\beta U) 4\pi r^2 \mathrm{d}r$ is the accessible reaction volume, R the reaction radius, β the inverse thermal energy, and U the pair potential. The harmonic repulsion potential reduces V_eff with respect to the volume of the reactive sphere. The expression (4.6) originates from an approximation for k_on in a sufficiently well-mixed (i.e., reaction–limited) and sufficiently diluted system. The derivation can be found in [243] based on calculating the total association rate constant k_on for an isolated pair of A and B particles. In this case one obtains $\lambda_\text{off} = 5 \times 10^{-5}\,\text{ns}^{-1}$ for the microsopic dissociation rate constant. The microscopic association rate constant reads $\lambda_\text{on} = 10^{-3}\,\text{ns}^{-1}$ for the noninteracting system and $\lambda_\text{on} = 2.89 \times 10^{-3}\,\text{ns}^{-1}$ for the interacting system. Note that for non-reversible binary reactions without interaction potentials the formula provided by [56, 64, 65] describes the relation between λ and k for slow diffusion encounter. In the case of non-reversible binary reactions with interaction potentials and slow diffusion encounter such a relation can still be numerically computed [249].

Using the macroscopic rate constants k_on and k_off, a solution can be calculated for the mass-action RRE (cf. Section 2.3.1). This solution serves as a reference for the non-interacting system (no potentials), because the system parameters put the reaction kinetics in the mass-action limit.

In the non-interacting system, the `ReaDDy2` solution and the RRE solution indeed agree (cf. Figure 4.5a,c). In the case of interacting particles, see Figure 4.5b,d; an exact reference is unknown. We observe deviations from the RRE solution that become more pronounced with increasing particle densities. A difference between the two reaction schemes can also be seen. The Doi reaction scheme shows faster equilibration compared to RRE for increasing density, whereas the iPRD–DB scheme shows slower equilibration, as it has a chance to reject individual reaction events based on the change in potential energy. Thus an increased density leads to more rejected events, consistent with the physical intuition that equilibration in a dense system should be slowed down. Furthermore the equilibrated states differ depending on the reaction scheme, showing a dependence on the particle density. For denser systems the iPRD–DB scheme favors fewer A and B particles than the Doi scheme, consistent with the density-dependent equilibria described in [243].

4.4.2 Diffusion

Next we simulate and validate the diffusive behavior of non-interacting particle systems and the subdiffusive behavior of dense interacting particle systems. The simulation box contains particles with diffusion coefficient D_0 and is equipped with softly repelling walls, in order to introduce finite size effects. The observations are carried out with and without interaction potential. In the case without interaction potential we compare with an analytical solution and the case of an interaction potential is compared to the literature.

Length x is given in units of σ, time t is given in units of σ^2/D_0, and energy is given in units of $k_B T$. The cubic box has an edge length of $\ell \approx 28\sigma$.

The non-interacting particle simulation has a mean-squared displacement of particles in agreement with the analytic solution given by Fick's law for diffusion in three dimensions

$$\langle (\mathbf{x}_t - \mathbf{x}_0)^2 \rangle = 6 D_0 t, \tag{4.7}$$

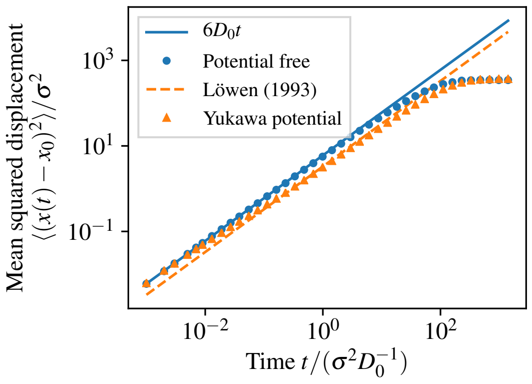


Figure 4.6: **Diffusion in crowded environments.** Mean squared displacement as a function of time. Multiple particles are diffusing with intrinsic diffusion coefficient D_0 in a cubic box with harmonically repelling walls. Triangles were obtained by using the Yukawa repulsion potential (4.8) between all particles. The dashed line represents an effective diffusion coefficient from the literature [250] for the same Yukawa repulsion potential.

where $\mathbf{x}_t$ is the position of a particle at time t (see Figure 4.6). For long timescales $t \geq 10^1$, transport is obstructed by walls, which results in finite size saturation.

Figure 4.6 also shows that more complex transport can be modeled, as, e.g., found in crowded systems. Particles interact via the Yukawa potential [251]

$$U(r) = \begin{cases} U_0\sigma \exp\left(-\lambda \frac{r-\sigma}{\sigma}\right)/r & \text{, for } r \leq r_c, \\ 0 & \text{, otherwise,} \end{cases} \tag{4.8}$$

where $U_0 = k_BT$ is a repulsion energy, σ is the length scale, $\lambda = 8$ is the screening parameter, and $r_c = 2.5\sigma$ the cutoff radius.

The particle density is $n\sigma^3 = 0.6$ with n being the number density. In such a particle system, the mean-squared displacement differs significantly from the analytical result for free diffusion after an initial time $t \geq 10^{-2}$ in which particles travel their mean free path length with diffusion constant D_0. At intermediate timescales $t \in [10^{-2}, 10^{-1})$, particle transport is subdiffusive due to crowding. At long timescales, $t \in [10^{-1}, 10^1)$, the particles are again diffusive with an effective diffusion coefficient D that is reduced to reflect the effective mobility in the crowded systems. We compare this to an effective diffusion coefficient obtained by Brownian dynamics simulations from Löwen and Szamel [250] and find that they qualitatively agree. For large timescales $t \geq 10^1$ finite size saturation can explicitly be observed as almost every particle has been repelled at least once by the boundaries.

To quantitatively compare the long-time effective diffusion coefficient D, we set up 1100 particles in a periodic box without repelling walls with the edge length chosen to give the densired density $n\sigma^3 = 0.6$. The cutoff of the potential (4.8) is set to $r_c = 5\sigma$, where $U(r_c) < 10^{-14}k_BT$. The particle suspension is equilibrated for at least $t_{\text{eq}} \geq 3$ with a time-step size of $\tau = 10^{-5}$. We observe the mean squared displacement until $t_{\text{obs}} = 4.5$ and measure the diffusion coefficient as the slope of a linear function for $t \in [4, 4.5)$. We obtain $D/D_0 = 0.54 \pm 0.01$, which agrees with the reference value [250] $D^{\text{ref}}/D_0 = 0.55 \pm 0.01$.

Table 4.2: **Thermodynamic equilibrium properties of a Lennard–Jones colloidal fluid in a** (N, V, T) **ensemble.** Results of the ReaDDy 2 framework are compared to other simulation frameworks and analytical results for validation.

	density ρ^*	pressure P^*	energy u^*
ReaDDy 2	0.3	1.0253 ± 0.0004	-1.6704 ± 0.0003
HALMD [252]	0.3	1.0234 ± 0.0003	-1.6731 ± 0.0004
Johnson et al. [253]	0.3	1.023 ± 0.002	-1.673 ± 0.002
Ayadim et al. [254]	0.3	1.0245	-1.6717
ReaDDy 2	0.6	3.711 ± 0.002	-3.2043 ± 0.0004
HALMD [252]	0.6	3.6976 ± 0.0008	-3.2121 ± 0.0002
Johnson et al. [253]	0.6	3.69 ± 0.01	-3.212 ± 0.003
Ayadim et al. [254]	0.6	3.7165	-3.2065

4.4.3 Thermodynamic equilibrium properties

We validate that ReaDDy2's integration of equations of motion yields the correct thermodynamics of a Lennard-Jones colloidal fluid in an (N, V, T) ensemble. To this end, we simulate a system of N particles confined to a periodic box with volume V at temperature T. The results and comparisons with other simulation frameworks and analytical results are shown in Table 4.2. The particles interact via the Lennard–Jones potential

$$U(r) = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right],$$

with ε being the depth of the potential well and σ the diameter of particles. The potential is cut off at $r_C = 4\sigma$ and shifted to avoid a discontinuity. The rescaled temperature is $T^* = k_B T \varepsilon^{-1} = 3$. We perform simulations of the equilibrated Lennard–Jones system for 10^6 integration steps with rescaled time step size $\tau^* = 10^{-4}$. Time units are σ^2/D and are determined by the self-diffusion coefficient D of the particles. We measure the rescaled pressure $P^* = P\sigma^3\varepsilon^{-1}$ by estimating the virial term from forces acting in the system as described in [255]. Additionally, we measure the rescaled potential energy per particle $u^* = UN^{-1}\varepsilon^{-1}$. Both pressure and potential energy are calculated every 100th time step. This sampling gives rise to the mean and its error of the mean given for the ReaDDy2 results in Table 4.2. Comparing HALMD [252] and ReaDDy2, the latter shows larger energy and pressure in the third decimal place for the lower density $\rho^* = 0.3$. For the higher density $\rho^* = 0.6$ pressure differs in the first decimal place and energy in the second. This can be explained by ReaDDy2 using an Euler–Maruyama scheme (4.2) to integrate motion of particles, which has a discretization error of first order in the time step size $\mathcal{O}(\tau)$. On the other hand HALMD uses a Velocity–Verlet method [256], which has a discretization error of second order in the time step size $\mathcal{O}(\tau^2)$.

4.4.4 Topology reactions

We illustrate ReaDDy2's ability to model complex reactions between multi-particle complexes, called "topology reactions". We model polymers as linear chains of beads, held together by harmonic bonds and stiffened by harmonic angle potentials.

When considering just one worm-like chain with a certain amount of beads n, its equilibrium

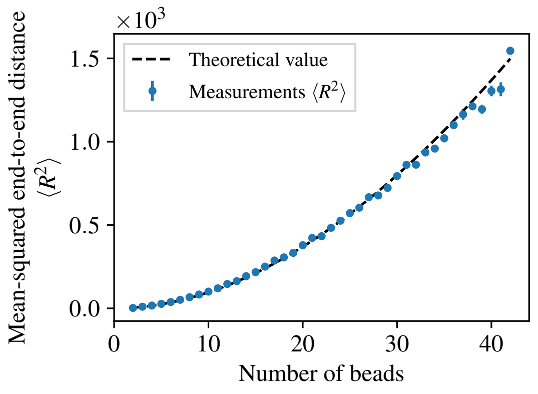


Figure 4.7: **Mean-squared end-to-end distance of worm-like chains.** The theoretical mean-squared end-to-end distance of worm-like chains as a function of number of beads (4.9) is compared to simulation data obtained from linear chains of beads as described in Section 4.2.3. Error bars depict errors over the mean from multiple measurements.

mean-squared end-to-end distance should behave like [257]

$$\langle R^2 \rangle = 2l_p R_{\max} - 2l_p^2 \left(1 - \exp\left(-\frac{R_{\max}}{l_p}\right)\right), \tag{4.9}$$

where $l_p = 4lk(k_B T)^{-1}$ is the persistence length, $R_{\max} = (n-1)l$ the chain contour length, l the bond length, and k the force constant of the harmonic angles. In order to verify that the considered chain model obeys the mechanics of a worm-like chain, the theoretical mean-squared end-to-end distance (4.9) can be compared to observations from simulations, see Figure 4.7. For each fixed number of beads, an isolated chain was relaxed into an equilibrium state without performing topology reactions, yielding a squared end-to-end distance at the end of the simulation. This experiment was repeated 51 times. From the figure it can be observed that there is good agreement between the theoretical and measured mean-squared end-to-end distances. A more detailed description of the used simulation parameters can be found in Appendix A.3.

In a system with many of these chains, we introduce two different particle types for the beads. Either they are head particles and located at the ends of a polymer chain or they are core particles and located between the head particles, as shown in Figure 4.8a,c in blue and orange, respectively.

We impose two different topology reactions in the system with many chains (Figure 4.8a):

1. Association: Two nearby head particles (distance $\leq R$) can connect with rate λ_1. The topology is changed by adding an edge between the connected particles, resulting in the addition of one bond and two angle potentials. Additionally, the particle types of the two connected particles change from “head” to “core”.

2. Dissociation: A chain with n particles can dissociate with microscopic rate $n\lambda_2$, such that longer chains have a higher probability to dissociate than shorter chains. When a dissociation occurs, a random edge between two core particles is removed. The particle types of the respective core particles are changed to “head”. As a result, the graph decays into two connected components which subsequently are treated as autonomous topology instances.

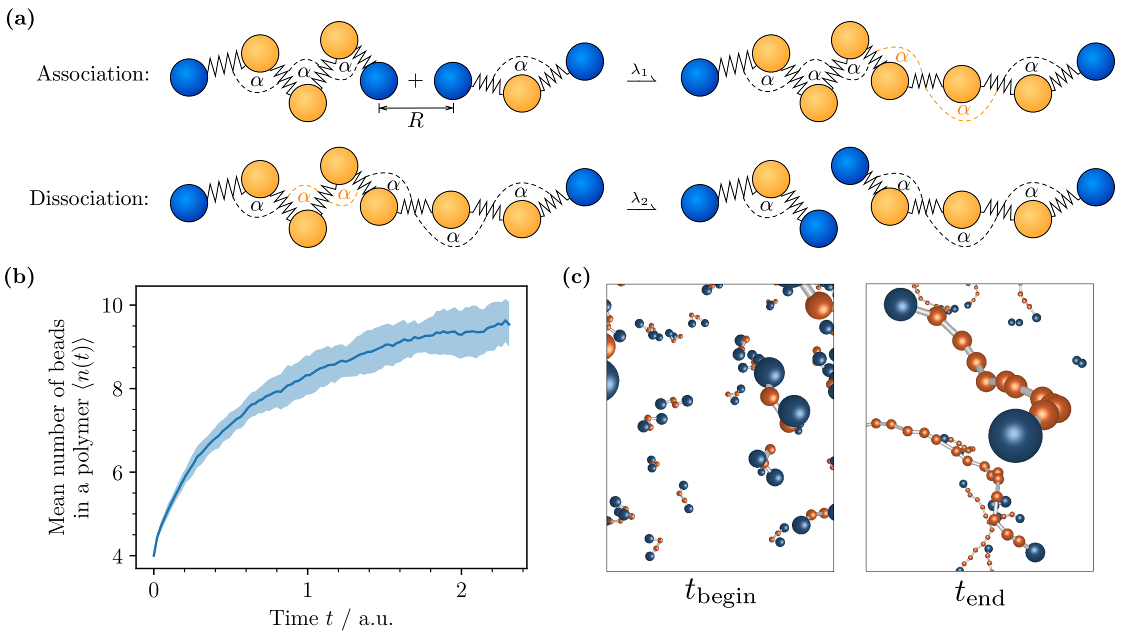


Figure 4.8: **Topology reactions example.** Illustrative simulation of polymer assembly/disassembly using topology reactions. **(a)** Sketch of the involved topology reactions. *Association*: When two ends of different topologies come closer than R, there is a rate λ_1 that an edge is formed. *Dissociation*: The inverse of association with a rate λ_2 and a randomly drawn edge that is removed. **(b)** The number of beads in a polymer $\langle n(t) \rangle$ over time averaged over 15 realizations. **(c)** Two representative particle configurations showing the initial state and the end state at time t_{begin} and t_{end}, respectively.

The temporal evolution of the average length of polymer chains is depicted in Figure 4.8b. The simulation was performed 15 times with an initial configuration of 500 polymers containing four beads each. After sufficient time $\langle n(t) \rangle$ reaches an equilibrium value. Over the course of the simulation the polymers diffuse and form longer polymers. This can also be observed from the two snapshots shown in Figure 4.8c, depicting a representative initial configuration at t_{begin} and a representative configuration at the end of the simulation at time. In that case, there are polymers of many different lengths.

4.4.5 Nontrivial bimolecular association kinetics at high concentrations

This section studies a biologically inspired system with three macromolecules A, B, and C, that resemble, e.g., proteins in cytosol. The macromolecules A and B can form complexes C that also can dissociate back into their original components, i.e., we introduce reactions

$$\mathrm{A} + \mathrm{B} \rightleftharpoons \mathrm{C}. \tag{4.10}$$

This form of interaction has been studied for proteins bovine serum albumin and hen egg white lysozyme in coarse-grained atomistic detail in [258] and for Barnase and Barstar in [259]. Here, we consider the case where the association reaction of (4.10) does not preserve volume, i.e., the complex C is more compact.

The presence of ions in aqueous solutions has effects on protein interactions [260], therefore we assume the reversibly associating macromolecules to be weakly charged and thus subject to the Debye–Hückel interaction potential [261] including an additional repulsion term

$$U_{s_1 s_2}(r) = q_{s_1} q_{s_2} \frac{e^2}{\varepsilon_0 \varepsilon_r} \frac{\exp(-\kappa r)}{r} + U_r \left(\frac{\sigma_{s_1 s_2}}{r} \right)^{12}, \tag{4.11}$$

where $s_1, s_2 \in \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$, q are partial charges associated with the macromolecules, e is the elementary charge, ε_0 is the vacuum permittivity, ε_r is the relative permittivity of an aqueous solution, κ is the screening parameter that describes shielding due to ions in the solution, U_r is the repulsion energy, and $\sigma_{s_1 s_2} = r_{s_1} + r_{s_2}$ is the sum of two particle radii. Here, we do not take hydration effects into account.

We investigate the equilibrium constant $K = [\mathrm{A}][\mathrm{B}]/[\mathrm{C}]$ for different number densities $n = (N_A + N_B)/2 + N_C$. In case of a reversibly associating fluid described by the law of mass action, the equilibrium constant is given by $K = k_{\text{off}}/k_{\text{on}}$, where k_{on} is the macroscopic association rate constant of (4.10) and k_{off} the respective dissociation rate constant. In a well-mixed (i.e., reaction-limited) and sufficiently diluted system, k_{on} can be approximated as in Section 4.4.1. However, for a diffusion-influenced process which we consider here, k_{on} is typically understood as a harmonic mean of encounter and formation rates [262–265], i.e., $k_{\text{on}}^{-1} = k_{\text{enc}}^{-1} + k_{\text{form}}^{-1}$. At low densities, only two-body interactions between A and B determine the on-rate constant, in this limit, k_{on} can be evaluated numerically as a function of the microscopic association rate constant λ_{on} in the presence of the interaction potential, based on solving the Smoluchowski diffusion equation with a sink term that accounts for the volume reaction model, see [249]. Furthermore, in dense reversibly associating fluids, many-body interactions have an influence on k_{on}, in particular due to competition for reactants, clustering, volume exclusion, and caging [265].

Thus, it is challenging to find a consistent analytical description over multiple orders of magnitudes in density. In contrast, we perform an empirical evaluation by simulations as shown in Figure 4.9. To this end, we set up 6 simulations for different $n \in [2 \times 10^1, 1.5 \times 10^4]$ in a constant volume which then are allowed to relax into an equilibrium state subject to detailed-balance and yield a measurement $K(n)$. The exact simulation parameters can be found in the Appendix A.2.

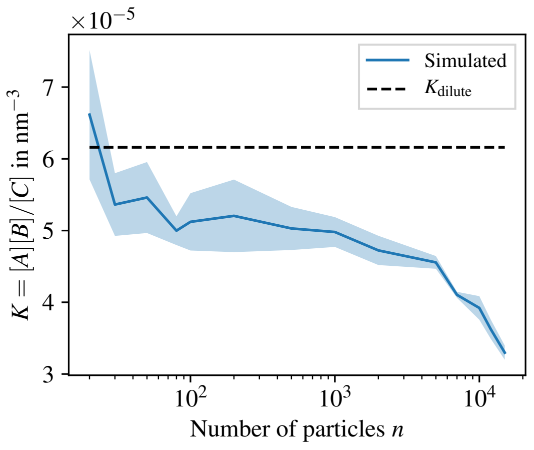


Figure 4.9: **Equilibrium constant transition from dilute to dense systems.** The equilibrium constant K is obtained by simulation for different choices of the number of particles $n = (N_A + N_B)/2 + N_C$ which corresponds to a density due to constant volume of the simulation box and compared to an analytically obtained equilibrium constant of a dilute system (dashed line). The number of particles n remains constant during the course of a simulation. The shaded areas are standard deviations from the recorded data.

The reference value for the dilute case is given by $K_{\text{dilute}} = k_{\text{off}}/k_{\text{on}}^{\text{dilute}}$, where $k_{\text{off}} = \lambda_{\text{off}}$ and $k_{\text{on}}^{\text{dilute}}$ is a function of the microscopic association rate constant λ_{on} as well as the interaction potential (4.11) and is numerically computed as described in [249].

We show that the reference value K_{dilute} is recovered by the simulation for low densities. For increasing densities more complex behavior can be observed. In particular, there is a drop in the value of K for $n \gtrsim 10^2$ which then is followed by a roughly stable regime up to $n \approx 5 \times 10^3$. For even higher densities, the equilibrium state is dominated by the complexes C likely due to finite size of the simulation volume. This drop in the equilibrium constant is in accord with Le Châtelier's principle [266], i.e., the system prefers the state of lower free energy.

4.5 Availability and Future Directions

We have described the iPRD simulation framework `ReaDDy2` for combined particle interaction dynamics and reaction kinetics, which permits to conduct highly realistic simulations of signal transduction in crowded cellular environments or chemical nanoreactors with complex geometries. `ReaDDy2` follows up upon and significantly extends the simulation package `ReaDDy`. `ReaDDy2` is significantly faster than its predecessor, it can be easily installed as a Python conda package, and it can be flexibly used and reconfigured via its Python interface.

In comparison to molecular dynamics software packages, `ReaDDy2` does not include long range interactions. The software comes with a set of default interaction potentials. These include, e.g., harmonic repulsion which can model steric repulsion, Lennard–Jones interaction, and screened electrostatics which provide a way to model charged interaction at short ranges. Furthermore, `ReaDDy2` allows for implementation of any short-ranged potential via a C++ interface. It is possible to implement and subsequently use hydration models which are short-ranged [267, 268] in the `ReaDDy2` framework. Hydrodynamic interactions are currently not included. They can be added by, e.g., providing an appropriate integrator which represents these interactions by a particle pairwise friction tensor [247].

Currently all pair potentials implemented in `ReaDDy2` are isotropic, however anisotropic interactions can be emulated by using particle complexes, in particular allowing for patchy particles. If the particles and interactions should be anisotropic themselves, a new computation kernel or appropriate integrator can be implemented into the framework via the C++ interface.

We have conducted a set of numerical studies, showing that `ReaDDy2` produces quantitatively accurate results where references from analytical solutions or other simulation packages were available, and physically meaningful results where reference solutions were not available.

For a quick and easy start into simulating and developing with `ReaDDy2` step by step tutorials, sample code, and further details are available online (https://readdy.github.io/). The software itself is Open Source and available under a permissive license in order to enable a broad group of people to run simulations without forcing them to make their own work public.

`ReaDDy2` has been designed to be easily extensible. Planned extensions include simulation kernels for specialized hardware platforms, such as graphics processors and highly parallel HPC environments. Also planned is a MD-GFRD integrator [238] to speed up computations in dilute systems, and a particle-based membrane model as described in [234] that reproduces mechanical properties of cellular membranes.

In its current state, membranes can be modeled in terms of external forces, i.e., constraining particles onto two-dimensional surfaces. As these constraints only apply to selected particle types, it is possible to, e.g., grow polymers against a static membrane, where one end is anchored.

Chapter 5

Discovering governing reactions from concentration data

The results of this chapter were originally presented in Ref. P2:

> M. Hoffmann, C. Fröhner, F. Noé. "Reactive SINDy: Discovering governing reactions from concentration data". In: *J. Chem. Phys.* (2019). URL: `https://doi.org/10.1063/1.5066099`.

Text and illustrations have been adopted largely unchanged in this document. Reprinted from The Journal of chemical physics "Reactive SINDy: Discovering governing reactions from concentration data", Hoffmann, Fröhner, and Noé, 2019, with the permission of AIP Publishing.

Moritz Hoffmann (MH) and Christoph Fröhner (CF) developed this work collaboratively. MH and CF contributed roughly equally to this work and are consequently both shared first author. The work is therefore also part of the dissertation of CF. The author contributions were as follows: MH, CF, and Frank Noé (FN) conceived the project and laid out the theory. MH and CF set up the software pipeline for generation of training data and cross validation procedures. MH implemented the minimization procedure. MH laid out and performed numerical experiments in the low-noise limit (Section 5.3.1.1). CF ran the cross validation for noisy measurements (Sections 5.3.1.2 and 5.3.1.3). CF applied the `reactive SINDy` method to the MAPK example (Section 5.3.2). MH set up the application of `reactive SINDy` to the predator-prey example (Section 5.3.3). MH and CF analyzed and visualized all resulting data. All contributors wrote the paper.

In this chapter we introduce `reactive SINDy`, an extension of SINDy (introduced in Section 3.5) that focuses on the detection of reaction networks from species concentration data. The method is applied to several model systems to study its correctness and effectiveness. Code related to this chapter can be found under `https://github.com/readdy/readdy_learn`.

5.1 Introduction

Mapping out the reaction networks behind biological processes, such as gene regulation in cancer [17], is paramount to understanding the mechanisms of life and disease. A well-known example of gene regulation is the lactose operon whose crystal structure was resolved in [269] and dynamics were modeled in [270]. The system's "combinatorial control" in E. coli cells was quantitatively investigated in [271], in particular studying repression and activation effects. These gene-regulatory effects often appear in complex networks [272] and there exist databases resolving these for certain types of cells, e.g., E. coli cells [273] and yeast cells [274]. Another example where mapping the active reactions is important is that of chemical reactors [275], where understanding which reactions are accessible for a given set of educts and reaction conditions is important to design synthesis pathways [276, 277].

The traditional approach to determine a reaction network is to propose the structure of the network based on chemical insight and subsequently fit the parameters given available data [22]. To decipher complex reaction environments such as biological cells, it would be desirable to have a data-driven approach that can answer the question which reactions are underlying a given observation, e.g., the time series of a set of reactants. However, in sufficiently complex reaction environments the number of reactive species and possible reactions is practically unlimited–as an illustration, consider vast amount of possible isomerizations and post-translational modifications for a single protein molecule. Therefore, the more specific formulation is "given observations of a set of chemical species, what is the *minimal set* of reactions necessary to explain their time evolution?". This formulation calls for a machine learning method that can infer the reaction network underlying the observation data.

Knowledge about the reaction network can be applied to parameterize other numerical methods to further investigate the processes at hand. Such methods include particle-based approaches derived from the CME [60, 62, 110, 111, 278], as well as highly detailed but parameter-rich methods such as PBRD or iPRD [P1, 57, 58, 68, 69, 243, 279] capable of fully resolving molecule positions in space and time–see Section 2.4, Chapter 4, as well as Refs. 73, 280 for recent reviews.

Existing methods to infer regulatory networks include ARACNE [281] that uses experimental essay data and information theory, as well as the likelihood approach presented in [282] that takes the stochasticity of observed reactant time series into account.

The method presented in this chapter can identify underlying complex reaction networks from concentration time series by following the law of parsimony, i.e., by inducing sparsity in the resulting reaction network. This promotes the interpretability of the model and avoids overfitting. We formulate the problem as data-driven identification of a dynamical system, which renders the method consistent with and an extension of the framework of sparse identification of nonlinear dynamics (SINDy) (cf. Section 3.5). Specifically, the problem of identifying a reaction network from time traces of reactant concentrations can be solved by finding a linear combination from a library of candidate nonlinear functions (ansatz functions) that each corresponds to a reaction acting on a set of reactants. With this formulation, the reaction rates can be determined *via* regression. Sparsity is induced by equipping the regression algorithms with a sparsity inducing regularization.

We extend and apply SINDy to the case of learning reaction networks from non-equilibrium concentration data. Similar approaches make use of SINDy but do not resolve specific reactions [208], use weak formulations to avoid numerical temporal derivatives [283], or use compressive sensing and sparse Bayesian learning [284].

Our extension of the original SINDy method mostly involves estimating parameters which are coupled across the equations of the arising dynamical system. In the context of learning reaction networks this means that we look for specific reactions and their rate constants that might have lead to the observations instead of net flux across species. We demonstrate the algorithm on a gene-regulatory network in three different scenarios of measurement: When there is no noise in the data we can find, given sufficient amounts of data, all relevant processes of the ground truth. If there is noise in the data we converge to the correct reaction network and rates with decreasing levels of noise. The third scenario generalizes the method to two measurements with different initial conditions, also converging to the correct model with decreasing levels of noise.

We additionally demonstrate the algorithm on time series data of the mitogen activated protein kinases (MAPK) pathway as an example for a bimodal system and on time series data of the Lotka–Volterra system (cf. Section 2.3.1) which describes oscillatory predator-prey dynamics subject to social friction. In both systems `reactive SINDy` recovers the generating reaction network whereas non-sparse estimation detects many spurious processes.

5.2 Reactive SINDy

We are observing the concentrations of S chemical species in time t according to the RRE model subject to the LMA (see Section 2.3.1).

To disentangle Equation (2.23) into single reactions, we choose a library of R possible ansatz reactions that each represent a single reaction:

$$\mathbf{y}_r(\mathbf{c}_t) = \begin{pmatrix} y_{r,1}(\mathbf{c}_t) \\ \vdots \\ y_{r,S}(\mathbf{c}_t) \end{pmatrix}, \quad r = 1, \dots, R. \tag{5.1}$$

With this ansatz, the reaction dynamics (2.23) becomes a set of linear equations

$$\dot{\mathbf{c}}_t^{(i)} = \sum_{r=1}^{R} y_{r,i}(\mathbf{c}_t)\xi_r, \quad i = 1, \dots, S, \tag{5.2}$$

where ξ_r are unknown parameters that represent the to-be estimated macroscopic rate constants. The two reactions in the example (2.24)-(2.25) would be modeled by the functions

$$\begin{aligned} y_1(\mathbf{x}) &= (-x_1, x_1, 0)^\top, \\ y_2(\mathbf{x}) &= (-x_1, 0, x_1)^\top, \end{aligned}$$

illustrating that the values of the coefficients ξ_1 and ξ_2 can be used to decide whether a single reaction is present and to what degree.

Now suppose we have measured the concentration vector (2.20) at T time points $t_1 < \dots < t_T$. We represent these data as a matrix

$$\mathbf{X} = \begin{pmatrix} \mathbf{c}_{t_1} & \mathbf{c}_{t_2} & \cdots & \mathbf{c}_{t_T} \end{pmatrix}^\top \in \mathbb{R}^{T\times S}. \tag{5.3}$$

Given this matrix, a library $\Theta : \mathbb{R}^{T\times S} \to \mathbb{R}^{T\times S\times R}$, $\mathbf{X} \mapsto \begin{pmatrix}\theta_1(\mathbf{X}) & \theta_2(\mathbf{X}) & \cdots & \theta_R(\mathbf{X})\end{pmatrix}$ of R candidate (ansatz) reactions can be proposed with corresponding reaction functions

$$\theta_r(\mathbf{X}) = \begin{pmatrix} \mathbf{y}_r(\mathbf{X}_{1*})^\top \\ \vdots \\ \mathbf{y}_r(\mathbf{X}_{T*})^\top \end{pmatrix} \in \mathbb{R}^{T\times S}, \quad r = 1, \dots, R, \tag{5.4}$$

where $\mathbf{X}_{i*}$ denotes the i-th row in $\mathbf{X}$. Applying the concentration trajectory to the library yields $\Theta(\mathbf{X}) \in \mathbb{R}^{T\times S\times R}$.

The goal is to find coefficients $\Xi = \begin{pmatrix}\xi_1 & \xi_2 & \cdots & \xi_R\end{pmatrix}^\top$, so that

$$\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi = \sum_{r=1}^{R} \theta_r(\mathbf{X})\xi_r. \tag{5.5}$$

Table 5.1: Initial conditions **(a)** and **(b)** used to generate concentration time series for gene-regulatory network in Section 5.3.1. Reaction rates can be found in Table B.1.

	$\mathrm{DNA_A}$	$\mathrm{mRNA_A}$	A	$\mathrm{DNA_B}$	$\mathrm{mRNA_B}$	B	$\mathrm{DNA_C}$	$\mathrm{mRNA_C}$	C
(a)	1	2	0	1	0	3	1	0	0
(b)	1	1.5	0	1	0	2	1	0	1

In particular, the system is linear in the coefficients Ξ, which makes regression tools such as elastic net regularization [285] applicable. To this end, one can consider the regularized minimization problem (`reactive SINDy`):

$$\hat{\Xi} = \arg\min_{\Xi} \left(\frac{1}{2T} \left\| \dot{\mathbf{X}} - \Theta(\mathbf{X})\Xi \right\|_F^2 + \alpha\lambda \|\Xi\|_1 + \alpha(1-\lambda)\|\Xi\|_2^2 \right) \quad \text{subject to } \Xi \geq 0 \tag{5.6}$$

Here, $\|\cdot\|_F$ denotes the Frobenius norm, $\lambda \in [0,1]$ is a hyperparameter that interpolates linearly between LASSO [286, 287] and Ridge [288] methods, and $\alpha \geq 0$ is a hyperparameter that, depending on λ, can induce sparsity and give preference to smaller solutions in the L_1 or L_2 sense. For $\alpha = 0$ the minimization problem reduces to standard least squares (LSQ) with the constraint $\Xi \geq 0$. `reactive SINDy` (5.6) is therefore a generalization of the SINDy method to vector-valued ansatz functions.

Since often only the concentration data $\mathbf{X}$ is available but not its temporal derivative, $\dot{\mathbf{X}}$ is approximated numerically by second order finite differences with the exception of boundary data. Once the pair $(\mathbf{X}, \dot{\mathbf{X}})$ is obtained, the problem becomes invariant under temporal reordering. Hence, when presented with multiple trajectories the data matrices $\mathbf{X}_i$ and $\dot{\mathbf{X}}_i$ can simply be concatenated.

In order to solve (5.6) the numerical sequential LSQ minimizer SLSQP [192] is applied via the software package SciPy [154].

5.3 Results

5.3.1 Application to a gene-regulatory network

We estimate the reactions of a gene-regulatory network from time series of concentrations of the involved molecules. Let $S := \{A, B, C\}$ be a set of three species of proteins which are being translated each from their respective mRNA molecule. Each mRNA in turn has a corresponding DNA which it is transcribed from. The proteins and mRNA molecules decay over time whereas the DNA concentration remains constant. The network contains reactions of the following form [289]

$$\begin{aligned}
\mathrm{DNA}_i &\rightharpoonup \mathrm{DNA}_i + \mathrm{mRNA}_i && \text{(transcription)},\\
\mathrm{mRNA}_i &\rightharpoonup \mathrm{mRNA}_i + i && \text{(translation)},\\
\mathrm{mRNA}_i &\rightharpoonup \emptyset && \text{(decay of mRNA)},\\
i &\rightharpoonup \emptyset && \text{(decay of protein)},\\
i + \mathrm{mRNA}_j &\rightharpoonup i && \text{(regulation of } j \in S\text{)},
\end{aligned}$$

for each of the species $i \in S$.

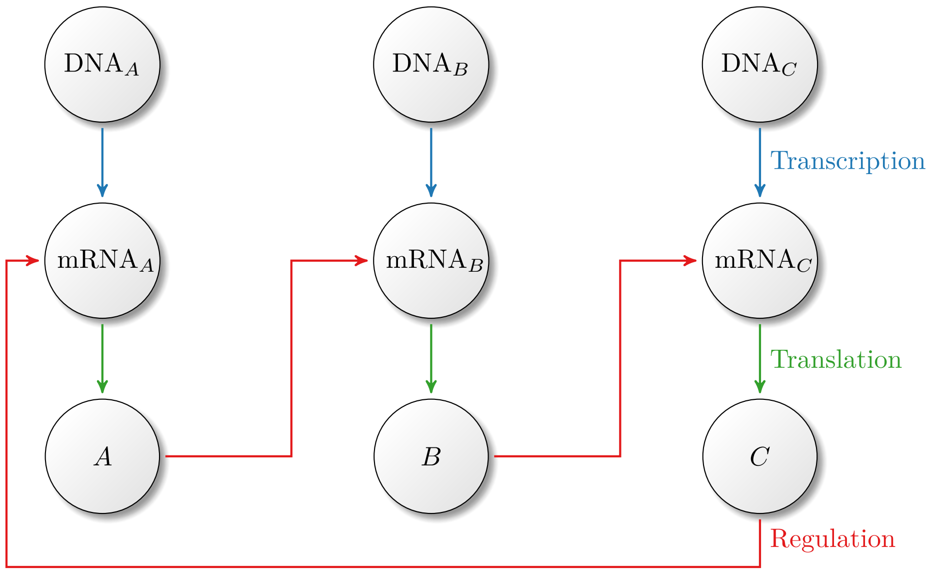


Figure 5.1: **Gene-regulatory network.** The regulation network example described in Section 5.3. Each circle depicts a species, each arrow corresponds to one reaction. Blue arrows denote transcription from DNA to mRNA, green arrows denote translation from mRNA to protein, and red arrows denote the regulatory network.

These reactions model a regulation of species j by virtue of the fact that the transcription product inhibits the transcription processes. In our example proteins of type A regulate the $\mathrm{mRNA_B}$ molecules, proteins of type B regulate the $\mathrm{mRNA_C}$ molecules and proteins of type C regulate the $\mathrm{mRNA_A}$ molecules (see Figure 5.1). Using this reaction model, time series of concentrations are generated using the rates given in Table B.1 under the initial condition described in Table 5.1a, which were chosen so that all the reactions in the reaction model significantly contribute to the temporal evolution of the system's concentrations. The generation samples the integrated equations equidistantly with a discrete time step of $\tau = 3 \cdot 10^{-3}$ yielding 667 frames which amounts to a cumulative time of roughly $T = 2$.

The proposed estimation method is applied to analyze these time series of concentrations in order to recover the underlying reaction network from data. To this end we use the library of ansatz functions given in Table B.1, which contains a large number of possible reactions, only few of which are actually part of the model.

5.3.1.1 The low-noise limit

We first demonstrate that the true reaction network can be reconstructed when using a finite amount of observation data without additional measurement noise, i.e., the observations are reflecting the true molecule concentrations at any given time point. The minimization problem (5.6) is solved using the concentration time series shown in Figure 5.1b.

We first set the hyperparameter $\alpha = 0$ in the minimization problem (5.6), which results in constrained least-squares regression without any of the regularization terms. In this case we estimate a reaction network that can reproduce the observations almost exactly (cf. Figure 5.2). However, the result is mechanistically wrong as the sparsity pattern does not match the reaction network used to generate the data. On the one hand many spurious reactions are estimated that were not in the true reaction scheme and would lead to wrong conclusions about the mechanism, such as $\mathrm{A} + \mathrm{A} \rightharpoonup \mathrm{A}$ and $\mathrm{A} + \mathrm{C} \rightharpoonup \mathrm{C}$.

More dramatically, the reaction responsible for the decay of A particles is completely ignored (cf. Figure 5.3).

Next, we sought sparse solutions by using $\alpha > 0$ and additionally eliminating reactions with rate constants smaller than a cutoff value κ. For a suitable choice of hyperparameters $\alpha \approx 1.91 \cdot 10^{-7}$, $\lambda = 1$, and $\kappa = 0.22$, a sparse solution is obtained that finds the correct reaction scheme and also recovers the decay reaction (see Figure 5.3).

The value of the cutoff κ was determined by comparing the magnitude of estimated rates and finding a gap, see Figure 5.4. The hyperparameter pair (α, λ) was obtained by a grid search and evaluating the difference $\|\hat{\Xi}_{\alpha,\lambda} - \Xi\|_1$, where $\hat{\Xi}_{\alpha,\lambda}$ is the estimated model under a particular hyperparameter choice and Ξ is the ground truth. If the ground truth is unknown, a hyperparameter pair can be estimated by utilizing cross-validation as in the following sections.

5.3.1.2 Data with stochastic noise

In contrast to Section 5.3.1.1, we now employ data that includes measurement noise. Such noise can originate from uncertainties in the experimental setup or from shot noise in single- or few-molecule measurements. In gene-regulatory networks such noise is commonly observed when only few copy numbers of mRNA are present [290–292]. In order to simulate noise from few copies of molecules, the system of Section 5.3.1 with initial conditions as given in Table 5.1a is integrated using the Gillespie stochastic simulation algorithm (SSA) [110, 111], see also Section 2.3.2. In the limit of many particles and realizations, the Gillespie SSA converges to the integrated RRE subject to the LMA. As our model is based on exactly these dynamics, the initial condition's concentrations are interpreted in terms of hundreds of particles. Each realization is then transformed back to

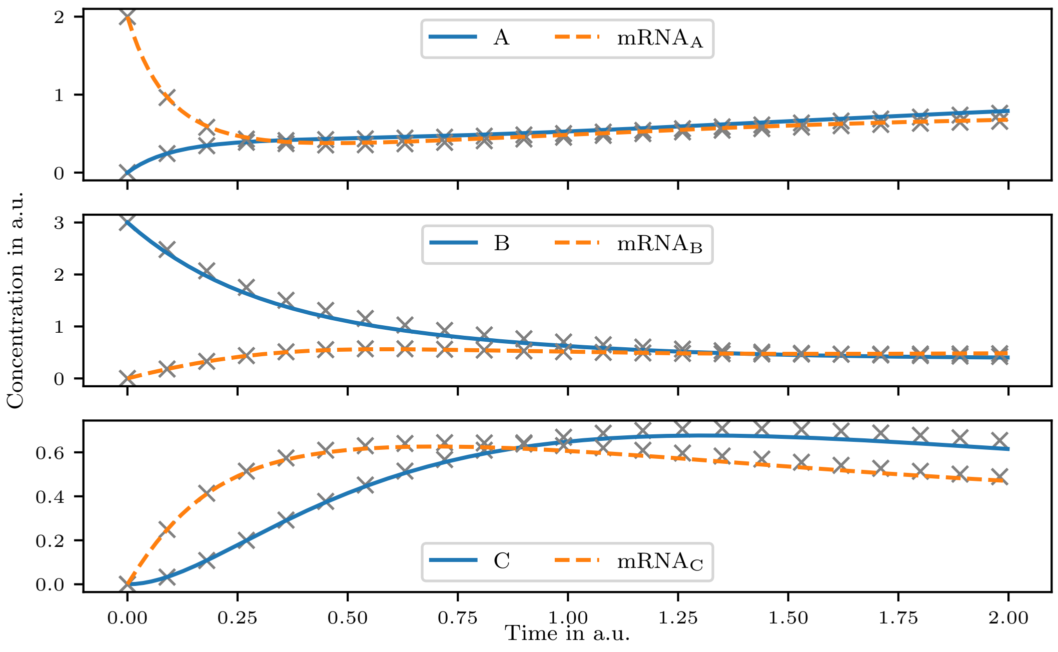


Figure 5.2: **Gene-regulatory network concentration time series.** Concentration time series generated from integrating the reaction network shown in Figure 5.1a. The initial condition prescribes positive concentration values only for B protein and $mRNA_A$ species (see Table 5.1a). This initial condition is used in the subsequent sections for further analysis. Gray dots depict concentration time series yielded from the least squares (LSQ) rates estimated in Section 5.3.1.1.

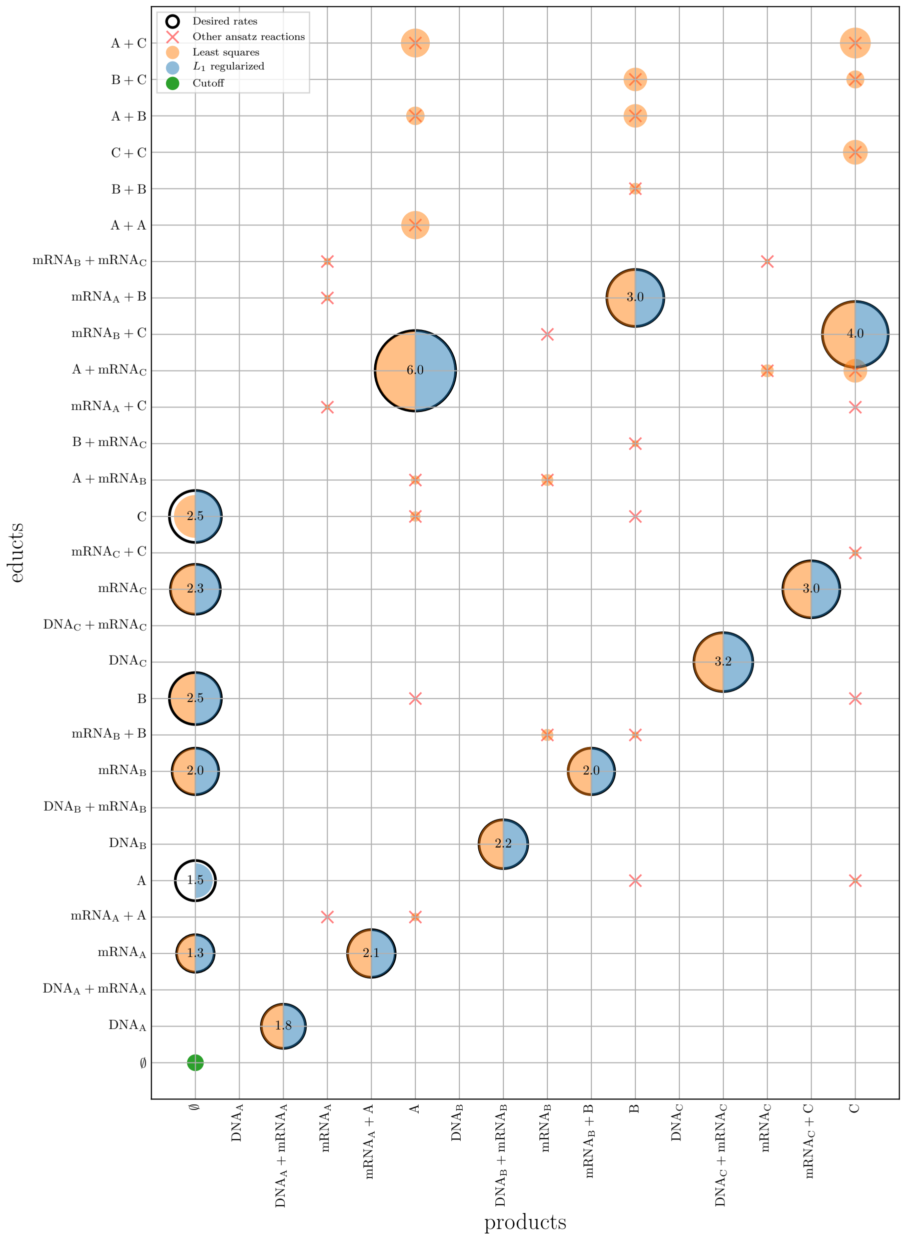


Figure 5.3: **Reaction rate sparsity pattern.** Estimated reaction rates in the system described in Section 5.3.1.1. The y and x axes contain reaction educts and products, respectively. A circle at position (i, j) represents a reaction $i \rightharpoonup j$ whose rate is proportional to the area of the circle. The black outlines denote the reactions with which the system was generated and contain the respective rate value. Red crosses denote reactions that were used as additional ansatz reactions. Blue circles are estimated by least squares and orange circles depict rates which were obtained by solving the minimization problem (5.6). The latter rates are subject to a cutoff $\kappa = 0.22$ corresponding to the green circle's area under which a sparse solution with the correct processes can be recovered. If a certain rate was estimated in both cases, two wedges instead of one circle are displayed.

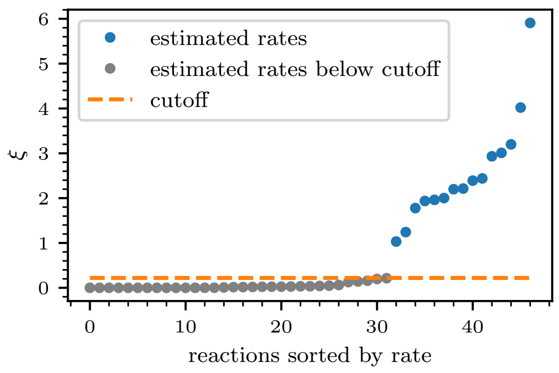


Figure 5.4: **Reaction rate cutoff.** Reaction rates sorted by their magnitude to determine the cutoff $\kappa = 0.22$ of Section 5.3.1.1. The rates were estimated using the regularized minimization problem (5.6).

a time series of concentrations. We define the noise level as the mean-squared deviation of the concentration time series from the integrated reaction-rate equations. Data with different noise levels are prepared by averaging multiple realizations of the time series obtained by the Gillespie SSA.

It can be observed that decreasing levels of noise lead to fewer spurious reactions when applying `reactive SINDy` (5.6), see Figure 5.5a. Also the estimation error $\|\xi - \hat{\xi}\|_1$ with respect to the ground truth ξ decreases with decreasing levels of noise (see Figure 5.5b). In both cases, the regularized method with a suitable hyperparameter pair (α, λ) performs better than LSQ.

The hyperparameters (α, λ) are obtained by shuffling the data and performing a 10-fold cross validation.

5.3.1.3 Multiple initial conditions

Preparing the experiment that generates the data in different initial conditions can help identifying the true reaction mechanisms as a more diverse dataset makes it easier to confirm or exclude the participation of specific reactions. This section extends the analysis of the previous Section 5.3.1.2 to two initial conditions, where the first initial condition is identical to the one used previously and the second initial condition is given in Table 5.1b.

The corresponding time series are depicted in Fig. 5.6a. The gray graph corresponds to a sample trajectory generated by the Gillespie SSA. For both initial conditions the same time step of $\tau = 3 \cdot 10^{-3}$ has been applied, amounting to $2 \cdot 667 = 1334$ frames. Once the data matrices

$$\mathbf{X}_1 = \begin{pmatrix} \boldsymbol{x}_1(t_1) & \cdots & \boldsymbol{x}_1(t_{667}) \end{pmatrix}, \; \mathbf{X}_2 = \begin{pmatrix} \boldsymbol{x}_2(t_1) & \cdots & \boldsymbol{x}_2(t_{667}) \end{pmatrix}$$

and the corresponding derivatives $\dot{\mathbf{X}}_1$, $\dot{\mathbf{X}}_2$ have been obtained, the frames are concatenated so that

$$\mathbf{X} = \begin{pmatrix} \boldsymbol{x}_1(t_1) & \cdots & \boldsymbol{x}_1(t_{667}) & \boldsymbol{x}_2(t_1) & \cdots & \boldsymbol{x}_2(t_{667}) \end{pmatrix},$$

analogously for $\dot{\mathbf{X}}$.

Similarly to Section 5.3.1.2, decreasing levels of noise lead to fewer spurious reactions (see Figure 5.6b) and a smaller L_1 distance to the ground truth (Figure 5.6c). Again applying the

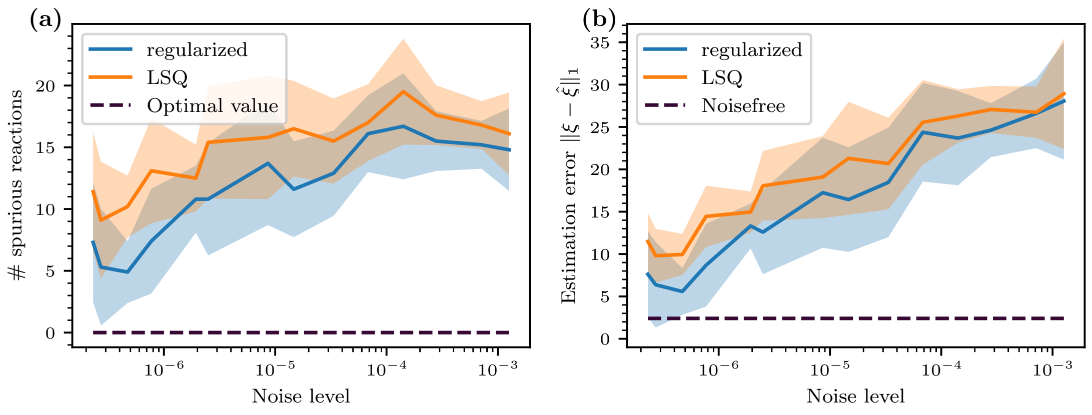


Figure 5.5: **Convergence to ground truth under decreasing noise levels.** Convergence of the estimation error when estimating the system described in Section 5.3.1.1 with varying levels of noise by application of `reactive SINDy` (5.6) with and without regularization in blue and orange, respectively. The procedure was independently repeated 10 times with different realizations giving rise to the mean and standard deviation depicted by solid lines and shaded areas, respectively. **(a)** The number of detected spurious reactions up to the cutoff value introduced in Section 5.3.1.1 over different levels of noise. **(b)** The estimation error given by the mean absolute error between the generating reaction rates ξ and the estimated reaction rates $\hat{\xi}$ over different levels of noise.

optimization problem with a suitable set of parameters $(\alpha, \lambda, \kappa)$ performs better than LSQ. Compared to the previous section the convergence is better due to twice as much available data. At noise levels of smaller than roughly 10^{-6} the model can reliably be recovered when using the regularized method.

The hyperparameters (α, λ) are obtained by shuffling the data and performing a 20-fold cross validation.

5.3.2 Application to the MAPK cascade

The `reactive SINDy` method is applied to the mitogen activated protein kinases (MAPK) pathway [11] which is an important regulatory mechanism of biological cells to respond to stimuli and is involved in proliferation, differentiation, inflammation, and apoptosis [12]. Single-cell MAPK kinetics can be observed experimentally [14]. Mathematically MAPK kinetics are often modelled using reaction rate equations [293, 294] which enables analysis using `reactive SINDy`.

Generally a MAPK pathway consists of multiple stages of kinases that are either inactive or active, denoted by "*". Their activation occurs due to phosphorylation catalyzed by the upstream kinase of the previous stage, dephosphorylation is catalyzed by phosphatases. When the kinase is active it can activate other downstream kinases of the next stage. The initial activation is often due to an external stimulus. The response of the whole cascade is the amount of activated substrate after the final stage, typically measured as a function of the initial stimulus.

Here the MAPK pathway is modeled with three stages of kinases MAPK, MAPKK, and MAPKKK. The initial stimulus is called S and the final substrate to be activated is a transcription

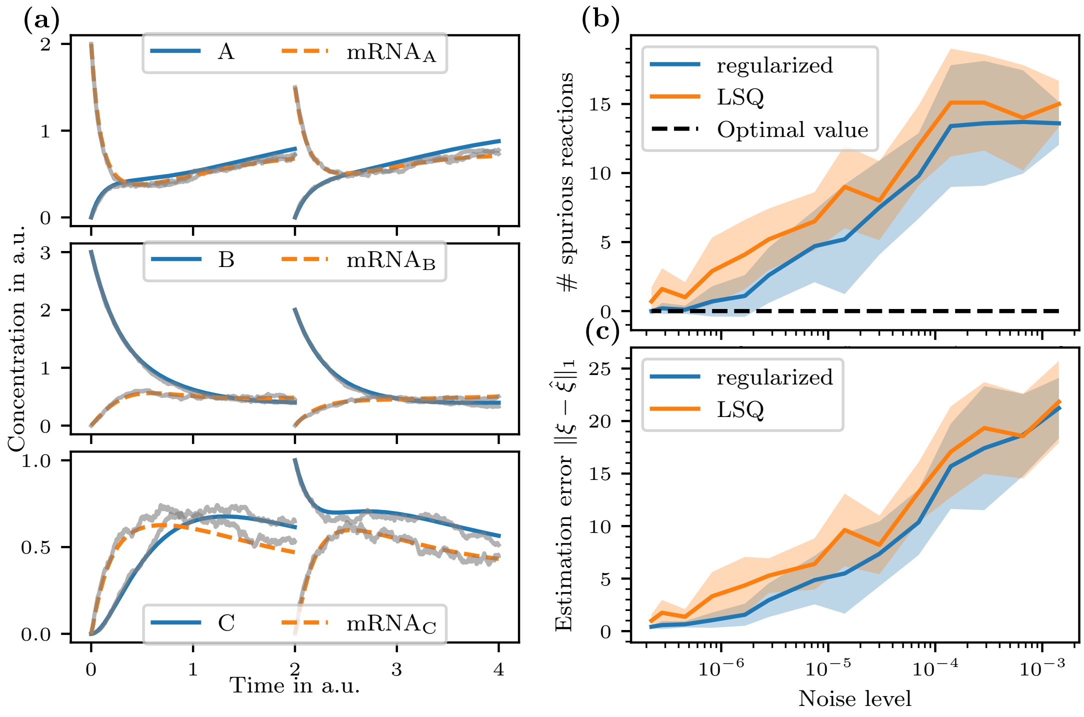


Figure 5.6: **Convergence to ground truth with two different initial conditions.** Convergence of estimation error of reaction schemes from noisy gene-regulation data starting from two different initial conditions under decreasing levels of noise. The minimization problem (5.6) was solved for $\alpha = 0$ (LSQ) and with regularization. This was repeated 10 times on different sets of observation data generated by Gillespie SSA, giving rise to mean and standard deviation (solid lines and shaded areas, respectively). **(a)** Concentration time series corresponding to the initial conditions, generated by integrating the reaction-rate equations. The first initial condition is identical to the one used in Section 5.3.1.1 and Section 5.3.1.2. The second initial condition (Table 5.1b) prescribes positive initial concentrations for mRNA_A, B, and C species. The gray graphs are sample realizations of integration using the Gillespie SSA. **(b)-(c)** Analogously to Figure 5.5 with the difference that 20-fold cross validation was used for hyperparameter estimation.

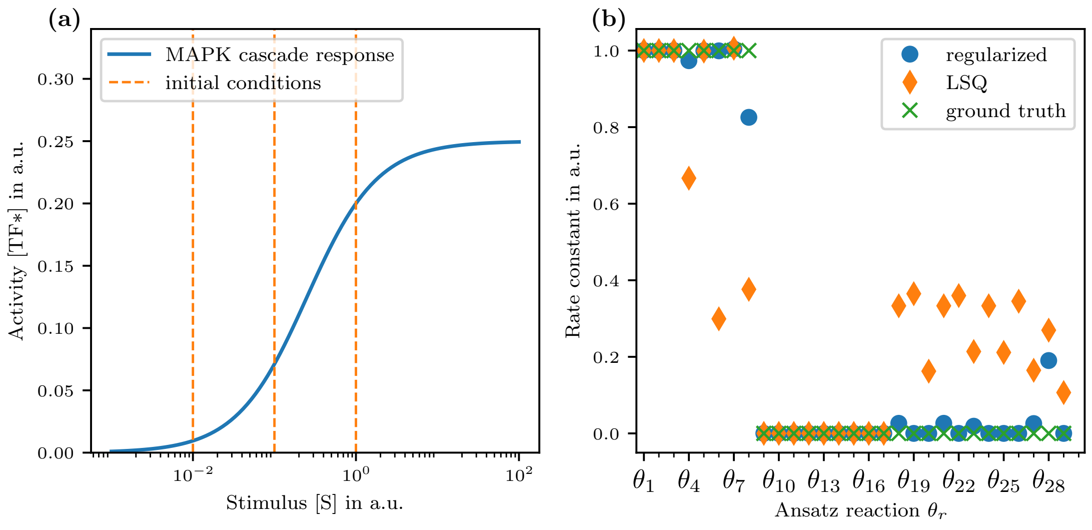


Figure 5.7: **Reactive SINDy on MAPK.** Application of `reactive SINDy` to the mitogen activated protein kinases (MAPK) pathway system. **(a)** The response curve of the MAPK cascade as a function of external stimulus given as a constant concentration [S]. The activity is the steady state concentration of activated transcription factors [TF∗]. Dashed lines show the values of [S] at which concentration time series data was generated. **(b)** Estimated rate coefficients of candidate reactions (see Table B.2) after application of `reactive SINDy` (regularized) to the time series data. LSQ and the ground truth model for comparison.

factor TF. The ground truth reaction network consists of activation/phosphorylation reactions

$$\begin{aligned}
\mathrm{S} + \mathrm{MAPKKK} &\rightharpoonup \mathrm{S} + \mathrm{MAPKKK}{*}\\
\mathrm{MAPKKK}{*} + \mathrm{MAPKK} &\rightharpoonup \mathrm{MAPKKK}{*} + \mathrm{MAPKK}{*}\\
\mathrm{MAPKK}{*} + \mathrm{MAPK} &\rightharpoonup \mathrm{MAPKK}{*} + \mathrm{MAPK}{*}\\
\mathrm{MAPK}{*} + \mathrm{TF} &\rightharpoonup \mathrm{MAPK}{*} + \mathrm{TF}{*}
\end{aligned}$$

and deactivation/dephosphorylation reactions

$$\begin{aligned}
\mathrm{MAPKKK}{*} &\rightharpoonup \mathrm{MAPKKK}\\
\mathrm{MAPKK}{*} &\rightharpoonup \mathrm{MAPKK}\\
\mathrm{MAPK}{*} &\rightharpoonup \mathrm{MAPK}\\
\mathrm{TF}{*} &\rightharpoonup \mathrm{TF}.
\end{aligned}$$

For simplicity we assume phosphatase to be abundant such that deactivations effectively become first order reactions. The external stimulus S is not consumed such that time integration of these reactions yields a steady state in which the response, i.e., the concentration [TF∗] can be measured as a function of the stimulus concentration [S].

Using the rate constants given in Table B.2 we obtain the response curve given in Figure 5.7a.

We generate concentration time series data of the MAPK reactions above at three different initial conditions, each differing in the amount of stimulus [S]. The response yielded by the chosen initial conditions is marked in Figure 5.7a by vertical dashed lines. The concatenated time series is a dataset of 300 frames in total. We use the library Θ of ansatz reactions given in Table B.2. The hyperparameter $\alpha = 6.6 \times 10^{-9}$ was determined by shuffling the data and performing 15-fold cross validation. The estimated rate constants were obtained by solving the minimization problem (5.6) with $\lambda = 1$. The results are given in Figure 5.7b. Least-squares estimation detects 5 of the 8 reaction processes that belong to the ground truth model. However it also detects 12 spurious reaction processes (θ_{18} - θ_{29}). `Reactive SINDy` estimation detects all reactions of the ground truth, two processes (θ_4 and θ_8) show deviations in rate constants. Generally `reactive SINDy` yields a sparse model which allows further simplification of the reaction network by dropping out reaction processes that lie beneath a certain cutoff. In this case for example a cutoff of $\kappa = 0.25$ would directly recover the ground truth reaction network. Quantitatively, one may consider the L_1 norm of the relative distance of estimated rate constants $\hat{\xi}_r$ to the non-zero rate constants of the ground truth ξ_r

$$\sum_{r=1}^{8} \left| (\hat{\xi}_r - \xi_r)/\xi_r \right|$$

which yields 167% error for least-squares and 21% error for `reactive SINDy`.

5.3.3 Application to the Lotka–Volterra system

As biological pathways often exhibit oscillatory behavior [295] which can stem from positive or negative feedback loops [296] we apply `reactive SINDy` to an idealized oscillatory system, namely the Lotka–Volterra system that was already used in examples in Sections 2.3.1–2.3.2.

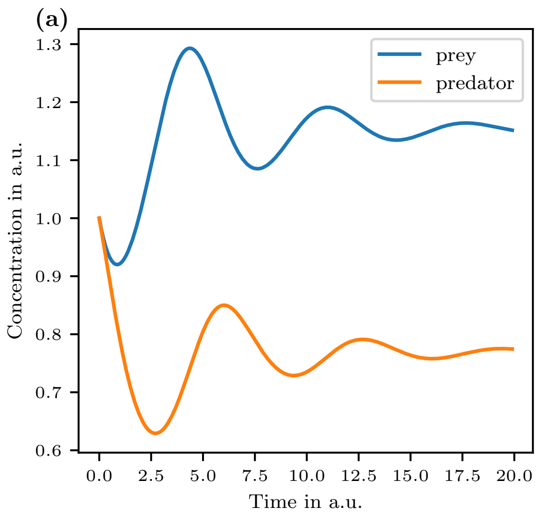
The predator-prey dynamics of two species A (prey) and B (predator) is defined by the reaction network given in Equation (2.29). From this model we generated concentration time series data with 200 frames which is displayed in Figure 5.8a. The library of ansatz reactions Θ is given in Table B.3. The hyperparameter $\alpha = 2.7 \times 10^{-7}$ was determined by shuffling the data and performing 5-fold cross validation. The estimated rate constants were obtained by solving the minimization problem (5.6) with $\lambda = 1$. The results are depicted in Figure 5.8b. Least-squares estimation detects all reactions of the ground truth model but also two spurious processes (θ_6 and θ_7) with a higher rate than the first two underlying processes (θ_1 and θ_2). `Reactive SINDy` recovers the true reaction network with minor deviations in rate constants. As in Section 5.3.2, considering the L_1 norm of the relative distance to the ground truth for non-zero rate constants

$$\sum_{r=1}^{5} \left| (\hat{\xi}_r - \xi_r)/\xi_r \right|$$

yields 75% error for least-squares and 7% error for `reactive SINDy`.

5.4 Conclusions

In this chapter we have extended the SINDy method to `reactive SINDy`, not only parsimoniously detecting potentially nonlinear terms in a dynamical system from noisy data, but also yielding, in this case, a sparse set of rates with respect to generating reactions (5.4). Mathematically this has been achieved by permitting vector-valued basis functions and obtaining a tensor linear regression problem. We have applied this method on data generated from a gene-regulation network, a MAPK pathway, and a Lotka–Volterra system and could successfully recover the underlying reaction networks.



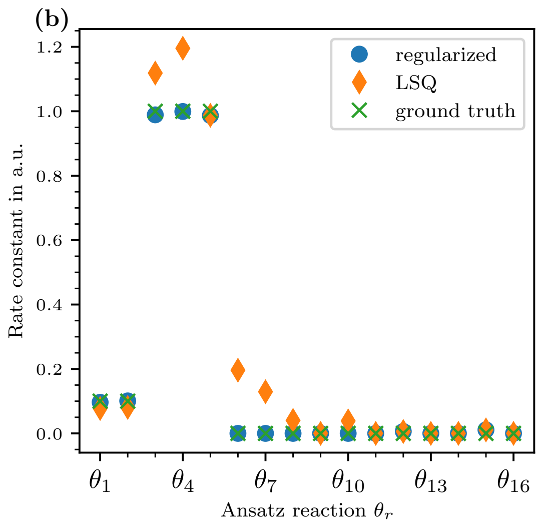


Figure 5.8: **Reactive SINDy on Lotka–Volterra.** Application of reactive SINDy to the Lotka–Volterra system with social friction. **(a)** Concentration data as a function of time for predator and prey species. **(b)** Estimated rate coefficients of candidate reactions (see Table B.3) after application of `reactive SINDy` (regularized) to the time series data. Least squares (LSQ) estimation and the ground truth model for comparison.

The studies of Section 5.3.1.2 and Section 5.3.1.3 have shown that the applied regularization terms can mitigate noise up to a certain degree compared to the unregularized method, so that identification of the reaction network is more robust and closer to the ground truth. Potentially, this method could be used to identify reaction networks from time series measurements even if the initial conditions are not always exactly identical, as was demonstrated in Section 5.3.1.3.

One apparent limitation is that the method can only be applied if the data stems from the equilibration phase, as the concentration-based approach has derivatives equal zero in the equilibrium, which precludes the reaction dynamics to be recovered. Thus, in the case of oscillatory systems the reaction network can be recovered robustly.

Chapter 6

Summary and outlook

We presented and implemented methods for the efficient generation and analysis of time series data for many-body systems, focusing mainly on biologically motivated applications. Often these systems, e.g., in the study of biomolecular processes, are highly complex and difficult to observe experimentally with a high spatiotemporal resolution. Computer simulations, however, can attain a high level of resolution within reach of the underlying model, as is the case in molecular dynamics. The high resolution comes with the caveat of large amounts of data, i.e., long trajectories in a high-dimensional space, rendering inspection and analysis of the dominant processes a challenging task.

In the first part of the thesis (Chapter 3) we presented several machine learning approaches that aim to estimate reduced dynamical models from long and high-dimensional trajectories. These dynamical models can reflect governing dynamics, dominant processes, (meta-)stable structures, and structures that are geometrically stable over time. We developed the novel machine learning library `deeptime` with the goals to make community-specific methods accessible to a broad user base by implementing them in a general-purpose way, and to provide a library that is easy to install, extend, and maintain. These goals were achieved: Alongside a high degree of modularization and few hard dependencies in the implementation, we further offer a comprehensive documentation and a wide range of mostly domain unspecific example datasets. To get a unified view, we compared a majority of the implemented methods from a theoretical point of view—in particular reviewing connections between methods—and conducted numerical studies on model systems. These studies mainly deal with the projection of time series data onto dominant processes and coherent set detection tasks. The studies are set up so that they can serve as benchmarks for future additions to the library and/or development of new methods. For coherent set detection, we defined a novel coherence score to quantitatively determine the estimation quality, which allowed us to draw comparisons and rank methods. Although most implemented methods focus on finding finite-dimensional approximations of transfer operators, `deeptime` complements these by including the sparse identification of nonlinear dynamics (SINDy) method, which approximates the underlying governing dynamics.

In the future, it would be desirable to develop methods which are capable of estimating dynamical models with fluctuating numbers of particles. That way, the full flexibility and information content of particle-based reaction dynamics simulations could be leveraged. In terms of implemented methods there are various possible directions: transition manifold learning methods, more SINDy variants, different flavors of dynamic mode decomposition, more basis transformations, or kernel implementations. Support for other deep learning frameworks (e.g., JAX) could be added—either directly or as a separate plugin library. From a more programmatic point of view, the addition of built-in support for streaming large amounts of data not completely fitting into memory is beneficial. While it is currently possible for users to stream the data themselves and use it on estimators which are capable of being partially fit, it would greatly improve the convenience if such implementations were already available. Furthermore, `deeptime` would benefit from more example datasets as well as tutorial notebooks.

In the second part of the thesis (Chapter 4), we developed and introduced the novel interacting-particle reaction dynamics software `ReaDDy2` for the simulation of biological macromolecules' bulk behavior. The package approaches the problem of particle simulators being bounded by computational effort, meaning that there is a trade-off between underlying model resolution and accessible sizes of systems and timescales. `ReaDDy2` operates on a mesoscopic scale where, e.g., proteins are treated as individual particles or small particle complexes. At its resolution level, we showed that it is significantly faster than comparable simulation packages. It is suitable to simulate biological processes—like signal transduction in crowded cellular environments or chemical nanoreactors with complex geometries—involving $\sim 5 \cdot 10^4$ particles on the millisecond to second timescale while using integration time steps in the range of nanoseconds. These timescales are on current hardware beyond the reach of more fine-grained approaches such as molecular dynamics. We successfully verified that `ReaDDy2` simulations yield quantitatively correct results for a set of numerical experiments. By a novel generalization of chemical reactions to reactions on a particle topology level, we enabled even more complex models that are capable of simulating, e.g., the dynamical growth of geometrically complex structures such as actin filaments. Through the successful implementation of `ReaDDy2`, we created new possibilities to further our insight into biological processes.

Future developments could introduce anisotropic interactions as well as particles with orientation. Currently, anisotropy and orientation can only be emulated by using connected particle complexes. `ReaDDy2` would also benefit from the addition of coarse-grained particle-based membrane models. Support for specialized hardware like graphics cards or distributed environments like supercomputers could be added to enable the study of larger systems and/or longer timescales.

In the third part of the thesis (Chapter 5), we presented `reactive SINDy`. It is an extension of the SINDy method and can parsimoniously detect a network of reactions in terms of reaction rate equations based on concentration time series data of species. A crucial ingredient for the method are ansatz functions which are coupled over a system of ordinary differential equations, since we aim to recover reaction rates as well as stochiometry. To this end we formulate `reactive SINDy` as a tensor linear regression problem and successfully demonstrate its abilities on model systems.

As a future direction, the method could be phrased with respect to the chemical master equation instead of reaction rate equations. Currently, `reactive SINDy` functions best when applied to off-equilibrium data, since equilibrated concentrations yield no more information with respect to their time derivative. An reaction rate equation formulation potentially allows the inference of governing reactions based on stochastic fluctuations even in case of steady state data.

In this thesis, we developed general methods and software packages aiding both in the simulation and analysis of many-body systems. Although we focused mostly on biological processes, these tools can be applied to a wider range of problems. For instance, the introduced machine learning tools are also ideal to analyze data from fluid dynamics, both from simulation data as well as from real measurements, such as atmospheric flow data. Moreover, although `ReaDDy2` was developed for modeling reaction kinetics on a mesoscopic scale, it could also be used to study topics ranging from epidemics to population dynamics, pattern formation, or soft matter physics. By the successful formulation and implementation of these methods, we thus provide accessible theoretical and computational platforms to study a wide range of many-body systems.

Bibliography

[P1] M. Hoffmann, C. Fröhner, and F. Noé. “ReaDDy 2: Fast and flexible software framework for interacting-particle reaction dynamics”. In: *PLOS Computational Biology* 15.2 (Feb. 28, 2019). Ed. by Q. Cui, e1006830. ISSN: 1553-7358. DOI: `10.1371/journal.pcbi.1006830` (cited on pp. 1, 6, 10, 59, 82).

[P2] M. Hoffmann, C. Fröhner, and F. Noé. “Reactive SINDy: Discovering governing reactions from concentration data”. In: *The Journal of chemical physics* 150.2 (2019), p. 025101. DOI: `10.1063/1.5066099` (cited on pp. 1, 6, 81).

[P3] M. Hoffmann and F. Noé. “Generating valid Euclidean distance matrices”. In: *arXiv preprint arXiv:1910.03131* (2019). DOI: `10.48550/arXiv.1910.03131` (cited on p. 1).

[P4] M. Hoffmann et al. “Deeptime: a Python library for machine learning dynamical models from time series data”. In: *Machine Learning: Science and Technology* 3.1 (Dec. 2021), p. 015009. DOI: `10.1088/2632-2153/ac3de0` (cited on pp. 1, 6, 16, 17, 19, 29).

[P5] M. K. Scherer, B. Trendelkamp-Schroer, F. Paul, G. Pérez-Hernández, M. Hoffmann, N. Plattner, C. Wehmeyer, J.-H. Prinz, and F. Noé. “PyEMMA 2: A Software Package for Estimation, Validation, and Analysis of Markov Models”. In: *Journal of Chemical Theory and Computation* 11.11 (Nov. 10, 2015), pp. 5525–5542. DOI: `10.1021/acs.jctc.5b00743` (cited on pp. 1, 8, 31, 33).

[P6] M. K. Scherer, B. E. Husic, M. Hoffmann, F. Paul, H. Wu, and F. Noé. “Variational selection of features for molecular kinetics”. In: *The Journal of chemical physics* 150.19 (2019), p. 194108. DOI: `10.1063/1.5083040` (cited on pp. 1, 37).

[P7] K. Shmilovich, M. Stieffenhofer, N. E. Charron, and M. Hoffmann. “Temporally coherent backmapping of molecular trajectories from coarse-grained to atomistic resolution”. In: *arXiv preprint arXiv:2205.05213* (2022). DOI: `10.48550/arXiv.2205.05213` (cited on p. 1).

[1] M. J. Rust, M. Bates, and X. Zhuang. “Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM)”. In: *Nature Methods* 3.10 (Oct. 2006), pp. 793–796. ISSN: 1548-7091. DOI: `10.1038/nmeth929` (cited on p. 4).

[2] M. Adrian, J. Dubochet, J. Lepault, and A. W. McDowall. “Cryo-electron microscopy of viruses”. In: *Nature* 308.5954 (1984), pp. 32–36. DOI: `10.1038/308032a0` (cited on p. 4).

[3] J.-P. Renaud, A. Chari, C. Ciferri, W.-t. Liu, H.-W. Rémigy, H. Stark, and C. Wiesmann. “Cryo-EM in drug discovery: achievements, limitations and prospects”. In: *Nature Reviews Drug Discovery* 17.7 (July 2018), pp. 471–492. ISSN: 1474-1776. DOI: `10.1038/nrd.2018.77` (cited on p. 4).

[4] J. C. Kendrew, G. Bodo, H. M. Dintzis, R. G. Parrish, H. Wyckoff, and D. C. Phillips. “A three-dimensional model of the myoglobin molecule obtained by x-ray analysis”. In: *Nature* 181.4610 (1958), pp. 662–666. DOI: 10.1038/181662a0 (cited on p. 4).

[5] K. Takahashi, S. Tănase-Nicola, and P. R. ten Wolde. “Spatio-temporal correlations can drastically change the response of a MAPK pathway”. In: *Proceedings of the National Academy of Sciences* 107.6 (Feb. 9, 2010), pp. 2473–2478. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0906885107 (cited on p. 4).

[6] M. Gunkel, J. Schöneberg, W. Alkhaldi, S. Irsen, F. Noé, U. B. Kaupp, and A. Al-Amoudi. “Higher-order architecture of rhodopsin in intact photoreceptors and its implication for phototransduction kinetics”. In: *Structure* 23.4 (2015), pp. 628–638. DOI: 10.1016/j.str.2015.01.015 (cited on pp. 4, 60).

[7] E. Betzig, G. H. Patterson, R. Sougrat, O. W. Lindwasser, S. Olenych, J. S. Bonifacino, M. W. Davidson, J. Lippincott-Schwartz, and H. F. Hess. “Imaging intracellular fluorescent proteins at nanometer resolution”. In: *science* 313.5793 (2006), pp. 1642–1645. DOI: 10.1126/science.1127344 (cited on p. 4).

[8] S. T. Hess, T. P. Girirajan, and M. D. Mason. “Ultra-high resolution imaging by fluorescence photoactivation localization microscopy”. In: *Biophysical journal* 91.11 (2006), pp. 4258–4272. DOI: 10.1529/biophysj.106.091116 (cited on p. 4).

[9] B.-C. Chen et al. “Lattice light-sheet microscopy: Imaging molecules to embryos at high spatiotemporal resolution”. In: *Science* 346.6208 (2014), p. 1257998. DOI: 10.1126/science.1257998 (cited on p. 4).

[10] J. M. Holton. “A beginner’s guide to radiation damage”. In: *Journal of Synchrotron Radiation* 16.2 (Mar. 1, 2009), pp. 133–142. ISSN: 0909-0495. DOI: 10.1107/S0909049509004361 (cited on p. 4).

[11] J. Xing, D. D. Ginty, and M. E. Greenberg. “Coupling of the RAS-MAPK pathway to gene activation by RSK2, a growth factor-regulated CREB kinase”. In: *Science* 273.5277 (1996), pp. 959–963. DOI: 10.1126/science.273.5277.959 (cited on pp. 4, 90).

[12] W. Zhang and H. T. Liu. “MAPK signal pathways in the regulation of cell proliferation in mammalian cells”. In: *Cell Research* 12.1 (Mar. 2002), pp. 9–18. ISSN: 1001-0602. DOI: 10.1038/sj.cr.7290105 (cited on pp. 4, 90).

[13] L. Chang and M. Karin. “Mammalian MAP kinase signalling cascades”. In: *Nature* 410.6824 (Mar. 2001), pp. 37–40. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/35065000 (cited on p. 4).

[14] H. Ryu, M. Chung, J. Song, S. S. Lee, O. Pertz, and N. L. Jeon. “Integrated Platform for Monitoring Single-cell MAPK Kinetics in Computer-controlled Temporal Stimulations”. In: *Scientific Reports* 8.1 (Dec. 2018), p. 11126. ISSN: 2045-2322. DOI: 10.1038/s41598-018-28873-1 (cited on pp. 4, 90).

[15] B. Trzaskowski, D. Latek, S. Yuan, U. Ghoshdastider, A. Debinski, and S. Filipek. “Action of Molecular Switches in GPCRs - Theoretical and Experimental Studies”. In: *Current Medicinal Chemistry* 19.8 (Mar. 1, 2012), pp. 1090–1109. ISSN: 09298673. DOI: 10.2174/092986712799320556 (cited on p. 4).

[16] J. A. Beavo and L. L. Brunton. “Cyclic nucleotide research – still expanding after half a century”. In: *Nature Reviews Molecular Cell Biology* 3.9 (Sept. 2002), pp. 710–717. ISSN: 1471-0072, 1471-0080. DOI: 10.1038/nrm911 (cited on p. 4).

[17] R. Abramovitch, E. Tavor, J. Jacob-Hirsch, E. Zeira, N. Amariglio, O. Pappo, G. Rechavi, E. Galun, and A. Honigman. “A pivotal role of cyclic AMP-responsive element binding protein in tumor progression”. In: *Cancer research* 64.4 (2004), pp. 1338–1346. DOI: 10.1158/0008-5472.can-03-2089 (cited on pp. 4, 81).

[18] D. Dragun, A. Philippe, R. Catar, and B. Hegner. “Autoimmune mediated G-protein receptor activation in cardiovascular and renal pathologies”. In: *Thrombosis and haemostasis* 101.04 (2009), pp. 643–648. DOI: 10.1160/TH08-10-0710 (cited on p. 4).

[19] A. Thathiah and B. De Strooper. “The role of G protein-coupled receptors in the pathology of Alzheimer’s disease”. In: *Nature Reviews Neuroscience* 12.2 (2011), pp. 73–87. DOI: 10.1038/nrn2977 (cited on p. 4).

[20] S. R. Agarwal, C. E. Clancy, and R. D. Harvey. “Mechanisms restricting diffusion of intracellular cAMP”. In: *Scientific reports* 6.1 (2016), pp. 1–11. DOI: 10.1038/srep19577 (cited on p. 4).

[21] M. D. Houslay. “Underpinning compartmentalised cAMP signalling through targeted cAMP breakdown”. In: *Trends in biochemical sciences* 35.2 (2010), pp. 91–100. DOI: 10.1016/j.tibs.2009.09.007 (cited on p. 4).

[22] J. Schöneberg, M. Heck, K. P. Hofmann, and F. Noé. “Explicit Spatiotemporal Simulation of Receptor-G Protein Coupling in Rod Cell Disk Membranes”. In: *Biophysical Journal* 107.5 (Sept. 2014), pp. 1042–1053. ISSN: 00063495. DOI: 10.1016/j.bpj.2014.05.050 (cited on pp. 4, 60, 82).

[23] K. Raghavachari. “Perspective on “Density functional thermochemistry. III. The role of exact exchange””. In: *Theoretical Chemistry Accounts* 103.3 (2000), pp. 361–363. DOI: 10.1007/978-3-662-10421-7_60 (cited on p. 5).

[24] J. Tirado-Rives and W. L. Jorgensen. “Performance of B3LYP density functional methods for a large set of organic molecules”. In: *Journal of chemical theory and computation* 4.2 (2008), pp. 297–306. DOI: 10.1021/ct700248k (cited on p. 5).

[25] N. Schuch and F. Verstraete. “Computational complexity of interacting electrons and fundamental limitations of density functional theory”. In: *Nature Physics* 5.10 (Oct. 2009), pp. 732–735. ISSN: 1745-2473, 1745-2481. DOI: 10.1038/nphys1370 (cited on p. 5).

[26] J. B. Klauda, R. M. Venable, J. A. Freites, J. W. O’Connor, D. J. Tobias, C. Mondragon-Ramirez, I. Vorobyov, A. D. MacKerell, and R. W. Pastor. “Update of the CHARMM All-Atom Additive Force Field for Lipids: Validation on Six Lipid Types”. In: *J. Phys. Chem. B* 114.23 (June 2010), pp. 7830–7843. ISSN: 1520-6106. DOI: 10.1021/jp101759q (cited on p. 5).

[27] J. A. Maier, C. Martinez, K. Kasavajhala, L. Wickstrom, K. E. Hauser, and C. Simmerling. “ff14SB: Improving the Accuracy of Protein Side Chain and Backbone Parameters from ff99SB”. In: *J. Chem. Theory Comput.* 11.8 (Aug. 2015), pp. 3696–3713. ISSN: 1549-9618, 1549-9626. DOI: 10.1021/acs.jctc.5b00255 (cited on p. 5).

[28] C. Tian et al. “ff19SB: Amino-Acid-Specific Protein Backbone Parameters Trained against Quantum Mechanics Energy Surfaces in Solution”. In: *Journal of Chemical Theory and Computation* 16.1 (Jan. 14, 2020), pp. 528–552. ISSN: 1549-9618, 1549-9626. DOI: 10.1021/acs.jctc.9b00591 (cited on p. 5).

[29] K. Vanommeslaeghe et al. “CHARMM general force field: A force field for drug-like molecules compatible with the CHARMM all-atom additive biological force fields”. In: *Journal of Computational Chemistry* (2009), NA–NA. ISSN: 01928651, 1096987X. DOI: 10.1002/jcc.21367 (cited on p. 5).

[30] J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, and D. A. Case. “Development and testing of a general amber force field”. In: *Journal of Computational Chemistry* 25.9 (July 15, 2004), pp. 1157–1174. ISSN: 0192-8651, 1096-987X. DOI: `10.1002/jcc.20035` (cited on p. 5).

[31] T. Darden, D. York, and L. Pedersen. “Particle mesh Ewald: An N log(N) method for Ewald sums in large systems”. In: *The Journal of chemical physics* 98.12 (1993), pp. 10089–10092. DOI: `10.1063/1.464397` (cited on p. 5).

[32] A. Warshel and M. Levitt. “Theoretical studies of enzymic reactions: dielectric, electrostatic and steric stabilization of the carbonium ion in the reaction of lysozyme”. In: *Journal of molecular biology* 103.2 (1976), pp. 227–249. DOI: `10.1016/0022-2836(76)90311-9` (cited on p. 5).

[33] G. R. Bowman, V. S. Pande, and F. Noé, eds. *An Introduction to Markov State Models and Their Application to Long Timescale Molecular Simulation.* Vol. 797. Advances in Experimental Medicine and Biology. Dordrecht: Springer Netherlands, 2014. ISBN: 978-94-007-7605-0. DOI: `10.1007/978-94-007-7606-7` (cited on p. 5).

[34] B. Leimkuhler and C. Matthews. *Molecular Dynamics.* Vol. 39. Interdisciplinary Applied Mathematics. Cham: Springer International Publishing, 2015. ISBN: 978-3-319-16375-8. DOI: `10.1007/978-3-319-16375-8` (cited on p. 5).

[35] N. Plattner and F. Noé. “Protein conformational plasticity and complex ligand-binding kinetics explored by atomistic simulations and Markov models”. In: *Nature Communications* 6.1 (Nov. 2015), p. 7653. ISSN: 2041-1723. DOI: `10.1038/ncomms8653` (cited on p. 5).

[36] F. Paul, C. Wehmeyer, E. T. Abualrous, H. Wu, M. D. Crabtree, J. Schöneberg, J. Clarke, C. Freund, T. R. Weikl, and F. Noé. “Protein-peptide association kinetics beyond the seconds timescale from atomistic simulations”. In: *Nature Communications* 8.1 (Dec. 2017), p. 1095. ISSN: 2041-1723. DOI: `10.1038/s41467-017-01163-6` (cited on p. 5).

[37] D. E. Shaw et al. “Anton 2: Raising the Bar for Performance and Programmability in a Special-Purpose Molecular Dynamics Supercomputer”. In: *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis.* SC14: International Conference for High Performance Computing, Networking, Storage and Analysis. New Orleans, LA: IEEE, Nov. 2014, pp. 41–53. ISBN: 978-1-4799-5500-8. DOI: `10.1109/SC.2014.9` (cited on p. 5).

[38] R. C. Godwin, R. Melvin, and F. R. Salsbury. “Molecular Dynamics Simulations and Computer-Aided Drug Discovery”. In: *Computer-Aided Drug Discovery.* Ed. by W. Zhang. Series Title: Methods in Pharmacology and Toxicology. New York, NY: Springer New York, 2015, pp. 1–30. ISBN: 978-1-4939-3519-2. DOI: `10.1007/7653_2015_41` (cited on p. 5).

[39] P. Eastman et al. “OpenMM 7: Rapid development of high performance algorithms for molecular dynamics”. In: *PLOS Computational Biology* 13.7 (July 26, 2017). Ed. by R. Gentleman, e1005659. ISSN: 1553-7358. DOI: `10.1371/journal.pcbi.1005659` (cited on p. 5).

[40] J. A. Anderson, J. Glaser, and S. C. Glotzer. “HOOMD-blue: A Python package for high-performance molecular dynamics and hard particle Monte Carlo simulations”. In: *Computational Materials Science* 173 (Feb. 2020), p. 109363. ISSN: 09270256. DOI: `10.1016/j.commatsci.2019.109363` (cited on p. 5).

[41] A. P. Thompson et al. “LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales”. In: *Comp. Phys. Comm.* 271 (2022), p. 108171. DOI: `10.1016/j.cpc.2021.108171` (cited on pp. 5, 60).

[42] T. Mitra, S. N. Menon, and S. Sinha. "Emergent memory in cell signaling: Persistent adaptive dynamics in cascades can arise from the diversity of relaxation time-scales". In: *Scientific Reports* 8.1 (Dec. 2018), p. 13230. ISSN: 2045-2322. DOI: 10.1038/s41598-018-31626-9 (cited on p. 5).

[43] B. J. Canagarajah, A. Khokhlatchev, M. H. Cobb, and E. J. Goldsmith. "Activation Mechanism of the MAP Kinase ERK2 by Dual Phosphorylation". In: *Cell* 90.5 (Sept. 1997), pp. 859–869. ISSN: 00928674. DOI: 10.1016/S0092-8674(00)80351-7 (cited on p. 5).

[44] G. Jaeger. "What in the (quantum) world is macroscopic?" In: *American Journal of Physics* 82.9 (Sept. 2014), pp. 896–905. ISSN: 0002-9505, 1943-2909. DOI: 10.1119/1.4878358 (cited on p. 5).

[45] H.-P. Kriegel, P. Kröger, and A. Zimek. "Clustering High-Dimensional Data: A Survey on Subspace Clustering, Pattern-Based Clustering, and Correlation Clustering". In: *ACM Trans. Knowl. Discov. Data* 3.1 (Mar. 2009). ISSN: 1556-4681. DOI: 10.1145/1497577.1497578 (cited on p. 7).

[46] K. Pearson. "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572. DOI: 10.1080/14786440109462720 (cited on pp. 7, 30).

[47] H. Hotelling. "Analysis of a complex of statistical variables into principal components". In: *Journal of educational psychology* 24.6 (1933), p. 417. DOI: 10.1037/h0071325 (cited on pp. 7, 30).

[48] F. R. Bach and M. I. Jordan. "Kernel independent component analysis". In: *Journal of machine learning research* 3.Jul (2002), pp. 1–48. DOI: 10.1162/153244303768966085 (cited on pp. 8, 38).

[49] A. Mardt, L. Pasquali, H. Wu, and F. Noé. "VAMPnets for Deep Learning of Molecular Kinetics". In: *Nat. Commun.* 9.1 (Jan. 2018), pp. 1–11. ISSN: 2041-1723. DOI: 10.1038/s41467-017-02388-1 (cited on pp. 8, 31, 41).

[50] K. A. Beauchamp, G. R. Bowman, T. J. Lane, L. Maibaum, I. S. Haque, and V. S. Pande. "MSMBuilder2: Modeling Conformational Dynamics on the Picosecond to Millisecond Scale". In: *Journal of Chemical Theory and Computation* 7.10 (Oct. 11, 2011), pp. 3412–3419. ISSN: 1549-9618, 1549-9626. DOI: 10.1021/ct200463m (cited on pp. 8, 31).

[51] M. Macklin and M. Müller. "Position based fluids". In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), pp. 1–12. DOI: 10.1145/2461912.2461984 (cited on p. 8).

[52] W. G. Bickley. "LXXIII. The plane jet". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 23.156 (1937), pp. 727–731. DOI: 10.1080/14786443708561847 (cited on pp. 8, 45).

[53] I. I. Rypina, M. G. Brown, F. J. Beron-Vera, H. Koçak, M. J. Olascoaga, and I. A. Udovydchenkov. "On the Lagrangian dynamics of atmospheric zonal jets and the permeability of the stratospheric polar vortex". In: *Journal of the Atmospheric Sciences* 64.10 (2007), pp. 3595–3610. DOI: 10.1175/JAS4036.1 (cited on pp. 8, 45).

[54] S. L. Brunton, J. L. Proctor, and J. N. Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *Proceedings of the National Academy of Sciences* 113.15 (2016), pp. 3932–3937. DOI: 10.1073/pnas.1517384113 (cited on pp. 8, 31, 54, 55).

[55] S. Winkelmann and C. Schütte. *Stochastic Dynamics in Computational Biology*. Frontiers in Applied Dynamical Systems: Reviews and Tutorials. Cham: Springer International Publishing, 2020. ISBN: 978-3-030-62386-9. DOI: 10.1007/978-3-030-62387-6 (cited on pp. 10, 22, 24, 28).

[56] R. Erban and S. J. Chapman. "Stochastic modelling of reaction–diffusion processes: algorithms for bimolecular reactions". In: *Physical Biology* 6.4 (Aug. 21, 2009), p. 046001. ISSN: 1478-3975. DOI: 10.1088/1478-3975/6/4/046001 (cited on pp. 10, 25, 73).

[57] J. S. van Zon and P. R. ten Wolde. "Simulating Biochemical Networks at the Particle Level and in Time and Space: Green's Function Reaction Dynamics". In: *Physical Review Letters* 94.12 (Apr. 1, 2005), p. 128103. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.94.128103 (cited on pp. 10, 60, 82).

[58] A. Donev, C.-Y. Yang, and C. Kim. "Efficient reactive Brownian dynamics". In: *The Journal of Chemical Physics* 148.3 (Jan. 21, 2018), p. 034103. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.5009464 (cited on pp. 10, 82).

[59] S. A. Isaacson. "A Convergent Reaction-Diffusion Master Equation". In: *The Journal of Chemical Physics* 139.5 (Aug. 7, 2013), p. 054101. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.4816377 (cited on p. 10).

[60] S. A. Isaacson. "The Reaction-Diffusion Master Equation as an Asymptotic Approximation of Diffusion to a Small Target". In: *SIAM Journal on Applied Mathematics* 70.1 (Jan. 2009), pp. 77–111. ISSN: 0036-1399, 1095-712X. DOI: 10.1137/070705039 (cited on pp. 10, 82).

[61] R. Erban, S. J. Chapman, and P. K. Maini. "A practical guide to stochastic simulations of reaction-diffusion processes". In: *arXiv preprint arXiv:0704.1908* (2007), p. 35. DOI: 10.48550/arXiv.0704.1908 (cited on p. 10).

[62] S. Winkelmann and C. Schütte. "Hybrid models for chemical reaction networks: Multiscale theory and application to gene regulatory systems". In: *The Journal of Chemical Physics* 147.11 (Sept. 21, 2017), p. 114115. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.4986560 (cited on pp. 10, 22, 82).

[63] B. Franz, M. B. Flegg, S. J. Chapman, and R. Erban. "Multiscale Reaction-Diffusion Algorithms: PDE-Assisted Brownian Dynamics". In: *SIAM Journal on Applied Mathematics* 73.3 (Jan. 2013), pp. 1224–1247. ISSN: 0036-1399, 1095-712X. DOI: 10.1137/120882469 (cited on p. 10).

[64] M. Doi. "Theory of diffusion-controlled reaction between non-simple molecules. I". In: *Chemical Physics* 11.1 (Oct. 1975), pp. 107–113. ISSN: 03010104. DOI: 10.1016/0301-0104(75)80043-7 (cited on pp. 10, 25, 73).

[65] M. Doi. "Theory of diffusion-controlled reaction between non-simple molecules. II". In: *Chemical Physics* 11.1 (Oct. 1975), pp. 115–121. ISSN: 03010104. DOI: 10.1016/0301-0104(75)80044-9 (cited on pp. 10, 25, 73).

[66] E. Teramoto and N. Shigesada. "Theory of bimolecular reaction processes in liquids". In: *Progress of Theoretical Physics* 37.1 (1967), pp. 29–51. DOI: 10.1143/PTP.37.29 (cited on pp. 10, 25).

[67] M. Doi. "Stochastic theory of diffusion-controlled reaction". In: *Journal of Physics A: Mathematical and General* 9.9 (Sept. 1976), pp. 1479–1495. ISSN: 0305-4470, 1361-6447. DOI: 10.1088/0305-4470/9/9/009 (cited on pp. 10, 25).

[68] J. Schöneberg and F. Noé. “ReaDDy - A Software for Particle-Based Reaction-Diffusion Dynamics in Crowded Cellular Environments”. In: *PLoS ONE* 8.9 (Sept. 11, 2013). Ed. by R. J. Najmanovich, e74261. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0074261 (cited on pp. 10, 26, 60, 69, 70, 82).

[69] J. S. van Zon and P. R. ten Wolde. “Green’s-function reaction dynamics: A particle-based approach for simulating biochemical networks in time and space”. In: *The Journal of Chemical Physics* 123.23 (Dec. 15, 2005), p. 234910. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.2137716 (cited on pp. 10, 60, 82).

[70] R. A. Kerr, T. M. Bartol, B. Kaminsky, M. Dittrich, J.-C. J. Chang, S. B. Baden, T. J. Sejnowski, and J. R. Stiles. “Fast Monte Carlo simulation methods for biological reaction-diffusion systems in solution and on surfaces”. In: *SIAM journal on scientific computing* 30.6 (2008), pp. 3126–3149. DOI: 10.1137/070692017 (cited on p. 10).

[71] P. J. Michalski and L. M. Loew. “SpringSaLaD: a spatial, particle-based biochemical simulation platform with excluded volume”. In: *Biophysical journal* 110.3 (2016), pp. 523–529. DOI: 10.1016/j.bpj.2015.12.026 (cited on p. 10).

[72] S. N. Arjunan and K. Takahashi. “Multi-algorithm particle simulations with Spatiocyte”. In: *Protein Function Prediction*. Springer, 2017, pp. 219–236. DOI: 10.1007/978-1-4939-7015-5_16 (cited on p. 10).

[73] S. S. Andrews. “Particle-Based Stochastic Simulators”. In: *Encyclopedia of Computational Neuroscience*. Ed. by D. Jaeger and R. Jung. New York, NY: Springer New York, 2018, pp. 1–5. ISBN: 978-1-4614-7320-6. DOI: 10.1007/978-1-4614-7320-6_191-2 (cited on pp. 10, 82).

[74] T. R. Sokolowski, J. Paijmans, L. Bossen, T. Miedema, M. Wehrens, N. B. Becker, K. Kaizu, K. Takahashi, M. Dogterom, and P. R. Ten Wolde. “eGFRD in all dimensions”. In: *The Journal of chemical physics* 150.5 (2019), p. 054108. DOI: 10.1063/1.5064867 (cited on p. 10).

[75] S. N. Arjunan, A. Miyauchi, K. Iwamoto, and K. Takahashi. “pSpatiocyte: a high-performance simulator for intracellular reaction-diffusion systems”. In: *BMC bioinformatics* 21.1 (2020), pp. 1–21. DOI: 10.1186/s12859-019-3338-8 (cited on p. 10).

[76] M. F. Hagan. “Modeling viral capsid assembly”. In: *Advances in chemical physics* 155 (2014), p. 1. DOI: 10.1002/9781118755815.ch01 (cited on p. 11).

[77] S. K. Sadiq. “Reaction–diffusion basis of retroviral infectivity”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2080 (2016), p. 20160148. DOI: 10.1098/rsta.2016.0148 (cited on p. 11).

[78] J. Jäger, P. Patra, C. P. Sanchez, M. Lanzer, and U. S. Schwarz. “A particle-based computational model to analyse remodelling of the red blood cell cytoskeleton during malaria infections”. In: *PLOS Computational Biology* 18.4 (Apr. 8, 2022). Ed. by A. L. Marsden, e1009509. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1009509 (cited on p. 11).

[79] L. B. Koralov and I . G. Sinaĭ. *Theory of probability and random processes*. Second edition, second printing. Universitext. Heidelberg: Springer, 2012. 351 pp. ISBN: 978-3-540-25484-3. DOI: 10.1007/978-3-540-68829-7 (cited on pp. 13, 26).

[80] S. Banach and A. Tarski. “Sur la décomposition des ensembles de points en parties respectivement congruentes”. In: *Fundamenta Mathematicae* 6 (1924), pp. 244–277. ISSN: 0016-2736, 1730-6329. DOI: 10.4064/fm-6-1-244-277 (cited on p. 14).

[81] J. Elstrodt. *Maß- und Integrationstheorie*. Vol. 8. Berlin ; Heidelberg: Springer Spektrum, 1996. ISBN: 978-3-662-57939-8. DOI: `10.1007/978-3-642-17905-1` (cited on p. 14).

[82] H. Bauer. *Probability theory*. De Gruyter studies in mathematics 23. Berlin ; New York: Walter de Gruyter, 1996. 523 pp. ISBN: 978-3-11-013935-8. DOI: `10.1515/9783110814668` (cited on pp. 14–16).

[83] B. K. Øksendal. *Stochastic differential equations: an introduction with applications*. 6. ed., 6. corr. print. Universitext. Berlin ; Heidelberg: Springer, 2013. 379 pp. ISBN: 978-3-642-14394-6. DOI: `10.1007/978-3-642-14394-6` (cited on p. 14).

[84] B. O. Koopman. "Hamiltonian Systems and Transformation in Hilbert Space". In: *Proceedings of the National Academy of Sciences* 17.5 (1931), pp. 315–318. DOI: `10.1073/pnas.17.5.315` (cited on pp. 16, 17).

[85] P. Gaspard. *Chaos, Scattering and Statistical Mechanics*. Cambridge Nonlinear Science Series. Cambridge, England: Cambridge University Press, 1998. DOI: `10.1017/CBO9780511628856` (cited on p. 16).

[86] I. Mezić. "Spectral properties of dynamical systems, model reduction and decompositions". In: *Nonlinear Dynamics* 41.1 (2005), pp. 309–325. DOI: `10.1007/s11071-005-2824-x` (cited on p. 16).

[87] S. Klus, P. Koltai, and C. Schütte. "On the numerical approximation of the Perron–Frobenius and Koopman operator". In: *Journal of Computational Dynamics* 3.1 (2016), pp. 51–79. DOI: `10.3934/jcd.2016003` (cited on pp. 16, 18, 19).

[88] P. Koltai, H. Wu, F. Noé, and C. Schütte. "Optimal data-driven estimation of generalized Markov state models for non-equilibrium dynamics". In: *Computation* 6.1 (2018), p. 22. DOI: `10.3390/computation6010022` (cited on pp. 16, 38, 45).

[89] S. Klus, F. Nüske, P. Koltai, H. Wu, I. Kevrekidis, C. Schütte, and F. Noé. "Data-driven model reduction and transfer operator approximation". In: *Journal of Nonlinear Science* 28.3 (2018), pp. 985–1010. DOI: `10.1007/s00332-017-9437-7` (cited on pp. 16, 18, 34, 36).

[90] H. Wu and F. Noé. "Variational approach for learning Markov processes from time series data". In: *Journal of Nonlinear Science* 30.1 (2020), pp. 23–66. DOI: `10.1007/s00332-019-09567-y` (cited on pp. 16, 18, 37, 45, 50).

[91] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz. "Modern Koopman Theory for Dynamical Systems". In: *arXiv preprint arXiv:2102.12086* (2021). DOI: `10.48550/arXiv.2102.12086` (cited on p. 16).

[92] A. Lasota and M. C. Mackey. *Chaos, fractals, and noise: Stochastic aspects of dynamics*. Vol. 97. New York, NY: Springer Science & Business Media, 1994. ISBN: 978-1-4612-4286-4. DOI: `10.1007/978-1-4612-4286-4` (cited on p. 18).

[93] A. Boyarsky and P. Góra. *Laws of Chaos: Invariant Measures and Dynamical Systems in One Dimension*. English. OCLC: 958523230. Cambridge, MA: Birkhäuser Boston, 1997. ISBN: 978-1-4612-2024-4. DOI: `10.1007/978-1-4612-2024-4` (cited on p. 18).

[94] W. Tian and H. Wu. "Kernel Embedding Based Variational Approach for Low-Dimensional Approximation of Dynamical Systems". In: *Computational Methods in Applied Mathematics* (2021). DOI: `10.1515/cmam-2020-0130` (cited on pp. 18, 38, 39, 41).

[95] A. Denner. "Coherent structures and transfer operators". Dissertation. München, Germany: Technische Universität München, 2017 (cited on p. 19).

[96] S. P. Meyn and R. L. Tweedie. *Markov chains and stochastic stability*. London, England: Springer Science & Business Media, 2012. ISBN: 978-1-4471-3267-7. DOI: 10.1007/978-1-4471-3267-7 (cited on p. 19).

[97] C. Schütte and M. Sarich. *Metastability and Markov State Models in Molecular Dynamics: Modeling, Analysis, Algorithmic Approaches*. Vol. 24. Providence, RI: American Mathematical Soc., 2013. ISBN: 978-1-4704-1439-9. DOI: 10.1090/cln/024 (cited on p. 19).

[98] V. N. Kolokoltsov. *Markov processes, semigroups, and generators*. De Gruyter studies in mathematics 38. Berlin ; New York: De Gruyter, 2011. 430 pp. ISBN: 978-3-11-025010-7. DOI: 10.1515/9783110250114 (cited on p. 19).

[99] G. A. Pavliotis. *Stochastic Processes and Applications*. Vol. 60. Texts in Applied Mathematics. New York, NY: Springer New York, 2014. ISBN: 978-1-4939-1322-0. DOI: 10.1007/978-1-4939-1323-7 (cited on p. 20).

[100] A. Kolmogorov. "Über die analytischen Methoden in der Wahrscheinlichkeitsrechnung". In: *Mathematische Annalen* 104 (1931), pp. 415–458. DOI: 10.1007/BF01457949 (cited on p. 20).

[101] A. D. Fokker. "Die mittlere Energie rotierender elektrischer Dipole im Strahlungsfeld". In: *Annalen der Physik* 348.5 (1914), pp. 810–820. ISSN: 00033804, 15213889. DOI: 10.1002/andp.19143480507 (cited on p. 20).

[102] M. Planck. "Über einen Satz der statistischen Dynamik und seine Erweiterung in der Quantentheorie". In: *Sitzungberichte der Königlich-Preussischen Akademie der Wissenschaften* 24 (1917), pp. 324–341 (cited on p. 20).

[103] P. E. Kloeden and E. Platen. *Numerical solution of stochastic differential equations*. corr. 3. print. Applications of mathematics 23. Berlin: Springer, 2010. 636 pp. ISBN: 978-3-642-08107-1. DOI: 10.1007/978-3-662-12616-5 (cited on p. 20).

[104] P. Waage and C. M. Gulberg. "Studies concerning affinity". In: *Journal of chemical education* 63.12 (1986), p. 1044. DOI: 10.1021/ed063p1044 (cited on p. 22).

[105] D. T. Gillespie. "Stochastic Simulation of Chemical Kinetics". In: *Annual Review of Physical Chemistry* 58.1 (May 2007), pp. 35–55. ISSN: 0066-426X, 1545-1593. DOI: 10.1146/annurev.physchem.58.032806.104637 (cited on p. 22).

[106] A. J. Lotka. *Elements of physical biology*. Williams & Wilkins, 1925 (cited on p. 23).

[107] V. Volterra. "Variazioni e fluttuazioni del numero d'individui in specie animali conviventi". In: (1927) (cited on p. 23).

[108] D. T. Gillespie. "A rigorous derivation of the chemical master equation". In: *Physica A: Statistical Mechanics and its Applications* 188.1-3 (1992), pp. 404–425. DOI: 10.1016/0378-4371(92)90283-V (cited on p. 24).

[109] D. J. Higham. "Modeling and simulating chemical reactions". In: *SIAM review* 50.2 (2008), pp. 347–368. DOI: 10.1137/060666457 (cited on p. 24).

[110] D. T. Gillespie. "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions". In: *Journal of computational physics* 22.4 (1976), pp. 403–434. DOI: 10.1016/0021-9991(76)90041-3 (cited on pp. 24, 82, 86).

[111] D. T. Gillespie. "Exact stochastic simulation of coupled chemical reactions". In: *The journal of physical chemistry* 81.25 (1977), pp. 2340–2361. DOI: 10.1021/j100540a008 (cited on pp. 24, 82, 86).

[112] M. v. Smoluchowski. “Grundriß der Koagulationskinetik kolloider Lösungen”. In: *Kolloid-Zeitschrift* 21.3 (Sept. 1917), pp. 98–104. ISSN: 0303-402X, 1435-1536. DOI: 10.1007/BF01427232 (cited on p. 25).

[113] V. Fock. “Konfigurationsraum und zweite Quantelung”. In: *Zeitschrift für Physik* 75.9 (1932), pp. 622–647. DOI: 10.1007/BF01344458 (cited on p. 28).

[114] M. J. del Razo, D. Frömberg, A. V. Straube, C. Schütte, F. Höfling, and S. Winkelmann. “A probabilistic framework for particle-based reaction–diffusion dynamics using classical Fock space representations”. In: *Letters in Mathematical Physics* 112.3 (June 2022), p. 49. ISSN: 0377-9017, 1573-0530. DOI: 10.1007/s11005-022-01539-w (cited on p. 28).

[115] L. Buitinck et al. “API design for machine learning software: experiences from the scikit-learn project”. In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122. DOI: 10.48550/arXiv.1309.0238 (cited on pp. 30, 32).

[116] L. Molgedey and H. G. Schuster. “Separation of a mixture of independent signals using time delayed correlations”. In: *Physical Review Letters* 72.23 (1994), p. 3634. DOI: 10.1103/PhysRevLett.72.3634 (cited on pp. 30, 35).

[117] Y. Naritomi and S. Fuchigami. “Slow dynamics in protein fluctuations revealed by time-structure based independent component analysis: The case of domain motions”. In: *The Journal of chemical physics* 134.6 (2011), 02B617. DOI: 10.1063/1.3554380 (cited on pp. 30, 35).

[118] P. J. Schmid. “Dynamic mode decomposition of numerical and experimental data”. In: *Journal of fluid mechanics* 656 (2010), pp. 5–28. DOI: 10.1017/S0022112010001217 (cited on pp. 30, 33, 35).

[119] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz. “On dynamic mode decomposition: theory and applications”. In: *Journal of Computational Dynamics* 1.2 (2014), pp. 391–421. DOI: 10.3934/jcd.2014.1.391 (cited on pp. 30, 33, 35).

[120] A. Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035. DOI: 10.5555/3454287.3455008 (cited on pp. 31, 32).

[121] C. Wehmeyer and F. Noé. “Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics”. In: *J. Chem. Phys.* 148.24 (June 2018), p. 241703. DOI: 10.1063/1.5011399 (cited on pp. 31, 41).

[122] S. E. Otto and C. W. Rowley. “Linearly-Recurrent Autoencoder Networks for Learning Dynamics”. In: *SIAM Journal on Applied Dynamical Systems* 18.1 (2019), pp. 558–593. DOI: 10.1137/18M1177846 (cited on pp. 31, 41).

[123] H. Wu, A. Mardt, L. Pasquali, and F. Noe. “Deep Generative Markov State Models”. In: *Advances in Neural Information Processing Systems* 31 (2018), pp. 3975–3984. DOI: 10.5555/3327144.3327312 (cited on pp. 31, 39).

[124] A. Mardt and F. Noé. “Progress in deep Markov State Modeling: Coarse graining and experimental data restraints”. In: *The Journal of Chemical Physics* 155.21 (2021), p. 214106. DOI: 10.1063/5.0064668 (cited on p. 31).

[125] B. Lusch, J. N. Kutz, and S. L. Brunton. “Deep learning for universal linear embeddings of nonlinear dynamics”. In: *Nature Communications* 9.1 (2018), p. 4950. DOI: 10.1038/s41467-018-07210-0 (cited on pp. 31, 41).

[126] C. X. Hernández, H. K. Wayment-Steele, M. M. Sultan, B. E. Husic, and V. S. Pande. “Variational encoding of complex dynamics”. In: *Physical Review E* 97.6 (2018), p. 062412. DOI: 10.1103/PhysRevE.97.062412 (cited on pp. 31, 41).

[127] C. Schütte, A. Fischer, W. Huisinga, and P. Deuflhard. “A Direct Approach to Conformational Dynamics Based on Hybrid Monte Carlo”. In: *J. Comput. Phys.* 151.1 (May 1999), pp. 146–168. ISSN: 0021-9991. DOI: 10.1006/jcph.1999.6231 (cited on pp. 31, 50).

[128] W. C. Swope, J. W. Pitera, and F. Suits. “Describing Protein Folding Kinetics by Molecular Dynamics Simulations. 1. Theory”. In: *J. Phys. Chem. B* 108.21 (May 2004), pp. 6571–6581. ISSN: 1520-6106. DOI: 10.1021/jp037421y (cited on pp. 31, 50).

[129] N. Singhal, C. D. Snow, and V. S. Pande. “Using Path Sampling to Build Better Markovian State Models: Predicting the Folding Rate and Mechanism of a Tryptophan Zipper Beta Hairpin”. In: *J. Chem. Phys.* 121.1 (June 2004), pp. 415–425. ISSN: 0021-9606. DOI: 10.1063/1.1738647 (cited on pp. 31, 50).

[130] F. Noé, I. Horenko, C. Schütte, and J. C. Smith. “Hierarchical Analysis of Conformational Dynamics in Biomolecules: Transition Networks of Metastable States”. In: *J. Chem. Phys.* 126.15 (Apr. 2007), p. 155102. ISSN: 0021-9606. DOI: 10.1063/1.2714539 (cited on pp. 31, 50).

[131] F. Noé. “Probability Distributions of Molecular Observables Computed from Markov Models”. In: *J. Chem. Phys.* 128.24 (June 2008), p. 244103. ISSN: 0021-9606. DOI: 10.1063/1.2916718 (cited on pp. 31, 50).

[132] F. Noé, C. Schütte, E. Vanden-Eijnden, L. Reich, and T. R. Weikl. “Constructing the Equilibrium Ensemble of Folding Pathways from Short Off-Equilibrium Simulations”. In: *Proc. Natl. Acad. Sci.* 106.45 (Nov. 2009), pp. 19011–19016. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0905466106 (cited on pp. 31, 50).

[133] J.-H. Prinz, H. Wu, M. Sarich, B. Keller, M. Senne, M. Held, J. D. Chodera, C. Schütte, and F. Noé. “Markov Models of Molecular Kinetics: Generation and Validation”. In: *J. Chem. Phys.* 134.17 (2011), p. 174105. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.3565032 (cited on pp. 31, 50, 52).

[134] J. D. Chodera and F. Noé. “Markov State Models of Biomolecular Conformational Dynamics”. In: *Current Opinion in Structural Biology* 25 (Apr. 2014), pp. 135–144. ISSN: 0959440X. DOI: 10.1016/j.sbi.2014.04.002 (cited on pp. 31, 50).

[135] B. E. Husic and V. S. Pande. “Markov State Models: From an Art to a Science”. In: *J. Am. Chem. Soc.* 140.7 (Feb. 2018), pp. 2386–2396. ISSN: 0002-7863, 1520-5126. DOI: 10.1021/jacs.7b12191 (cited on pp. 31, 50).

[136] L. E. Baum and T. Petrie. “Statistical Inference for Probabilistic Functions of Finite State Markov Chains”. In: *The Annals of Mathematical Statistics* 37.6 (1966), pp. 1554–1563. DOI: 10.1214/aoms/1177699147 (cited on pp. 31, 52).

[137] L. R. Rabiner. “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition”. In: *Proc. IEEE* 77.2 (Feb. 1989), pp. 257–286. ISSN: 0018-9219. DOI: 10.1109/5.18626 (cited on pp. 31, 54).

[138] F. Nüske, H. Wu, J.-H. Prinz, C. Wehmeyer, C. Clementi, and F. Noé. “Markov State Models from Short Non-Equilibrium Simulations—Analysis and Correction of Estimation Bias”. In: *The Journal of Chemical Physics* 146.9 (Mar. 2017), p. 094104. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.4976518 (cited on pp. 31, 52).

[139] D. R. Roe and T. E. Cheatham III. “PTRAJ and CPPTRAJ: software for processing and analysis of molecular dynamics trajectory data”. In: *Journal of chemical theory and computation* 9.7 (2013), pp. 3084–3095. DOI: `10.1021/ct400341p` (cited on p. 31).

[140] T. D. Romo, N. Leioatts, and A. Grossfield. “Lightweight object oriented structure analysis: tools for building tools to analyze molecular dynamics simulations”. In: *Journal of computational chemistry* 35.32 (2014), pp. 2305–2318. DOI: `10.1002/jcc.23753` (cited on p. 31).

[141] R. T. McGibbon, K. A. Beauchamp, M. P. Harrigan, C. Klein, J. M. Swails, C. X. Hernández, C. R. Schwantes, L.-P. Wang, T. J. Lane, and V. S. Pande. “MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories”. In: *Biophys. J.* 109.8 (Oct. 2015), pp. 1528–1532. ISSN: 0006-3495. DOI: `10.1016/j.bpj.2015.08.015` (cited on p. 31).

[142] N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein. “MDAnalysis: a toolkit for the analysis of molecular dynamics simulations”. In: *Journal of computational chemistry* 32.10 (2011), pp. 2319–2327. DOI: `10.1002/jcc.21787` (cited on p. 31).

[143] H. Nguyen, D. R. Roe, J. Swails, and D. A. Case. “PYTRAJ: Interactive data analysis for molecular dynamics simulations”. In: *New Brunswick, NJ: Rutgers University* (2016). URL: `https://github.com/Amber-MD/pytraj` (cited on p. 31).

[144] R. J. Gowers, M. Linke, J. Barnoud, T. J. E. Reddy, M. N. Melo, S. L. Seyler, J. Domanski, D. L. Dotson, S. Buchoux, I. M. Kenney, et al. *MDAnalysis: a Python package for the rapid analysis of molecular dynamics simulations.* Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2019. DOI: `10.25080/MAJORA-629E541A-00E` (cited on p. 31).

[145] C. Wehmeyer, M. K. Scherer, T. Hempel, B. E. Husic, S. Olsson, and F. Noé. “Introduction to Markov state modeling with the PyEMMA software [Article v1.0]”. In: *Living Journal of Computational Molecular Science* 1.1 (2019). ISSN: 25756524. DOI: `10.33011/livecoms.1.1.5965` (cited on pp. 31, 33).

[146] D. De Sancho and A. Aguirre. “MasterMSM: A package for constructing master equation models of molecular dynamics”. In: *Journal of chemical information and modeling* 59.9 (2019), pp. 3625–3629. URL: `https://doi.org/10.1021/acs.jcim.9b00468` (cited on p. 31).

[147] N. Demo, M. Tezzele, and G. Rozza. “PyDMD: Python Dynamic Mode Decomposition”. In: *The Journal of Open Source Software* 3.22 (2018), p. 530. DOI: `10.21105/joss.00530` (cited on pp. 31, 33, 35).

[148] R. Weiss et al. *hmmlearn.* Version 0.2.5. Feb. 3, 2021. URL: `https://hmmlearn.readthedocs.io/` (cited on p. 31).

[149] B. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. Kutz, and S. Brunton. “PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data”. In: *Journal of Open Source Software* 5.49 (2020), p. 2104. DOI: `10.21105/joss.02104` (cited on pp. 31, 33, 55).

[150] M. Löning, A. Bagnall, S. Ganesh, V. Kazakov, J. Lines, and F. J. Király. “sktime: A unified interface for machine learning with time series”. In: *arXiv preprint arXiv:1909.07872* (2019). DOI: `10.48550/arXiv.1909.07872` (cited on p. 31).

[151] conda-forge community. *The conda-forge Project: Community-based Software Distribution Built on the conda Package Format and Ecosystem.* July 2015. DOI: `10.5281/zenodo.4774216` (cited on p. 31).

[152] W. Jakob, J. Rhinelander, and D. Moldovan. *pybind11 – Seamless operability between C++11 and Python*. 2017. URL: `https://github.com/pybind/pybind11` (cited on p. 31).

[153] C. R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: `10.1038/s41586-020-2649-2` (cited on pp. 31, 32).

[154] P. Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: `10.1038/s41592-019-0686-2` (cited on pp. 31, 32, 84).

[155] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: `10.1109/MCSE.2007.55` (cited on p. 32).

[156] H. Krekel, B. Oliveira, R. Pfannschmidt, F. Bruynooghe, B. Laugher, and F. Bruhin. *pytest 6.2*. 2004. URL: `https://github.com/pytest-dev/pytest` (cited on p. 32).

[157] T. Kluyver et al. “Jupyter Notebooks - a publishing format for reproducible computational workflows”. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ed. by F. Loizides and B. Scmidt. Netherlands: IOS Press, 2016, pp. 87–90. URL: `https://eprints.soton.ac.uk/403913/` (cited on p. 32).

[158] J. D. Chodera, P. Elms, F. Noé, B. Keller, C. M. Kaiser, A. Ewall-Wice, S. Marqusee, C. Bustamante, and N. S. Hinrichs. “Bayesian hidden Markov model analysis of single-molecule force spectroscopy: Characterizing kinetics under measurement uncertainty”. In: *arXiv preprint arXiv:1108.1430* (2011). DOI: `10.48550/arXiv.1108.1430` (cited on pp. 33, 54).

[159] C. W. Rowley, I. Mezic, S. Bagheri, P. Schlatter, and D. Henningson. “Spectral analysis of nonlinear flows”. In: *J. Fluid Mech.* 645 (2009), pp. 115–127. DOI: `10.1017/S0022112009992059` (cited on pp. 33, 35).

[160] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. Philadelphia, PA: SIAM, 2016. ISBN: 978-1-61197-449-2. DOI: `10.1137/1.9781611974508` (cited on pp. 33, 35).

[161] S. Klus, B. E. Husic, M. Mollenhauer, and F. Noé. “Kernel methods for detecting coherent structures in dynamical data”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 29.12 (2019), p. 123112. DOI: `10.1063/1.5100267` (cited on pp. 34, 38, 45).

[162] A. Glielmo, B. E. Husic, A. Rodriguez, C. Clementi, F. Noé, and A. Laio. “Unsupervised Learning Methods for Molecular Simulation Data”. en. In: *Chemical Reviews* (May 2021), acs.chemrev.0c01195. ISSN: 0009-2665, 1520-6890. DOI: `10.1021/acs.chemrev.0c01195` (cited on p. 34).

[163] J. L. Proctor, S. L. Brunton, and J. N. Kutz. “Dynamic mode decomposition with control”. In: *SIAM Journal on Applied Dynamical Systems* 15.1 (2016), pp. 142–161. DOI: `10.1137/15M1013857` (cited on p. 35).

[164] M. R. Jovanović, P. J. Schmid, and J. W. Nichols. “Sparsity-promoting dynamic mode decomposition”. In: *Physics of Fluids* 26.2 (2014), p. 024103. DOI: `10.1063/1.4863670` (cited on p. 35).

[165] N. B. Erichson, L. Mathelin, J. N. Kutz, and S. L. Brunton. “Randomized dynamic mode decomposition”. In: *SIAM Journal on Applied Dynamical Systems* 18.4 (2019), pp. 1867–1891. DOI: `10.1137/18M1215013` (cited on p. 35).

[166] S. L. Brunton, B. W. Brunton, J. L. Proctor, E. Kaiser, and J. N. Kutz. “Chaos as an intermittently forced linear system”. In: *Nature Communications* 8.19 (2017), pp. 1–9. DOI: `10.1038/s41467-017-00030-8` (cited on p. 35).

[167] S. Bagheri. “Effects of weak noise on oscillating flows: Linking quality factor, Floquet modes, and Koopman spectrum”. In: *Physics of Fluids* 26.9 (2014), p. 094104. DOI: 10.1063/1.4895898 (cited on p. 35).

[168] M. S. Hemati, C. W. Rowley, E. A. Deem, and L. N. Cattafesta. “De-Biasing the Dynamic Mode Decomposition for Applied Koopman Spectral Analysis”. In: *Theoretical and Computational Fluid Dynamics* 31.4 (2017), pp. 349–368. DOI: 10.1007/s00162-017-0432-2 (cited on p. 35).

[169] S. T. Dawson, M. S. Hemati, M. O. Williams, and C. W. Rowley. “Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition”. In: *Experiments in Fluids* 57.3 (2016), pp. 1–19. DOI: 10.1007/s00348-016-2127-7 (cited on p. 35).

[170] N. Takeishi, Y. Kawahara, Y. Tabei, and T. Yairi. “Bayesian Dynamic Mode Decomposition”. In: *Twenty-Sixth International Joint Conference on Artificial Intelligence* (2017). DOI: 10.24963/ijcai.2017/392 (cited on p. 35).

[171] T. Askham and J. N. Kutz. “Variable projection methods for an optimized dynamic mode decomposition”. In: *SIAM Journal on Applied Dynamical Systems* 17.1 (2018), pp. 380–416. DOI: 10.1137/M1124176 (cited on p. 35).

[172] O. Azencot, W. Yin, and A. Bertozzi. “Consistent dynamic mode decomposition”. In: *SIAM Journal on Applied Dynamical Systems* 18.3 (2019), pp. 1565–1585. DOI: 10.1137/18M1233960 (cited on p. 35).

[173] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. “A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition”. In: *Journal of Nonlinear Science* 25.6 (2015), pp. 1307–1346. DOI: 10.1007/s00332-015-9258-5 (cited on pp. 35, 36).

[174] G. Pérez-Hernández, F. Paul, T. Giorgino, G. D. Fabritiis, and F. Noé. “Identification of Slow Molecular Order Parameters for Markov Model Construction”. In: *J. Chem. Phys.* 139.1 (July 2013), p. 015102. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.4811489 (cited on pp. 35, 36).

[175] C. R. Schwantes and V. S. Pande. “Improvements in Markov state model construction reveal many non-native interactions in the folding of NTL9”. In: *Journal of chemical theory and computation* 9.4 (2013), pp. 2000–2009. DOI: 10.1021/ct300878a (cited on p. 35).

[176] F. Nüske, B. G. Keller, G. Pérez-Hernández, A. S. Mey, and F. Noé. “Variational approach to molecular kinetics”. In: *Journal of chemical theory and computation* 10.4 (2014), pp. 1739–1752. DOI: 10.1021/ct4009156 (cited on p. 36).

[177] F. Noé and F. Nüske. “A variational approach to modeling slow processes in stochastic dynamical systems”. In: *Multiscale Modeling & Simulation* 11.2 (2013), pp. 635–655. DOI: 10.1137/110858616 (cited on p. 36).

[178] B. E. Husic and F. Noé. “Deflation reveals dynamical structure in nondominant reaction coordinates”. In: *The Journal of Chemical Physics* 151.5 (2019), p. 054103. DOI: 10.1063/1.5099194 (cited on p. 37).

[179] T. F. Chan, G. H. Golub, and R. J. LeVeque. “Updating formulae and a pairwise algorithm for computing sample variances”. In: *COMPSTAT 1982 5th Symposium held at Toulouse 1982*. Springer. 1982, pp. 30–41. DOI: 10.1007/978-3-642-51461-6_3 (cited on p. 37).

[180] H. Hotelling. “Relations Between Two Sets of Variates”. In: *Biometrika* 28.3/4 (1936), pp. 321–377. DOI: 10.2307/2333955 (cited on p. 38).

[181] M. O. Williams, C. W. Rowley, and I. G. Kevrekidis. “A Kernel-Based Method for Data-Driven Koopman Spectral Analysis”. In: *Journal of Computational Dynamics* 2.2 (2015), pp. 247–265. DOI: 10.3934/jcd.2015005 (cited on p. 38).

[182] S. Klus, I. Schuster, and K. Muandet. “Eigendecompositions of transfer operators in reproducing kernel Hilbert spaces”. In: *Journal of Nonlinear Science* 30.1 (2020), pp. 283–315. DOI: 10.1007/s00332-019-09574-z (cited on p. 38).

[183] N. Takeishi, Y. Kawahara, and T. Yairi. “Learning Koopman Invariant Subspaces for Dynamic Mode Decomposition”. In: *Advances in Neural Information Processing Systems.* 2017, pp. 1130–1140. DOI: 10.5555/3294771.3294879 (cited on p. 41).

[184] E. Yeung, S. Kundu, and N. Hodas. “Learning Deep Neural Network Representations for Koopman Operators of Nonlinear Dynamical Systems”. In: *2019 American Control Conference (ACC).* 2019, pp. 4832–4839. DOI: 10.23919/ACC.2019.8815339 (cited on p. 41).

[185] W. Chen and A. L. Ferguson. “Molecular enhanced sampling with autoencoders: On-the-fly collective variable discovery and accelerated free energy landscape exploration”. In: *Journal of computational chemistry* 39.25 (2018), pp. 2079–2102. DOI: 10.1002/jcc.25520 (cited on p. 41).

[186] D. P. Kingma and M. Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013). DOI: 10.48550/arXiv.1312.6114 (cited on p. 41).

[187] D. J. Rezende, S. Mohamed, and D. Wierstra. “Stochastic backpropagation and approximate inference in deep generative models”. In: *International conference on machine learning.* PMLR. 2014, pp. 1278–1286. DOI: 10.5555/3044805.3045035 (cited on p. 41).

[188] T. Kadir and M. Brady. “Saliency, scale and image description”. In: *International Journal of Computer Vision* 45.2 (2001), pp. 83–105. DOI: 10.1023/A:1012460413855 (cited on p. 41).

[189] S. Lloyd. “Least squares quantization in PCM”. In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489 (cited on p. 42).

[190] R. T. McGibbon and V. S. Pande. “Variational cross-validation of slow dynamical modes in molecular kinetics”. In: *The Journal of chemical physics* 142.12 (2015), 03B621_1. DOI: 10.1063/1.4916292 (cited on p. 42).

[191] F. Noé and C. Clementi. “Kinetic distance and kinetic maps from molecular dynamics simulation”. In: *Journal of Chemical Theory and Computation* 11.10 (2015), pp. 5002–5011. DOI: 10.1021/acs.jctc.5b00553 (cited on p. 42).

[192] D. Kraft. “A software package for sequential quadratic programming”. In: *Technical Report DFVLR-FB 88-28, Institut für Dynamik der Flugsysteme, Oberpfaffenhofen* (1988). URL: https://books.google.de/books?id=4rKaGwAACAAJ (cited on pp. 44, 47, 84).

[193] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014). DOI: 10.48550/arXiv.1412.6980 (cited on p. 44).

[194] G. Froyland, N. Santitissadeekorn, and A. Monahan. “Transport in time-dependent dynamical systems: Finite-time coherent sets”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 20.4 (2010), p. 043116. DOI: 10.1063/1.3502450 (cited on p. 45).

[195] R. Banisch and P. Koltai. “Understanding the geometry of transport: Diffusion maps for Lagrangian trajectory data unravel coherent sets”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27.3 (2017), p. 035804. DOI: 10.1063/1.4971788 (cited on p. 45).

[196] G. Froyland and K. Padberg-Gehle. “Finite-time entropy: A probabilistic approach for measuring nonlinear stretching”. In: *Physica D: Nonlinear Phenomena* 241.19 (2012), pp. 1612–1628. DOI: 10.1016/j.physd.2012.06.010 (cited on p. 45).

[197] A. Hadjighasem, D. Karrasch, H. Teramoto, and G. Haller. “Spectral-clustering approach to Lagrangian vortex detection”. In: *Physical Review E* 93.6 (2016), p. 063107. DOI: 10.1103/PhysRevE.93.063107 (cited on p. 45).

[198] B. E. Husic, K. L. Schlueter-Kuck, and J. O. Dabiri. “Simultaneous coherent structure coloring facilitates interpretable clustering of scientific data by amplifying dissimilarity”. In: *Plos one* 14.3 (2019), e0212442. DOI: 10.1371/journal.pone.0212442 (cited on p. 45).

[199] P. Metzner, C. Schütte, and E. Vanden-Eijnden. “Transition Path Theory for Markov Jump Processes”. In: *Multiscale Model. Simul.* 7.3 (Jan. 2009), pp. 1192–1219. ISSN: 1540-3459, 1540-3467. DOI: 10.1137/070699500 (cited on p. 50).

[200] S. Röblitz and M. Weber. “Fuzzy Spectral Clustering by PCCA+: Application to Markov State Models and Data Classification”. In: *Adv. Data Anal. Classif.* 7.2 (June 2013), pp. 147–179. ISSN: 1862-5355. DOI: 10.1007/s11634-013-0134-6 (cited on pp. 50, 54).

[201] B. Trendelkamp-Schroer, H. Wu, F. Paul, and F. Noé. “Estimation and Uncertainty of Reversible Markov Models”. In: *J. Chem. Phys.* 143.17 (Nov. 2015), p. 174101. ISSN: 0021-9606. DOI: 10.1063/1.4934536 (cited on p. 52).

[202] S. Olsson, H. Wu, F. Paul, C. Clementi, and F. Noé. “Combining Experimental and Simulation Data of Molecular Processes via Augmented Markov Models”. In: *Proc. Natl. Acad. Sci.* 114.31 (Aug. 2017), pp. 8265–8270. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1704803114 (cited on p. 52).

[203] L. E. Baum and J. A. Eagon. “An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology”. In: *Bulletin of the American Mathematical Society* 73.3 (1967), pp. 360–363. DOI: bams/1183528841 (cited on p. 52).

[204] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. “A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains”. In: *The Annals of Mathematical Statistics* 41.1 (1970), pp. 164–171. DOI: 10.1214/aoms/1177697196 (cited on p. 52).

[205] L. E. Baum. “An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes”. In: *Inequalities III: Proceedings of the Third Symposium on Inequalities.* Vol. 3. 1. Academic Press, 1972, pp. 1–8 (cited on p. 52).

[206] A. Viterbi. “Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm”. In: *IEEE Trans. Inform. Theory* 13.2 (Apr. 1967), pp. 260–269. ISSN: 0018-9448, 1557-9654. DOI: 10.1109/TIT.1967.1054010 (cited on p. 54).

[207] S. Klus, F. Nüske, S. Peitz, J.-H. Niemann, C. Clementi, and C. Schütte. “Data-driven approximation of the Koopman generator: Model reduction, system identification, and control”. In: *Physica D: Nonlinear Phenomena* 406 (May 2020), p. 132416. ISSN: 01672789. DOI: 10.1016/j.physd.2020.132416 (cited on pp. 54, 55).

[208] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz. “Inferring Biological Networks by Sparse Identification of Nonlinear Dynamics”. In: *IEEE Transactions on Molecular, Biological and Multi-Scale Communications* 2.1 (2016), pp. 52–63. DOI: 10.1109/TMBMC.2016.2633265 (cited on pp. 54, 82).

[209] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. “Data-driven discovery of coordinates and governing equations”. In: *Proceedings of the National Academy of Sciences* 116.45 (2019), pp. 22445–22451. DOI: 10.1073/pnas.1906995116 (cited on pp. 54, 55).

[210] K. Champion, P. Zheng, A. Y. Aravkin, S. L. Brunton, and J. N. Kutz. “A unified sparse optimization framework to learn parsimonious physics-informed models from data”. In: *IEEE Access* 8 (2020), pp. 169259–169271. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3023625 (cited on p. 55).

[211] S. L. Brunton, J. L. Proctor, and J. N. Kutz. “Sparse identification of nonlinear dynamics with control (SINDYc)”. In: *IFAC-PapersOnLine* 49.18 (2016), pp. 710–715. DOI: 10.1016/j.ifacol.2016.10.249 (cited on p. 55).

[212] E. Kaiser, J. N. Kutz, and S. L. Brunton. “Sparse identification of nonlinear dynamics for model predictive control in the low-data limit”. In: *Proceedings of the Royal Society of London A* 474.2219 (2018). DOI: 10.1098/rspa.2018.0335 (cited on p. 55).

[213] J.-C. Loiseau and S. L. Brunton. “Constrained Sparse Galerkin Regression”. In: *Journal of Fluid Mechanics* 838 (2018), pp. 42–67. DOI: 10.1017/jfm.2017.823 (cited on p. 55).

[214] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz. “Data-driven discovery of partial differential equations”. In: *Science Advances* 3.e1602614 (2017). DOI: 10.1126/sciadv.1602614 (cited on p. 55).

[215] H. Schaeffer. “Learning partial differential equations via data discovery and sparse optimization”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473.2197 (2017), p. 20160446. DOI: 10.1098/rspa.2016.0446 (cited on p. 55).

[216] L. Boninsegna, F. Nüske, and C. Clementi. “Sparse learning of stochastic dynamical equations”. In: *The Journal of Chemical Physics* 148.24 (2018), p. 241723. DOI: 10.1063/1.5018409 (cited on p. 55).

[217] J. L. Callaham, J.-C. Loiseau, G. Rigas, and S. L. Brunton. “Nonlinear stochastic modelling with Langevin regression”. In: *Proceedings of the Royal Society A* 477.2250 (2021), p. 20210092. DOI: 10.1098/rspa.2021.0092 (cited on p. 55).

[218] M. Quade, M. Abel, J. Nathan Kutz, and S. L. Brunton. “Sparse identification of nonlinear dynamics for rapid model recovery”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28.6 (2018), p. 063116. DOI: 10.1063/1.5027470 (cited on p. 55).

[219] H. Schaeffer and S. G. McCalla. “Sparse model selection via integral terms”. In: *Physical Review E* 96.2 (2017), p. 023302. DOI: 10.1103/PhysRevE.96.023302 (cited on p. 55).

[220] D. R. Gurevich, P. A. Reinbold, and R. O. Grigoriev. “Robust and optimal sparse regression for nonlinear PDE models”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 29.10 (2019), p. 103113. DOI: 10.1063/1.5120861 (cited on p. 55).

[221] P. A. Reinbold, D. R. Gurevich, and R. O. Grigoriev. “Using noisy or incomplete data to discover models of spatiotemporal dynamics”. In: *Physical Review E* 101.1 (2020), p. 010203. DOI: 10.1103/PhysRevE.101.010203 (cited on p. 55).

[222] G. Tran and R. Ward. “Exact recovery of chaotic systems from highly corrupted data”. In: *Multiscale Modeling & Simulation* 15.3 (2017), pp. 1108–1129. DOI: 10.1137/16M1086637 (cited on p. 55).

[223] H. Schaeffer, G. Tran, and R. Ward. “Learning dynamical systems and bifurcation via group sparsity”. In: *arXiv preprint arXiv:1709.01558* (2017). DOI: 10.48550/arXiv.1709.01558 (cited on p. 55).

[224] L. Zhang and H. Schaeffer. "On the convergence of the SINDy algorithm". In: *Multiscale Modeling & Simulation* 17.3 (2019), pp. 948–972. DOI: 10.1137/18M1189828 (cited on p. 55).

[225] P. Gelß, S. Klus, J. Eisert, and C. Schütte. "Multidimensional approximation of nonlinear dynamical systems". In: *Journal of Computational and Nonlinear Dynamics* 14.6 (2019). DOI: 10.1115/1.4043148 (cited on p. 55).

[226] N. Deng, B. R. Noack, M. Morzyński, and L. R. Pastur. "Low-order model for successive bifurcations of the fluidic pinball". In: *Journal of fluid mechanics* 884.A37 (2020). DOI: 10.1017/jfm.2019.959 (cited on p. 55).

[227] S. Beetham, R. O. Fox, and J. Capecelatro. "Sparse identification of multiphase turbulence closures for coupled fluid–particle flows". In: *Journal of Fluid Mechanics* 914 (2021). DOI: 10.1017/jfm.2021.53 (cited on p. 55).

[228] O. E. Rössler. "An equation for continuous chaos". In: *Physics Letters A* 57.5 (1976), pp. 397–398. DOI: 10.1016/0375-9601(76)90101-8 (cited on p. 55).

[229] A. Arnold, O. Lenz, S. Kesselheim, R. Weeber, F. Fahrenberger, D. Roehm, P. Košovan, and C. Holm. "ESPResSo 3.1 — Molecular Dynamics Software for Coarse-Grained Models". In: *Meshfree Methods for Partial Differential Equations VI*. Ed. by M. Griebel and M. A. Schweitzer. Vol. 89. Lecture Notes in Computational Science and Engineering. Springer, 2013, pp. 1–23. DOI: 10.1007/978-3-642-32979-1_1 (cited on p. 60).

[230] H. J. Limbach, A. Arnold, B. A. Mann, and C. Holm. "ESPResSo—An Extensible Simulation Package for Research on Soft Matter Systems". In: *Comp. Phys. Comm.* 174.9 (May 2006), pp. 704–727. DOI: 10.1016/j.cpc.2005.10.005 (cited on p. 60).

[231] F. Höfling and T. Franosch. "Anomalous transport in the crowded world of biological cells". In: *Reports on Progress in Physics* 76.4 (Apr. 1, 2013), p. 046602. ISSN: 0034-4885, 1361-6633. DOI: 10.1088/0034-4885/76/4/046602 (cited on p. 60).

[232] A. Ullrich, M. A. Böhme, J. Schöneberg, H. Depner, S. J. Sigrist, and F. Noé. "Dynamical Organization of Syntaxin-1A at the Presynaptic Active Zone". In: *PLOS Computational Biology* 11.9 (Sept. 14, 2015). Ed. by K. T. Blackwell, e1004407. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1004407 (cited on p. 60).

[233] H. C. R. Klein and U. S. Schwarz. "Studying protein assembly with reversible Brownian dynamics of patchy particles". In: *The Journal of Chemical Physics* 140.18 (May 14, 2014), p. 184112. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.4873708 (cited on p. 60).

[234] M. Sadeghi, T. R. Weikl, and F. Noé. "Particle-based membrane model for mesoscopic simulation of cellular dynamics". In: *The Journal of Chemical Physics* 148.4 (Jan. 28, 2018), p. 044901. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.5009107 (cited on pp. 60, 80).

[235] Y. Posor et al. "Spatiotemporal control of endocytosis by phosphatidylinositol-3,4-bisphosphate". In: *Nature* 499.7457 (July 2013), pp. 233–237. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature12360 (cited on p. 60).

[236] J. Schöneberg, M. Lehmann, A. Ullrich, Y. Posor, W.-T. Lo, G. Lichtner, J. Schmoranzer, V. Haucke, and F. Noé. "Lipid-mediated PX-BAR domain recruitment couples local membrane constriction to endocytic vesicle fission". In: *Nature Communications* 8.1 (Aug. 2017), p. 15873. ISSN: 2041-1723. DOI: 10.1038/ncomms15873 (cited on p. 60).

[237] F. Nedelec and D. Foethke. "Collective Langevin dynamics of flexible cytoskeletal fibers". In: *New Journal of Physics* 9.11 (Nov. 2007), pp. 427–427. DOI: 10.1088/1367-2630/9/11/427 (cited on p. 60).

[238] L. Sbailò and F. Noé. “An efficient multi-scale Green’s function reaction dynamics scheme”. In: *The Journal of Chemical Physics* 147.18 (Nov. 14, 2017), p. 184106. ISSN: 0021-9606, 1089-7690. DOI: `10.1063/1.5010190` (cited on pp. 60, 80).

[239] A. Donev, V. V. Bulatov, T. Oppelstrup, G. H. Gilmer, B. Sadigh, and M. H. Kalos. “A First-Passage Kinetic Monte Carlo algorithm for complex diffusion–reaction systems”. In: *Journal of Computational Physics* 229.9 (May 2010), pp. 3214–3236. ISSN: 00219991. DOI: `10.1016/j.jcp.2009.12.038` (cited on p. 60).

[240] A. Vijaykumar, P. G. Bolhuis, and P. R. ten Wolde. “Combining molecular dynamics with mesoscopic Green’s function reaction dynamics simulations”. In: *The Journal of Chemical Physics* 143.21 (Dec. 7, 2015), p. 214102. ISSN: 0021-9606, 1089-7690. DOI: `10.1063/1.4936254` (cited on p. 60).

[241] A. Vijaykumar, T. E. Ouldridge, P. R. ten Wolde, and P. G. Bolhuis. “Multiscale simulations of anisotropic particles combining molecular dynamics and Green’s function reaction dynamics”. In: *The Journal of Chemical Physics* 146.11 (Mar. 21, 2017), p. 114106. ISSN: 0021-9606, 1089-7690. DOI: `10.1063/1.4977515` (cited on pp. 60, 63).

[242] J. Biedermann, A. Ullrich, J. Schöneberg, and F. Noé. “ReaDDyMM: Fast Interacting Particle Reaction-Diffusion Simulations Using Graphical Processing Units”. In: *Biophysical Journal* 108.3 (Feb. 2015), pp. 457–461. ISSN: 00063495. DOI: `10.1016/j.bpj.2014.12.025` (cited on p. 60).

[243] C. Fröhner and F. Noé. “Reversible Interacting-Particle Reaction Dynamics”. In: *The Journal of Physical Chemistry B* 122.49 (Dec. 13, 2018), pp. 11240–11250. ISSN: 1520-6106, 1520-5207. DOI: `10.1021/acs.jpcb.8b06981` (cited on pp. 61, 65, 72, 73, 82).

[244] M. von Smoluchowski. “Zur kinetischen Theorie der Brownschen Molekularbewegung und der Suspensionen”. In: *Annalen der Physik* 326.14 (1906), pp. 756–780. ISSN: 00033804, 15213889. DOI: `10.1002/andp.19063261405` (cited on p. 63).

[245] A. Einstein. “Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen”. In: *Annalen der Physik* 322.8 (1905), pp. 549–560. ISSN: 00033804, 15213889. DOI: `10.1002/andp.19053220806` (cited on p. 63).

[246] T. Munk, F. Höfling, E. Frey, and T. Franosch. “Effective Perrin theory for the anisotropic diffusion of a strongly hindered rod”. In: *EPL (Europhysics Letters)* 85.3 (Feb. 2009), p. 30003. ISSN: 0295-5075, 1286-4854. DOI: `10.1209/0295-5075/85/30003` (cited on p. 63).

[247] D. L. Ermak and J. A. McCammon. “Brownian dynamics with hydrodynamic interactions”. In: *The Journal of Chemical Physics* 69.4 (Aug. 15, 1978), pp. 1352–1360. ISSN: 0021-9606, 1089-7690. DOI: `10.1063/1.436761` (cited on pp. 63, 80).

[248] W. Humphrey, A. Dalke, and K. Schulten. “VMD: visual molecular dynamics”. In: *Journal of molecular graphics* 14.1 (1996), pp. 33–38. DOI: `10.1016/0263-7855(96)00018-5` (cited on p. 68).

[249] M. Dibak, C. Fröhner, F. Noé, and F. Höfling. “Diffusion-influenced reaction rates in the presence of pair interactions”. In: *The Journal of Chemical Physics* 151.16 (Oct. 28, 2019), p. 164105. ISSN: 0021-9606, 1089-7690. DOI: `10.1063/1.5124728` (cited on pp. 73, 78, 79).

[250] H. Lowen and G. Szamel. “Long-time self-diffusion coefficient in colloidal suspensions: theory versus simulation”. In: *Journal of Physics: Condensed Matter* 5.15 (1993), p. 2295. DOI: `10.1088/0953-8984/5/15/003` (cited on p. 74).

[251] H. Yukawa. “On the Interaction of Elementary Particles. I”. In: *Progress of Theoretical Physics Supplement* 1 (Jan. 1955), pp. 1–10. ISSN: 0375-9687. DOI: 10.1143/PTPS.1.1 (cited on p. 74).

[252] P. H. Colberg and F. Höfling. “Highly accelerated simulations of glassy dynamics using GPUs: Caveats on limited floating-point precision”. In: *Computer Physics Communications* 182.5 (2011), pp. 1120–1129. DOI: 10.1016/j.cpc.2011.01.009 (cited on p. 75).

[253] J. K. Johnson, J. A. Zollweg, and K. E. Gubbins. “The Lennard-Jones equation of state revisited”. In: *Molecular Physics* 78.3 (1993), pp. 591–618. DOI: 10.1080/00268979300100411 (cited on p. 75).

[254] A. Ayadim, M. Oettel, and S. Amokrane. “Optimum free energy in the reference functional approach for the integral equations theory”. In: *Journal of Physics: Condensed Matter* 21.11 (2009), p. 115103. DOI: 10.1088/0953-8984/21/11/115103 (cited on p. 75).

[255] M. P. Allen and D. J. Tildesley. *Computer simulation of liquids*. Oxford, England: Oxford university press, 2017. ISBN: 9780198803195. DOI: 10.1093/oso/9780198803195.001.0001 (cited on p. 75).

[256] W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson. “A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters”. In: *The Journal of chemical physics* 76.1 (1982), pp. 637–649. DOI: 10.1063/1.442716 (cited on p. 75).

[257] M. Rubinstein and R. H. Colby. *Polymer physics*. Vol. 23. New York, NY: Oxford university press, 2003. ISBN: 978-0-19-852059-7 (cited on p. 76).

[258] P. Mereghetti, M. Martinez, and R. C. Wade. “Long range Debye-Hückel correction for computation of grid-based electrostatic forces between biomacromolecules”. In: *BMC biophysics* 7.1 (2014), pp. 1–11. DOI: 10.1186/2046-1682-7-4 (cited on p. 78).

[259] N. Plattner, S. Doerr, G. De Fabritiis, and F. Noé. “Complete protein–protein association kinetics in atomic detail revealed by molecular dynamics simulations and Markov modelling”. In: *Nature chemistry* 9.10 (2017), pp. 1005–1011. DOI: 10.1038/nchem.2785 (cited on p. 78).

[260] M. T. Record, C. F. Anderson, and T. M. Lohman. “Thermodynamic analysis of ion effects on the binding and conformational equilibria of proteins and nucleic acids: the roles of ion association or release, screening, and ion effects on water activity”. In: *Quarterly reviews of biophysics* 11.2 (1978), pp. 103–178. DOI: 10.1017/S003358350000202X (cited on p. 78).

[261] P. Debye and E. Hückel. “Zur Theorie der Elektrolyte. I. Gefrierpunktserniedrigung und verwandte Erscheinungen”. In: *Phys. Z* 24 (1923), pp. 185–206 (cited on p. 78).

[262] D. Shoup and A. Szabo. “Role of diffusion in ligand binding to macromolecules and cell-bound receptors”. In: *Biophysical Journal* 40.1 (1982), pp. 33–39. DOI: 10.1016/S0006-3495(82)84455-X (cited on p. 78).

[263] A. Szabo, K. Schulten, and Z. Schulten. “First passage time approach to diffusion controlled reactions”. In: *The Journal of chemical physics* 72.8 (1980), pp. 4350–4357. DOI: 10.1063/1.439715 (cited on p. 78).

[264] R. F. Grote and J. T. Hynes. “The stable states picture of chemical reactions. II. Rate constants for condensed and gas phase reaction models”. In: *The Journal of Chemical Physics* 73.6 (1980), pp. 2715–2732. DOI: 10.1063/1.440485 (cited on p. 78).

[265] S. H. Northrup and J. T. Hynes. “Short range caging effects for reactions in solution. I. Reaction rate constants and short range caging picture”. In: *The Journal of Chemical Physics* 71.2 (1979), pp. 871–883. DOI: `10.1063/1.438378` (cited on p. 78).

[266] P. W. Atkins, J. De Paula, and J. Keeler. *Atkins’ Physical Chemistry.* Oxford, England: Oxford University Press, 2018. ISBN: 9780198769866 (cited on p. 79).

[267] E. Schneck, F. Sedlmeier, and R. R. Netz. “Hydration repulsion between biomembranes results from an interplay of dehydration and depolarization”. In: *Proceedings of the National Academy of Sciences* 109.36 (2012), pp. 14405–14409. DOI: `10.1073/pnas.1205811109` (cited on p. 79).

[268] B. Halle and M. Davidovic. “Biomolecular hydration: from water dynamics to hydrodynamics”. In: *Proceedings of the National Academy of Sciences* 100.21 (2003), pp. 12135–12140. DOI: `10.1073/pnas.2033320100` (cited on p. 79).

[269] M. Lewis, G. Chang, N. C. Horton, M. A. Kercher, H. C. Pace, M. A. Schumacher, R. G. Brennan, and P. Lu. “Crystal Structure of the Lactose Operon Repressor and Its Complexes with DNA and Inducer”. In: *Science* 271.5253 (Mar. 1996), pp. 1247–1254. DOI: `10.1126/science.271.5253.1247` (cited on p. 82).

[270] N. Yildirim and M. C. Mackey. “Feedback Regulation in the Lactose Operon: A Mathematical Modeling Study and Comparison with Experimental Data”. In: *Biophysical Journal* 84.5 (May 2003), pp. 2841–2851. ISSN: 00063495. DOI: `10.1016/S0006-3495(03)70013-7` (cited on p. 82).

[271] T. Kuhlman, Z. Zhang, M. H. Saier, and T. Hwa. “Combinatorial transcriptional control of the lactose operon of Escherichia coli”. In: *Proceedings of the National Academy of Sciences* 104.14 (Apr. 2007), pp. 6043–6048. DOI: `10.1073/pnas.0606717104` (cited on p. 82).

[272] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon. “Network motifs in the transcriptional regulation network of Escherichia coli”. In: *Nature Genetics* 31.1 (May 2002), pp. 64–68. ISSN: 1061-4036. DOI: `10.1038/ng881` (cited on p. 82).

[273] S. Gama-Castro et al. “RegulonDB version 9.0: high-level integration of gene regulation, coexpression, motif clustering and beyond”. In: *Nucleic Acids Research* 44.D1 (Jan. 2016), pp. D133–D143. ISSN: 0305-1048. DOI: `10.1093/nar/gkv1156` (cited on p. 82).

[274] T. I. Lee. “Transcriptional Regulatory Networks in Saccharomyces cerevisiae”. In: *Science* 298.5594 (Oct. 2002), pp. 799–804. ISSN: 00368075. DOI: `10.1126/science.1075090` (cited on p. 82).

[275] R. Roa, W. K. Kim, M. Kanduč, J. Dzubiella, and S. Angioletti-Uberti. “Catalyzed Bimolecular Reactions in Responsive Nanoreactors”. In: *ACS Catalysis* 7.9 (Sept. 2017), pp. 5604–5611. ISSN: 2155-5435. DOI: `10.1021/acscatal.7b01701` (cited on p. 82).

[276] P. Cong, R. D. Doolen, Q. Fan, D. M. Giaquinta, S. Guan, E. W. McFarland, D. M. Poojary, K. Self, H. W. Turner, and W. H. Weinberg. “High-Throughput Synthesis and Screening of Combinatorial Heterogeneous Catalyst Libraries”. In: *Angewandte Chemie International Edition* 38.4 (1999), pp. 483–488. ISSN: 1521-3773. DOI: `10.1002/(SICI)1521-3773(19990215)38:4<483::AID-ANIE483>3.0.CO;2-%23` (cited on p. 82).

[277] L. Kiwi-Minsker and A. Renken. “Microstructured reactors for catalytic reactions”. In: *Catalysis today* 110.1-2 (2005), pp. 2–14. DOI: `10.1016/j.cattod.2005.09.011` (cited on p. 82).

[278] S. Winkelmann and C. Schütte. “The spatiotemporal master equation: Approximation of reaction-diffusion dynamics via Markov state modeling”. In: *The Journal of Chemical Physics* 145.21 (2016), p. 214107. DOI: `10.1063/1.4971163` (cited on p. 82).

[279] S. S. Andrews. “Smoldyn: particle-based simulation with rule-based modeling, improved molecular interaction and a library interface”. In: *Bioinformatics* 33.5 (2017), pp. 710–717. DOI: 10.1093/bioinformatics/btw700 (cited on p. 82).

[280] J. Schöneberg, A. Ullrich, and F. Noé. “Simulation tools for particle-based reaction-diffusion dynamics in continuous space”. In: *BMC Biophys* 7.1 (2014), pp. 1–10. DOI: 10.1186/s13628-014-0011-5 (cited on p. 82).

[281] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Dalla Favera, and A. Califano. “ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context”. In: *BMC bioinformatics*. Vol. 7. 1. BioMed Central. 2006, pp. 1–15. DOI: 10.1186/1471-2105-7-S1-S7 (cited on p. 82).

[282] T. Tian, S. Xu, J. Gao, and K. Burrage. “Simulated maximum likelihood method for estimating kinetic rates in gene expression”. In: *Bioinformatics* 23.1 (2007), pp. 84–91. DOI: 10.1093/bioinformatics/btl552 (cited on p. 82).

[283] Y. Pantazis and I. Tsamardinos. “A unified approach for sparse dynamical system inference from temporal measurements”. In: *Bioinformatics* 35.18 (2019), pp. 3387–3396. DOI: 10.1093/bioinformatics/btz065 (cited on p. 82).

[284] W. Pan, Y. Yuan, J. Goncalves, and G.-b. Stan. “Reconstruction of arbitrary biochemical reaction networks: A compressive sensing approach”. In: *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, Dec. 2012, pp. 2334–2339. ISBN: 978-1-4673-2066-5. DOI: 10.1109/CDC.2012.6426216 (cited on p. 82).

[285] H. Zou and T. Hastie. “Regularization and variable selection via the elastic net”. In: *Journal of the royal statistical society: series B (statistical methodology)* 67.2 (2005), pp. 301–320. DOI: 10.1111/j.1467-9868.2005.00503.x (cited on p. 84).

[286] R. Tibshirani. “Regression selection and shrinkage via the lasso”. In: *Journal of the Royal Statistical Society Series B* 58.1 (1996), pp. 267–288. DOI: 10.1111/j.2517-6161.1996.tb02080.x (cited on p. 84).

[287] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY: Springer New York, 2009. ISBN: 978-0-387-84858-7. DOI: 10.1007/978-0-387-84858-7_2 (cited on p. 84).

[288] A. E. Hoerl and R. W. Kennard. “Ridge Regression: Biased Estimation for Nonorthogonal Problems”. In: *Technometrics* 12 (1970), pp. 55–67. DOI: 10.2307/1271436 (cited on p. 84).

[289] M. Thattai and A. van Oudenaarden. “Intrinsic noise in gene regulatory networks”. In: *Proceedings of the National Academy of Sciences* 98.15 (2001), pp. 8614–8619. ISSN: 0027-8424. DOI: 10.1073/pnas.151588598 (cited on p. 84).

[290] I. Golding, J. Paulsson, S. M. Zawilski, and E. C. Cox. “Real-time kinetics of gene activity in individual bacteria”. In: *Cell* 123.6 (2005), pp. 1025–1036. ISSN: 00928674. DOI: 10.1016/j.cell.2005.09.031 (cited on p. 86).

[291] O. G. Berg. “A model for the statistical fluctuations of proteins numbers in a microbial population”. In: *J. Theor. Biol.* 71.4 (1978), pp. 587–603. DOI: 10.1016/0022-5193(78)90326-0 (cited on p. 86).

[292] M. B. Elowitz. “Stochastic Gene Expression in a Single Cell”. In: *Science* 297.5584 (Aug. 2002), pp. 1183–1186. ISSN: 00368075. DOI: 10.1126/science.1070919 (cited on p. 86).

[293] W. Kolch, M. Calder, and D. Gilbert. “When kinases meet mathematics: the systems biology of MAPK signalling”. In: *FEBS letters* 579.8 (2005), pp. 1891–1895. DOI: `10.1016/j.febslet.2005.02.002` (cited on p. 90).

[294] R. J. Orton, O. E. Sturm, V. Vyshemirsky, M. Calder, D. R. Gilbert, and W. Kolch. “Computational modelling of the receptor-tyrosine-kinase-activated MAPK pathway”. In: *Biochemical Journal* 392.2 (2005), pp. 249–261. DOI: `10.1042/BJ20050908` (cited on p. 90).

[295] J. Iqbal and M. Zaidi. “TNF-induced MAP kinase activation oscillates in time”. In: *Biochemical and biophysical research communications* 371.4 (2008), pp. 906–911. DOI: `10.1016/j.bbrc.2008.03.113` (cited on p. 93).

[296] S.-Y. Shin, O. Rath, S.-M. Choo, F. Fee, B. McFerran, W. Kolch, and K.-H. Cho. “Positive- and negative-feedback regulations coordinate the dynamic behavior of the Ras-Raf-MEK-ERK signal transduction pathway”. In: *Journal of cell science* 122.3 (2009), pp. 425–435. DOI: `10.1242/jcs.036319` (cited on p. 93).

Appendix A

Simulation setups for ReaDDy2

A.1 Lotka–Volterra example

We present the unitless parameters used for the creation of Figure 2.6 in the theory Section 2.4. We define three species:

1. “A”: Prey particles. Diffusion constant $D_\mathrm{A} = 0.01$,

2. “B”: Predator particles. Diffusion constant $D_\mathrm{B} = 0.01$,

3. “W”: Spatial barrier particles. Diffusion constant $D_\mathrm{W} = 0$.

The *microscopic* reaction rates and radii of system (2.26) are chosen as

$\alpha = 2$ (prey birth rate),	$r_\alpha = 0.85$ (distance of spawned particles),
$\beta = 7.67$ (predator consume rate),	$r_\beta = 0.25$ (reaction radius),
$\gamma = 1.5$ (predator decay rate),	
$\lambda = \mu = 0.39$ (social friction rate),	$r_\lambda = r_\mu = 0.2$ (friction reaction radius).

Pairs of prey and wall or predator and wall particles are subject to a harmonic repulsion potential with cutoff radius $r = 1$ and force constant $k = 150$. The simulation box is equipped with harmonically repelling walls under the same force constant of $k = 150$. We integrate with a time step of $\tau = 10^{-3}$.

A.2 Density-dependent reaction kinetics

This table contains parameters for the Debye–Hückel system in the results Section 4.4.5.

Quantity	Symbol	Value	Unit
Thermal energy	k_BT	2.49	$\mathrm{kJ\,mol^{-1}}$
Volume	V	100^3	$\mathrm{nm^3}$
Radius A	r_A	1	nm
Radius B	r_B	0.8	nm
Radius C	r_C	1	nm
Diffusion coeff. A	D_A	0.01	$\mathrm{nm^2\,ns^{-1}}$
Diffusion coeff. B	D_B	0.0125	$\mathrm{nm^2\,ns^{-1}}$
Diffusion coeff. C	D_C	0.01	$\mathrm{nm^2\,ns^{-1}}$
Charge A	q_A	1.3	–
Charge B	q_B	-1	–
Charge C	q_C	0	–
Screening parameter	κ	3.82	$\mathrm{nm^{-1}}$
Debye-Hückel prefactor	$e^2\varepsilon_0^{-1}\varepsilon_r^{-1}$	2349	$\mathrm{kJ\,nm\,mol^{-1}}$
Repulsion energy	U_r	1.	$\mathrm{kJ\,mol^{-1}}$
Cutoff radius	r_cutoff	4.7	nm
Reaction radius	R	2.	nm
Equilibrium constant	K_dilute	6.16×10^{-5}	$\mathrm{nm^{-3}}$
Macroscopic rate constant	k_on	0.11	$\mathrm{nm^3\,ns^{-1}}$
Macroscopic rate constant	k_off	6.58×10^{-6}	$\mathrm{ns^{-1}}$
Microscopic rate constant	λ_on	5.61×10^{-3}	$\mathrm{ns^{-1}}$
Microscopic rate constant	λ_off	6.58×10^{-6}	$\mathrm{ns^{-1}}$
Timestep	τ	0.1	ns

A.3 Living polymers

This table contains parameters for the living polymers system in the results Section 4.4.4.

Quantity	Symbol	Value	Unit
Thermal energy	k_BT	2.437	$\mathrm{kJ\,mol^{-1}}$
Diffusion coeff.	D	2^{-3}	$\mathrm{nm^2\,ns^{-1}}$
Volume	V	100^3	$\mathrm{nm^3}$
Bond length	σ	1	nm
Bond formation rate	k_F	1	$\mathrm{ns^{-1}}$
Bond formation radius	r_F	1	nm
Bond dissociation rate	k_B	$5N\cdot10^{-6}$	$\mathrm{ns^{-1}}$
Time step	τ	1	ns

The dissociation reaction is implemented as follows.

```
import numpy as np
import readdy

def dissociation_rate_function(topology):
    edges = topology.graph.edges
    return .000005 * float(len(edges)) if len(edges) > 2 else 0.
```

```
def dissociation_reaction_function(topology):
    recipe = readdy.StructuralReactionRecipe(topology)
    edges = topology.graph.edges
    vertices = topology.graph.vertices

    # at least a structure like
    # v1 -- v2 -- v3 -- v4
    if len(edges) > 2:
        # find the end particles
        counts = defaultdict(int)
        for (eix1, eix2) in edges:
            counts[topology.particles[eix1]] += 1
            counts[topology.particles[eix2]] += 1

    # the end particles are the ones that have exactly one edge
    endpoints = []
    for pix in counts.keys():
        if counts[pix] == 1:
        endpoints.append(pix)

    # randomly draw an edge excluding the edges leading to the both ends
    edge_index = np.random.randint(0, len(edges) - 2)
    removed_edge = None
    # for each edge in the graph
    for (eix1, eix2) in edges:
        pix1 = topology.particles[eix1]
        pix2 = topology.particles[eix2]
        # check if it belongs to one of the end vertices
        if pix1 not in endpoints and pix2 not in endpoints:
        # if not, reduce edge_index until 0
        if edge_index == 0:
            # remove this edge
            recipe.remove_edge(e[0], e[1])
            removed_edge = e
            break
        else:
            edge_index -= 1

    pix1 = topology.particles[removed_edge[0]]
    pix2 = topology.particles[removed_edge[1]]

    # Set the correct particle types for new topologies
    recipe.change_particle_type([vix for vix, v in enumerate(vertices)
                                 if v.particle_index == pix1][0], "Head")
    recipe.change_particle_type([vix for vix, v in enumerate(vertices)
                                 if v.particle_index == pix2][0], "Head")

    return recipe
```

Appendix B

Ansatz libraries for reactive SINDy studies

Here we list all the used ansatz libraries Θ that were used in the “reactive SINDy” chapter (Chapter 5).

B.1 Gene-regulatory network ansatz

Full set of ansatz reactions Θ used in Section 5.3.1 for the gene-regulatory network. The given rate constants define the ground truth reaction model.

Reaction			rate	description
$\mathrm{DNA_A}$	$\rightharpoonup$	$\mathrm{DNA_A} + \mathrm{mRNA_A}$	$k_1 = 1.8$	transcription of $\mathrm{mRNA_A}$
$\mathrm{mRNA_A}$	$\rightharpoonup$	$\mathrm{mRNA_A} + \mathrm{A}$	$k_2 = 2.1$	translation of A proteins
$\mathrm{mRNA_A}$	$\rightharpoonup$	$\emptyset$	$k_3 = 1.3$	$\mathrm{mRNA_A}$ decay
A	$\rightharpoonup$	$\emptyset$	$k_4 = 1.5$	decay of A proteins
$\mathrm{DNA_B}$	$\rightharpoonup$	$\mathrm{DNA_B} + \mathrm{mRNA_B}$	$k_5 = 2.2$	transcription of $\mathrm{mRNA_B}$
$\mathrm{mRNA_B}$	$\rightharpoonup$	$\mathrm{mRNA_B} + \mathrm{B}$	$k_6 = 2.0$	translation of B proteins
$\mathrm{mRNA_B}$	$\rightharpoonup$	$\emptyset$	$k_7 = 2.0$	$\mathrm{mRNA_B}$ decay
B	$\rightharpoonup$	$\emptyset$	$k_8 = 2.5$	decay of B proteins
$\mathrm{DNA_C}$	$\rightharpoonup$	$\mathrm{DNA_C} + \mathrm{mRNA_C}$	$k_9 = 3.2$	transcription of $\mathrm{mRNA_C}$
$\mathrm{mRNA_C}$	$\rightharpoonup$	$\mathrm{mRNA_C} + \mathrm{C}$	$k_{10} = 3.0$	translation of C proteins
$\mathrm{mRNA_C}$	$\rightharpoonup$	$\emptyset$	$k_{11} = 2.3$	$\mathrm{mRNA_C}$ decay
C	$\rightharpoonup$	$\emptyset$	$k_{12} = 2.5$	decay of C proteins
$\mathrm{mRNA_A} + \mathrm{A}$	$\rightharpoonup$	A	$k_{13} = 0$	self regulation of A proteins
$\mathrm{mRNA_B} + \mathrm{B}$	$\rightharpoonup$	B	$k_{14} = 0$	self regulation of B proteins
$\mathrm{mRNA_C} + \mathrm{C}$	$\rightharpoonup$	C	$k_{15} = 0$	self regulation of C proteins
$\mathrm{mRNA_B} + \mathrm{A}$	$\rightharpoonup$	A	$k_{16} = 0$	regulation of $\mathrm{mRNA_B}$
$\mathrm{mRNA_C} + \mathrm{B}$	$\rightharpoonup$	B	$k_{17} = 0$	regulation of $\mathrm{mRNA_C}$
$\mathrm{mRNA_A} + \mathrm{C}$	$\rightharpoonup$	C	$k_{18} = 0$	regulation of $\mathrm{mRNA_A}$
$\mathrm{mRNA_C} + \mathrm{A}$	$\rightharpoonup$	A	$k_{16} = 6.0$	regulation of $\mathrm{mRNA_C}$
$\mathrm{mRNA_B} + \mathrm{C}$	$\rightharpoonup$	C	$k_{17} = 4.0$	regulation of $\mathrm{mRNA_B}$
$\mathrm{mRNA_A} + \mathrm{B}$	$\rightharpoonup$	B	$k_{18} = 3.0$	regulation of $\mathrm{mRNA_A}$
$\mathrm{mRNA_A} + \mathrm{A}$	$\rightharpoonup$	$\mathrm{mRNA_A}$	$k_{19} = 0$	artificial fusion
$\mathrm{mRNA_B} + \mathrm{B}$	$\rightharpoonup$	$\mathrm{mRNA_B}$	$k_{20} = 0$	artificial fusion
$\mathrm{mRNA_A} + \mathrm{B}$	$\rightharpoonup$	$\mathrm{mRNA_A}$	$k_{21} = 0$	artificial fusion
$\mathrm{mRNA_B} + \mathrm{C}$	$\rightharpoonup$	$\mathrm{mRNA_B}$	$k_{22} = 0$	artificial fusion
$\mathrm{mRNA_C} + \mathrm{A}$	$\rightharpoonup$	$\mathrm{mRNA_C}$	$k_{23} = 0$	artificial fusion
$\mathrm{mRNA_A} + \mathrm{C}$	$\rightharpoonup$	$\mathrm{mRNA_A}$	$k_{24} = 0$	artificial fusion
$\mathrm{mRNA_B} + \mathrm{A}$	$\rightharpoonup$	$\mathrm{mRNA_B}$	$k_{25} = 0$	artificial fusion
$\mathrm{A} + \mathrm{A}$	$\rightharpoonup$	A	$k_{26} = 0$	A regulates A
$\mathrm{B} + \mathrm{B}$	$\rightharpoonup$	B	$k_{27} = 0$	B regulates B
$\mathrm{C} + \mathrm{C}$	$\rightharpoonup$	C	$k_{28} = 0$	C regulates C
$\mathrm{B} + \mathrm{A}$	$\rightharpoonup$	A	$k_{29} = 0$	artificial fusion
$\mathrm{C} + \mathrm{B}$	$\rightharpoonup$	B	$k_{30} = 0$	artificial fusion
$\mathrm{A} + \mathrm{C}$	$\rightharpoonup$	C	$k_{31} = 0$	artificial fusion
$\mathrm{C} + \mathrm{A}$	$\rightharpoonup$	A	$k_{32} = 0$	artificial fusion
$\mathrm{B} + \mathrm{C}$	$\rightharpoonup$	C	$k_{33} = 0$	artificial fusion
$\mathrm{A} + \mathrm{B}$	$\rightharpoonup$	B	$k_{34} = 0$	artificial fusion
A	$\rightharpoonup$	B	$k_{35} = 0$	artificial conversion
B	$\rightharpoonup$	C	$k_{36} = 0$	artificial conversion
C	$\rightharpoonup$	A	$k_{37} = 0$	artificial conversion
A	$\rightharpoonup$	C	$k_{38} = 0$	artificial conversion
C	$\rightharpoonup$	B	$k_{39} = 0$	artificial conversion
B	$\rightharpoonup$	A	$k_{40} = 0$	artificial conversion
$\mathrm{mRNA_B} + \mathrm{mRNA_C}$	$\rightharpoonup$	$\mathrm{mRNA_A}$	$k_{41} = 0$	artificial fusion
$\mathrm{mRNA_C} + \mathrm{mRNA_B}$	$\rightharpoonup$	$\mathrm{mRNA_C}$	$k_{42} = 0$	artificial fusion
$\mathrm{mRNA_C} + \mathrm{A}$	$\rightharpoonup$	C	$k_{43} = 0$	artificial fusion

B.2 MAPK ansatz

Full set of ansatz reactions Θ used in Section 5.3.2 for the MAPK system. The given rate constants define the ground truth reaction model.

Reaction			rate	description
S + MAPKKK	$\rightharpoonup$	S + MAPKKK*	$k_1 = 1$	external stimulus activates MAPKKK
MAPKKK*	$\rightharpoonup$	MAPKKK	$k_2 = 1$	dephosphorylation
MAPKKK* + MAPKK	$\rightharpoonup$	MAPKKK* + MAPKK*	$k_3 = 1$	phosphorylation of MAPKK
MAPKK*	$\rightharpoonup$	MAPKK	$k_4 = 1$	dephosphorylation
MAPKK* + MAPK	$\rightharpoonup$	MAPKK* + MAPK*	$k_5 = 1$	phosphorylation of MAPK
MAPK*	$\rightharpoonup$	MAPK	$k_6 = 1$	dephosphorylation
MAPK* + TF	$\rightharpoonup$	MAPK* + TF*	$k_7 = 1$	phosphorylation of transcription factor
TF*	$\rightharpoonup$	TF	$k_8 = 1$	dephosphorylation
MAPKKK + MAPKK	$\rightharpoonup$	MAPKKK + MAPKK*	$k_9 = 0$	artificial reaction
MAPKKK + MAPK	$\rightharpoonup$	MAPK*	$k_{10} = 0$	artificial reaction
MAPKKK + TF	$\rightharpoonup$	MAPKKK + TF*	$k_{11} = 0$	artificial reaction
MAPKKK* + MAPK	$\rightharpoonup$	MAPKKK* + MAPK*	$k_{12} = 0$	artificial reaction
MAPKKK* + TF	$\rightharpoonup$	MAPKKK* + TF*	$k_{13} = 0$	artificial reaction
MAPKK + TF	$\rightharpoonup$	MAPKK + TF*	$k_{14} = 0$	artificial reaction
MAPKK* + TF	$\rightharpoonup$	MAPKK* + TF*	$k_{15} = 0$	artificial reaction
MAPK + TF	$\rightharpoonup$	MAPK + TF*	$k_{16} = 0$	artificial reaction
MAPKK + MAPK	$\rightharpoonup$	MAPKK + MAPK*	$k_{17} = 0$	artificial reaction
MAPKKK + MAPKK*	$\rightharpoonup$	MAPKKK + MAPKK	$k_{18} = 0$	artificial reaction
MAPKKK + MAPK*	$\rightharpoonup$	MAPKKK + MAPK	$k_{19} = 0$	artificial reaction
MAPKKK + TF*	$\rightharpoonup$	MAPKKK + TF	$k_{20} = 0$	artificial reaction
MAPKKK* + MAPKK*	$\rightharpoonup$	MAPKKK* + MAPKK	$k_{21} = 0$	artificial reaction
MAPKKK* + MAPK*	$\rightharpoonup$	MAPKKK* + MAPK	$k_{22} = 0$	artificial reaction
MAPKKK* + TF*	$\rightharpoonup$	MAPKKK* + TF	$k_{23} = 0$	artificial reaction
MAPKK + MAPK*	$\rightharpoonup$	MAPKK + MAPK	$k_{24} = 0$	artificial reaction
MAPKK + TF*	$\rightharpoonup$	MAPKK + TF	$k_{25} = 0$	artificial reaction
MAPKK* + MAPK*	$\rightharpoonup$	MAPKK* + MAPK	$k_{26} = 0$	artificial reaction
MAPKK* + TF*	$\rightharpoonup$	MAPKK* + TF	$k_{27} = 0$	artificial reaction
MAPK + TF*	$\rightharpoonup$	MAPK + TF	$k_{28} = 0$	artificial reaction
MAPK* + TF*	$\rightharpoonup$	MAPK* + TF	$k_{29} = 0$	artificial reaction

B.3 Lotka–Volterra ansatz

Full set of ansatz reactions Θ used in Section 5.3.3 for the Lotka–Volterra system. The given rate constants define the ground truth reaction model.

Reaction			rate	description
A + A	$\rightharpoonup$	$\emptyset$	$k_1 = 0.1$	social friction of prey
B + B	$\rightharpoonup$	$\emptyset$	$k_2 = 0.1$	social friction of predator
A	$\rightharpoonup$	A + A	$k_3 = 1$	prey growth
A + B	$\rightharpoonup$	B + B	$k_4 = 1$	predator eats prey
B	$\rightharpoonup$	$\emptyset$	$k_5 = 1$	predator decays
A + B	$\rightharpoonup$	A + A	$k_6 = 0$	artificial reaction
A	$\rightharpoonup$	$\emptyset$	$k_7 = 0$	artificial reaction
B + B	$\rightharpoonup$	B	$k_8 = 0$	artificial reaction
B	$\rightharpoonup$	B + B	$k_9 = 0$	artificial reaction
A + A	$\rightharpoonup$	A	$k_{10} = 0$	artificial reaction
A + B	$\rightharpoonup$	A	$k_{11} = 0$	artificial reaction
A + B	$\rightharpoonup$	B	$k_{12} = 0$	artificial reaction
A + A	$\rightharpoonup$	B	$k_{13} = 0$	artificial reaction
A	$\rightharpoonup$	B	$k_{14} = 0$	artificial reaction
B	$\rightharpoonup$	A	$k_{15} = 0$	artificial reaction
A	$\rightharpoonup$	B + B	$k_{16} = 0$	artificial reaction

Appendix C

Definition of an SDE simulator in deeptime

Double well 2D definition. Using `constexpr` specifiers, it is already known at compile-time that, e.g., the system lives in two-dimensional space and should use an Euler–Maruyama integrator.

```
template<typename T>
struct DoubleWell2D {
        using system_type = sde_tag;  // this is an SDE
        // the state-space dimension
        static constexpr std::size_t DIM = 2;
        // data type, e.g., float or double
        using dtype = T;
        // the type of state of the system, here $x_t \in \mathbb{R}^2$
        using State = Vector<T, DIM>;
        // use Euler-Maruyama
        using Integrator = EulerMaruyama<State, DIM>;

        // implementation of $V(x)$, optional
        constexpr dtype energy(const State &x) const {
                return (x[0]*x[0]-1.) * (x[0]*x[0]-1.) + x[1] * x[1];
        }

        // implementation of $-\nabla V(x)$
        constexpr State f(const State &x) const {
                return {{-4 * x[0] * x[0] * x[0] + 4 * x[0], -2 * x[1]}};
        }

        static constexpr Matrix<T, 2> sigma{{{{0.7, 0.0}}, {{0.0, 0.7}}}};
        // the time step to use for integration
        T h{1e-3};
        // the number of steps between each sample
        std::size_t nSteps{10000};
};
```

Acknowledgements

I would like to thank all of the people who have supported and encouraged me in my doctoral research. First and foremost I would like to thank Prof. Frank Noé. He has given me the opportunity to work on exciting problems in an exciting environment and surround myself with great and interesting people. I would also like to thank him for being an excellent supervisor as well as for his personal and scientific support whenever it was needed.

Over the years I had the opportunity to collaborate with and get to know many people in the AI4Science (former CMB) group, all of them were and are absolutely brilliant. With many of them a friendship emerged which hopefully long outlasts my time in the research group.

I would like to especially thank Laura, Tim, Selle, Mauricio, and Andreas K for reading and giving me feedback on the thesis. I am grateful to Mohsen for many interesting discussions and thine singal undefatigable mentoring. I would like to thank Chris F for being a great research companion during various projects as well as Stefan Klus for his support regarding the deeptime project.

In no particular order (`RNG`) I would like to thank my colleagues and friends Simon, Chris W, Martin, Fabian, Carmen, Andreas M, Mohsen, Guille, Mohammad, Shreyas, Katarina, Fabio, Jan, Chris F, Lluís, Jonas, Luca, Maaike, Manuel, Pablo, Hao, Jonas, Esam, Brooke, Aleksander, Mauricio, Andreas K, and Giovanni for fruitful discussions, support, and poking fun at each other.

I would also like to thank my dear friends Felix, Bart, and Alex for their moral support. I am deeply grateful to Laura for supporting me during trying moments and am equally happy that I am able to share the good ones with her. I would like to thank my parents for all their support, without which it would never have been possible for me to embark on this endeavor.

Finally, I would like to thank Deutsche Forschungsgemeinschaft (SFB 958 and SFB 1114) for their financial support.

Selbstständigkeitserklärung

Name: Hoffmann
Vorname: Moritz

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Dissertation selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Dissertation wurde in gleicher oder ähnlicher Form noch in keinem früheren Promotionsverfahren eingereicht.

Mit einer Prüfung meiner Arbeit durch ein Plagiatsprüfungsprogramm erkläre ich mich einverstanden.

Berlin, 02. Juni 2022,

Moritz Hoffmann

Zusammenfassung

Die vorliegende Arbeit präsentiert Methoden und Implementierungen, um Mehrkörpersysteme effizient zu simulieren und die generierten Daten zu analysieren. Wir konzentrieren uns hauptsächlich auf biologische Systeme und Prozesse, welche experimentell bei einer hohen raumzeitlichen Auflösung kaum beobachtbar sind. Computersimulationen hingegen erlauben im Rahmen ihres zugrundeliegenden Modells hohe Auflösungen. Der Fokus dieser Arbeit liegt auf der Simulation von partikelbasierten Modellen, welche große Mengen an hochdimensionalen Zeitseriendaten erzeugen können, wodurch Visualisierung und Analyse von dominanten Prozessen erheblich erschwert sind.

Im ersten Teil der Arbeit stellen wir daher mehrere Ansätze maschinellen Lernens vor, welche das Ziel haben, möglichst niedrigdimensionale Modelle zu schätzen. Diese repräsentieren dominante Prozesse, geometrisch stabile Strukturen oder die zugrundeliegende Dynamik. Die vorgestellten Methoden sind in der hier entwickelten Programmbibliothek `deeptime` enthalten, welche sie, obwohl sie aus verschiedenen Fachrichtungen stammen, einer breiten Nutzerbasis zugänglich macht.

Im zweiten Teil der Arbeit entwickeln wir die partikelbasierte Reaktionsdiffusionssimulationssoftware `ReaDDy2`. Sie arbeitet auf einer mesoskopischen Skala, in der zum Beispiel Proteine als einzelne Partikel oder kleine Partikelkomplexe dargestellt werden können. Wir zeigen, dass `ReaDDy2` auf diesem Auflösungsniveau wesentlich schneller als vergleichbare Softwarepakete arbeitet. Damit ist es, im Gegensatz zu höherauflösenden Methoden wie Moleküldynamik, möglich, Prozesse mit Zeitskalen im Millisekunden- bis Sekundenbereich, etwa Signaltransduktion in Zellumgebungen oder chemische Reaktoren mit komplizierten Geometrien, abzubilden. Eine Verallgemeinerung von chemischen Reaktionen einzelner Partikel auf die Struktur von Partikelkomplexen erlaubt es, dass sich die Komplexe dynamisch verändern können.

Im dritten Teil erarbeiten wir die `reactive SINDy` Methode, welche besonders dünn besetzte Reaktionsnetzwerke im Sinne der klassischen Reaktionskinetik anhand von Konzentrationszeitserien schätzen kann. Eine hierfür kritische Komponente sind über ein Differenzialgleichungssystem gekoppelte Ansatzfunktionen. Daher haben wir `reactive SINDy` als lineare Tensorregression formuliert.

Die eingeführten Methoden werden überwiegend anhand von biologischen Prozessen motiviert und validiert, können aber in einem allgemeineren Kontext eingesetzt werden. Zum Beispiel können dynamische Modelle aus Simulations- und Messdaten der Fluiddynamik geschätzt werden. Des Weiteren kann `ReaDDy2` genutzt werden, um etwa Epidemien oder Populationsdynamik abzubilden. Durch die Formulierung und Implementierung dieser Methoden schaffen wir neue Möglichkeiten, Mehrkörpersysteme zu simulieren und verstehen.