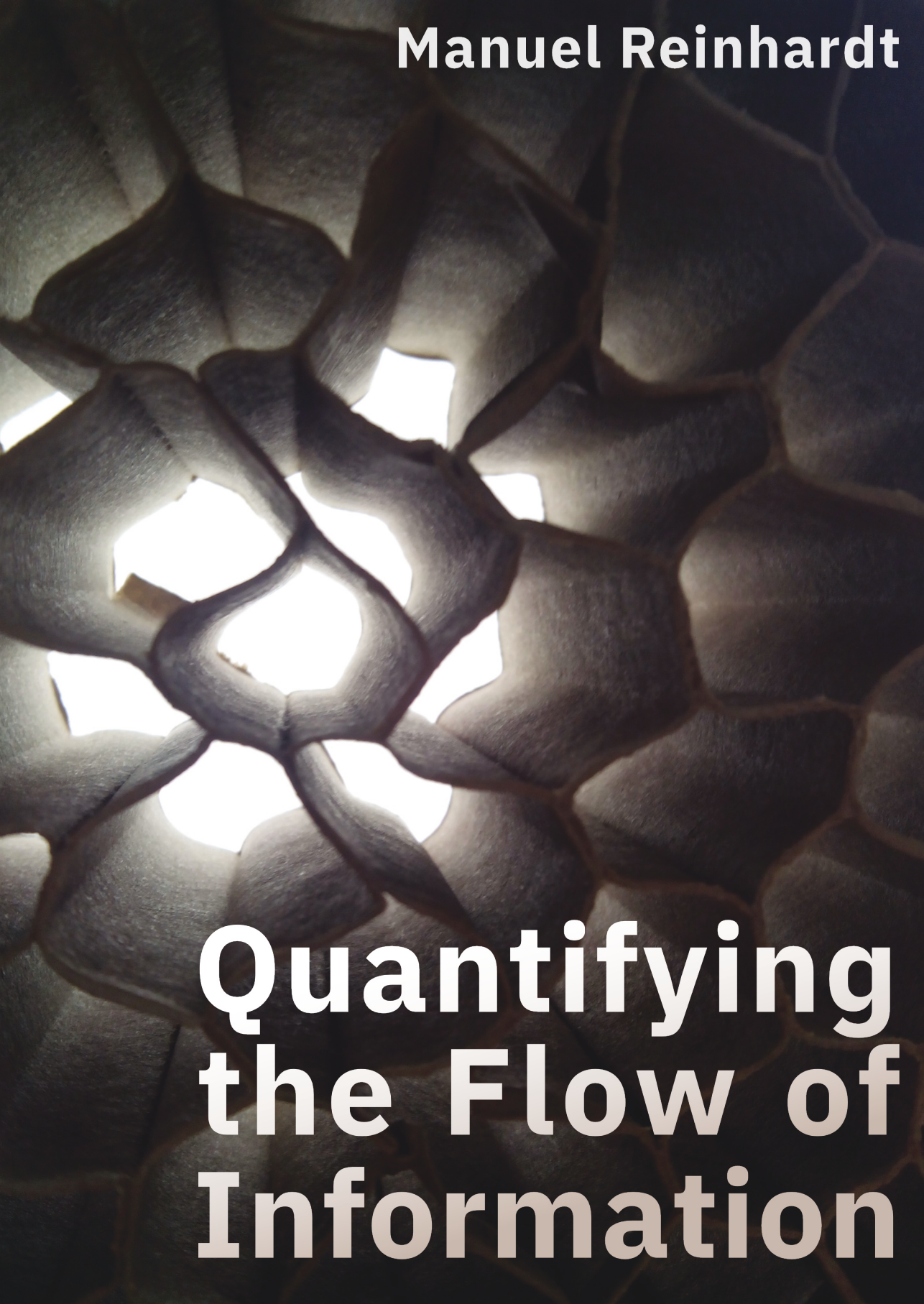

Manuel Reinhardt
Quantifying the Flow of Information

# Quantifying the Flow of Information

**Manuel Reinhardt**

**This dissertation was approved by**

| | |
|---|---|
| *Promoter:* | |
| prof.dr. Pieter Rein ten Wolde | Vrije Universiteit Amsterdam, The Netherlands |
| *Copromoter:* | |
| prof.dr. Gašper Tkačik | Institute of Science and Technology, Austria |

**Dissertation Committee**

| | |
|---|---|
| *Chairman:* | |
| Rector Magnificus, prof.dr. Jeroen J.G. Geurts | Vrije Universiteit Amsterdam, The Netherlands |
| *Committee:* | |
| dr. Greg Stephens | Vrije Universiteit Amsterdam, The Netherlands |
| dr. Clélia de Mulatier | University of Amsterdam, The Netherlands |
| prof.dr. Peter G. Bolhuis | University of Amsterdam, The Netherlands |
| prof.dr. Aleksandra Walczak | Ecole Normale Supérieure, Paris, France |
| prof.dr. Peter Swain | University of Edinburgh, United Kingdom |

*Printed by:* Ipskamp Printing, Enschede

*Cover:* Juana Ciudad Pizarro

ISBN: 978-94-92323-76-7

An electronic version of this thesis is available at http://ub.vu.nl and http://ir.amolf.nl. The DOI of this thesis is 10.5463/thesis.1092. Printed copies can be obtained by request via library@amolf.nl.

VRIJE UNIVERSITEIT

# Quantifying the Flow of Information

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor of Philosohpy aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. J.J.G. Geurts,
volgens besluit van de decaan
van de Faculteit der Bètawetenschappen
in het openbaar te verdedigen
op vrijdag 13 juni 2025 om 9.45 uur
in de universiteit

door

Manuel Johannes Reinhardt

geboren te Ebersberg, Duitsland

| | |
|---|---|
| promotor: | prof.dr. P.R. ten Wolde |
| copromotor: | prof.dr. G. Tkačik |
| promotiecommissie: | dr. G.J. Stephens<br>prof.dr. A.M. Walczak<br>prof.dr. P.S. Swain<br>prof.dr. P.G. Bolhuis<br>dr. C.M.C. de Mulatier |

# Contents

# 1 Introduction

*The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have meaning; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem. The significant aspect is that the actual message is one selected from a set of possible messages. The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design.*

—Claude Shannon, A Mathematical Theory of Communication (1948)

We live in the era of information. Information technology permeates every aspect of modern life, shaping how we communicate, learn, have social interactions, and spend our leisure time. Beyond daily life, information plays a crucial role in fields like physics, biology, neuroscience, and engineering, where it is used to study and enhance the function of complex systems and machines. Quantifying the flow of information within these domains is essential, and, although the concept of information is abstract, its power in explaining the processes that shape our world is profound.

The explanatory power of information stems from the intrinsic link between information and performance. Without a potential reward, or the possibility of avoiding harm, information has no value.[1] As a result, information collection and processing typically serves a clear purpose. For example, a self-driving car processes information from its sensors in order to make decisions about navigation [161]. Similarly, bacteria acquire chemical information about their environment in order to optimize their movement toward nutrients and away from toxins, maximizing their chance of survival [107]. More generally, in evolutionary biology the link between genetic information and fitness is explored [1, 77]. Thus, whether in biological organisms or engineered systems, understanding how information is used is essential for optimizing performance.

Quantifying information transmission is vital for understanding and improving natural or engineered information-processing systems. Shannon's *information theory* [164] provides the framework for studying the efficiency and reliability of any communication channel, whether it's a telephone line, a biochemical signaling cascade, or a neural pathway in the brain. The cornerstone of information theory is a set of mathematical definitions to rigorously quantify amounts of information. These makes it possible to determine, in absolute terms, the amount of information that is transmitted by a given information-processing mechanism, for a specific input signal. Moreover, it is possible to quantify the maximum amount of information that can be transmitted through a given mechanism under optimal conditions: this limit is known as the channel capacity, measured in bits per time unit. Shannon's information measures enable us to characterize a wide range of systems in terms of their information transmission capabilities.

Information theory has found many applications across disciplines, and is frequently used to understand and improve sensory or computational systems. In biology, information transmission is studied, e.g., in the brain, by analyzing the timing of electrical impulses between neurons [177, 154]. Within cells, information flow in

[1] In mathematical terms, this interplay between information and reward can be characterized by utility functions, which quantify the benefits of different actions based on available information [128, 158].

biochemical signaling and transcription regulation has been extensively studied by analyzing biochemical pathways [190, 111, 27]. In artificial intelligence, information theory has proven useful in improving learning in neural networks. The information bottleneck theory [185] suggests that the performance of neural networks can be enhanced by balancing compression and information retention during training [184, 166]. In economics and finance, information theory has been applied to describe financial markets [52] and to optimize financial decision-making under uncertainty [84]. In optics, information theory is employed to study the efficiency of signal processing in optical resonators, with applications in precision sensing and optical computing [5, 141]. Information theory boasts a wealth of applications and is essential for the analysis and theoretical understanding of information-processing systems.

The canonical measure for the quality of information transmission is the mutual information. It quantifies how much information is shared between two random variables, such as the input and output signals of an information-processing mechanism, see Fig. 1.1. Let $S$ and $X$ be two random variables that are jointly distributed according to the density $\mathrm{P}(s,x)$ and with marginal densities $\mathrm{P}(s)$ and $\mathrm{P}(x)$. The mutual information between $S$ and $X$ is then defined as

$$I(S,X) = \iint \mathrm{P}(s,x) \ln\left(\frac{\mathrm{P}(s,x)}{\mathrm{P}(s)\,\mathrm{P}(x)}\right) ds\,dx \tag{1.1}$$

and provides a measure of correlation between the random variables.[2] From the definition it follows that $I(S,X) = 0$ only if $S$ and $X$ are statistically independent, and $I(S,X) > 0$ otherwise. Thus, the mutual information quantifies the statistical dependence between random variables, equally characterizing the degrees of influence from $S \to X$ and from $X \to S$. Hence, the mutual information is a symmetric measure, satisfying $I(S,X) = I(X,S)$. In a typical information processing system, the input $S$ influences the output $X$ but there is no feedback from $X$ to $S$. In such cases, the mutual information $I(S,X)$ provides a measure for how effectively information about $S$ is transmitted through the system into the output $X$.

In biological systems, information transmission has frequently been quantified via the *instantaneous mutual information* (IMI) $I(S_{t_1}, X_{t_2})$, i.e. the mutual information between stimulus and response at two time points. This measure has been

[2] In contrast to other correlation measures used in statistics, such as the Pearson correlation coefficient, the mutual information captures both linear and nonlinear dependencies between variables. Additionally, in contrast to other correlation measures, the mutual information satisfies the data processing inequality, which states that no type of post-processing can increase the mutual information between the input and output [30, 90]. These properties make the mutual information uniquely suited for describing the fidelity of the input-output mapping in information-processing systems. Note however that a naïve use of the data processing inequality leads to seemingly contradictory results when applied to the stationary dynamics of processing cascades [142, 49, 34].

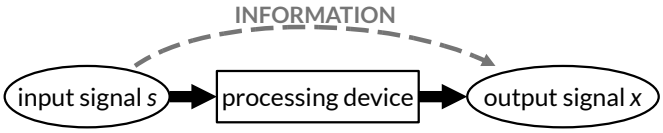



Figure 1.1: A generic information processing device takes an input signal $s$ and produces an output signal $x$. Because the output is correlated with the input, we can quantify the information that $s$ and $x$ share. This quantity is called the mutual information and measures the information that is transmitted.

applied for analyzing biochemical pathways [191, 205, 27, 23, 29, 135, 142] and neural spiking dynamics [177, 20]. However, in many cases, the IMI cannot correctly quantify information transmission due to correlations within the input or the output which reduce the total information transmitted. More generally, information may be encoded in the temporal patterns of signals, which cannot be captured by a pointwise information measure like the IMI. Thus, the IMI is generally inadequate for computing information transmission in systems which process dynamical signals.

There are many examples of information being encoded in dynamical features of signals. In cellular $Ca^{2+}$ signaling, information seems to be encoded in the timing and duration of calcium bursts [22], while in the MAPK pathway information is encoded in the amplitude and duration of the transient phosphorylation response to external stimuli [116, 126]. Moreover, there are reasons to believe that encoding information in dynamical signal features is advantageous for reliable information transmission [163]. Studying the information transmitted via temporal features is thus highly desirable but not possible with an instantaneous information measure. Therefore, in cases where the dynamics of input or output time-series may carry relevant information, the need for appropriate dynamical information measures has been widely recognized [176, 194, 156, 112, 181, 107, 126, 72, 196, 130].

The natural measure for quantifying information transmission via dynamical signals is the *trajectory mutual information*. It takes into account the total information encoded in the input and output trajectories of a system, and therefore captures all information transmitted over a specific time interval. Conceptually, its definition is simple. The trajectory mutual information is the mutual information between the input and output trajectories of a stochastic process, given by

$$I(\boldsymbol{S},\boldsymbol{X}) = \iint \mathrm{P}(\boldsymbol{s},\boldsymbol{x}) \ln\left(\frac{\mathrm{P}(\boldsymbol{s},\boldsymbol{x})}{\mathrm{P}(\boldsymbol{s})\,\mathrm{P}(\boldsymbol{x})}\right) d\boldsymbol{s}\, d\boldsymbol{x} \tag{1.2}$$

where the bold symbols $\boldsymbol{s}$ and $\boldsymbol{x}$ are used to denote trajectories. These trajectories

arise from a stochastic process that defines the joint probability distribution $\mathrm{P}(\boldsymbol{s}, \boldsymbol{x})$. The integral itself runs over all possible input and output trajectories.

The closely related *mutual information rate* is defined as the rate at which the trajectory mutual information increases with the duration of the trajectories in the long-time limit. Let $\boldsymbol{S}_T$ and $\boldsymbol{X}_T$ be trajectories of duration $T$, then the mutual information rate is given by

$$R = \lim_{T\to\infty} \frac{I(\boldsymbol{S}_T, \boldsymbol{X}_T)}{T} \,. \tag{1.3}$$

The mutual information rate quantifies how many independent messages can be transmitted per unit time, on average, via a communication channel. It depends on both, the signal statistics of the input, as well as the transmission properties of the channel. In the absence of feedback it is equal to the transfer entropy [160, 81].

The trajectory mutual information and the mutual information rate are fundamental measures for information transmission in dynamical systems. They serve as key performance metrics for biochemical signaling networks [194, 27], as well as for neural sensory systems [177, 20]. More generally, in communication channels with memory, the mutual information rate for the optimal input signal determines the channel capacity [30]. In financial markets, it quantifies correlations in stochastic time series, such as stock prices and trading volumes [52]. Finally, in non-equilibrium thermodynamics, the trajectory mutual information provides a link between information theory and stochastic thermodynamics [6, 73]. Efficient methods for calculating the trajectory mutual information and the mutual information rate are needed and constitute the primary objective of this thesis.

Unfortunately, calculating the mutual information between trajectories is notoriously difficult due to the high dimensionality of trajectory space [137]. Conventional approaches for computing mutual information require non-parametric estimates of the input and output entropy, typically obtained via histograms or kernel density estimators [177, 137, 27, 190, 192, 112]. However, the high-dimensional nature of trajectories makes it infeasible to obtain enough data for accurate non-parametric distribution estimates. Other non-parametric entropy estimators such as the k-nearest-neighbor estimator [81, 94] depend on a choice of metric in trajectory space and become unreliable for long trajectories [25]. Thus, except for very simple systems [112], the curse of dimensionality makes it infeasible to obtain accurate results for the trajectory mutual information using conventional mutual information estimators.

Due to the inherent difficulty of directly estimating the mutual information between trajectories, previous research has often employed simplified models or approximations. In some cases, the problem can be simplified by considering static (scalar) inputs instead of input signal trajectories [163, 25, 181]. But this approach ignores the dynamics of the input signal. Lower bounds for the mutual information

can be derived from the Donsker-Varadhan inequality [43, 13, 108], or obtained through general-purpose compression algorithms [9, 59, 25]. While exact analytical results for the trajectory mutual information are available for certain simple processes such as Gaussian [194] or Poisson channels [169, 61], many complex, realistic systems lack analytical solutions, and approximations have to be employed. For systems governed by a master equation, numerical or analytical approximations are sometimes feasible [46, 119] but these become intractable for complex systems. Finally, the Gaussian framework for approximating the mutual information rate is particularly widely used [194, 107, 72], though it assumes linear system dynamics and Gaussian noise statistics. These assumptions make it ill-suited for many realistic nonlinear information-processing systems.

To address the limitations of previous methods, we introduce *Path Weight Sampling* (PWS), a novel Monte Carlo technique for computing the trajectory mutual information efficiently and accurately. PWS leverages free-energy estimators from statistical physics and combines analytical and numerical methods to circumvent the curse of dimensionality associated with long trajectories. The approach relies on exact calculations of trajectory likelihoods derived analytically from a stochastic model. By averaging these likelihoods in a Monte Carlo fashion, PWS can accurately compute the trajectory mutual information, even in high-dimensional settings.

PWS is an exact Monte Carlo scheme, in the sense that it provides an unbiased statistical estimate of the trajectory mutual information. In PWS, the mutual information is computed via the identity

$$I(\boldsymbol{S},\boldsymbol{X}) = H(\boldsymbol{X}) - H(\boldsymbol{X} \mid \boldsymbol{S}) \tag{1.4}$$

as the difference between the marginal output entropy $H(\boldsymbol{X})$ associated with the marginal distribution $\mathrm{P}(\boldsymbol{x})$ of the output trajectories $\boldsymbol{x}$ and the conditional output entropy $H(\boldsymbol{X} \mid \boldsymbol{S})$ associated with $\mathrm{P}(\boldsymbol{x}|\boldsymbol{s})$, the conditional output distribution for a given input $\boldsymbol{s}$. Both entropies are evaluated as Monte-Carlo averages over the associated distribution, i.e., $H(\boldsymbol{X}) = -\langle \ln \mathrm{P}(\boldsymbol{x})\rangle$ and $H(\boldsymbol{X} \mid \boldsymbol{S}) = -\langle \ln \mathrm{P}(\boldsymbol{x}|\boldsymbol{s})\rangle$, where the notation $\langle\cdot\rangle$ denotes an average with respect to the joint distribution $\mathrm{P}(\boldsymbol{s},\boldsymbol{x})$. The key insights of PWS are that the conditional probability $\mathrm{P}(\boldsymbol{x}|\boldsymbol{s})$ can be directly evaluated from a generative model of the system, and that the marginal probability $\mathrm{P}(\boldsymbol{x})$ can be computed efficiently via marginalization using Monte Carlo procedures inspired by computational statistical physics.

The crux of PWS lies in the efficient computation of $\mathrm{P}(\boldsymbol{x})$ via the marginalization integral

$$\mathrm{P}(\boldsymbol{x}) = \int \mathrm{P}(\boldsymbol{x}|\boldsymbol{s})\,\mathrm{P}(\boldsymbol{s})\,d\boldsymbol{s}\,. \tag{1.5}$$

To evaluate this integral efficiently, we present different variants of PWS. In Chapter 2 we introduce *Direct PWS*, the simplest variant of PWS, where Eq. (1.5) is com-

puted bia a "brute-force" Monte Carlo approach that works well for short trajectories, but which becomes exponentially harder for long trajectories. In Chapter 3, we present two additional variants of PWS that evaluate the marginalization integral more efficiently, *RR-PWS* and *TI-PWS*. Rosenbluth-Rosenbluth PWS (RR-PWS) is based on efficient free-energy estimation techniques developed in polymer physics [155, 167, 69, 55]. Thermodynamic integration PWS (TI-PWS) uses techniques from transition path sampling to derive a MCMC sampler in trajectory space [19]. From this MCMC chain, we can compute the marginalization integral using thermodynamic integration [62, 127, 55]. Finally, in Chapter 6, we introduce a fourth marginalization technique based on variational inference via neural networks [88]. Its conceptual simplicity, coupled with powerful marginalization methods, make PWS a versatile framework for computing the trajectory mutual information in a variety of scenarios.

Yet, to compute the mutual information PWS requires evaluating the conditional trajectory probability $P(\boldsymbol{x}|\boldsymbol{s})$, which in turn requires a stochastic model defining a probability measure over trajectories. While (stochastic) mechanistic models of experimental systems are increasingly becoming available, the question remains whether PWS can be applied directly to experimental data when no such model is available. In Chapter 6, we show that machine learning can be used to construct a data-driven stochastic model that captures the trajectory statistics, i.e. $P(\boldsymbol{x}|\boldsymbol{s})$, enabling the application of PWS to experimental data.

We demonstrate the practical utility of PWS by calculating the trajectory mutual information for a range of systems. In Chapters 3 and 5, we study a minimal model for gene expression, showing that PWS can estimate the mutual information rate for this system more accurately than any previous technique. Using PWS, we reveal that the Gaussian approximation, though expected to hold due to the system's linearity, does not provide an accurate estimate in this case. In Chapters 5 and 6 we extend our analysis to simple nonlinear models for information transmission, comparing PWS results against the Gaussian approximation; for these models, PWS is the first technique capable of accurately computing trajectory mutual information. Moreover, in Chapter 4 we apply PWS to a complex stochastic model of bacterial chemotaxis, marking the first instance where the information rate for a system of this complexity can be computed exactly. Together, these examples demonstrate that an exact technique like PWS is indispensable for understanding information transmission in realistic scenarios.

## 1.1 Contributions of This Work

The main contributions of this thesis are as follows:

1. **PWS: A novel framework for computing the trajectory mutual information**: We introduce Path Weight Sampling, a computational framework for calculating the trajectory mutual information in dynamical stochastic systems. This framework is exact, applicable to both continuous and discrete time processes, and does not rely on any assumptions about the system's dynamics. PWS and its main variants are described in Chapters 2 and 3.

2. **Discovery of discrepancies between experiments and mathematical models of chemotaxis**: We apply PWS to various systems, including the complex bacterial chemotaxis signaling network. By studying the information transmission rate of chemotaxis and comparing our results against those of Mattingly et al. [107], we find that the widely-used MWC model of chemotaxis cannot explain the experimental data. We find that the number of receptor clusters is smaller and that the size of these clusters is larger than hitherto believed. We describe and characterize this finding in Chapter 4.

3. **Study of the accuracy of the gaussian approximation for the information rate**: In Chapter 5, we use PWS to quantitatively study the accuracy of the widely-used Gaussian approximation. Before PWS, no exact technique was available to obtain *ground truth* results of the mutual information rate for non-linear systems, and the accuracy of the Gaussian framework could not be evaluated. We reveal that the Gaussian model can be surprisingly inaccurate, even for linear reaction systems.

4. **Neural networks for learning the stochastic dynamics from time-series data**: In Chapter 6, we demonstrate that recent machine learning techniques can be employed to automatically learn the stochastic dynamics from experimental data. We show that by combining these learned models with PWS, it becomes possible to compute the trajectory mutual information directly from time-series data. This approach outperforms previous techniques, like the Gaussian approximation, for estimating information rates from data.

## 1.2 Thesis Outline

The remainder of this thesis is divided into 5 chapters. We first present three variants of PWS, all of which compute the conditional entropy in the same manner, but differ in the way this Monte Carlo averaging procedure for computing the marginal probability $\mathcal{P}[\boldsymbol{x}]$ is carried out. Chapters 2 to 4 of this thesis have been published

previously in *Physical Review X*.[3]

In Chapter 2 we present the simplest PWS variant, *Direct* PWS (DPWS). To compute $\mathcal{P}[\boldsymbol{x}]$, DPWS performs a brute-force average of the path likelihoods $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ over the input trajectories $\boldsymbol{s}$. While we show that this scheme works for simple systems, the brute-force Monte Carlo averaging procedure becomes more difficult for larger systems and exponentially harder for longer trajectories.

In Chapter 3, we present our second and third variant of PWS which are based on the realization that the marginal probability $\mathcal{P}[\boldsymbol{x}]$ is akin to a partition function. These schemes leverage techniques for computing free energies from statistical physics. We also apply PWS to a simple model system which consists of a simple pair of coupled birth-death processes which allows us to compare the efficiency of the three PWS variants, as well as to compare the PWS results against analytical results from the Gaussian approximation [194].

In Chapter 4, we apply PWS to the bacterial chemotaxis system, which is arguably the best characterized signaling system in biology. Mattingly et al. [107] recently argued that bacterial chemotaxis in shallow gradients is information limited. Yet, to compute the information rate from their experimental data they had to employ a Gaussian framework. PWS makes it possible to asses the accuracy of this approximation.

Chapter 5 is devoted to studying the accuracy of the Gaussian approximation for non-Gaussian systems. By understanding the limitations and strengths of the Gaussian approximation, this chapter aims to provide deeper insights into selecting the appropriate method for MI estimation depending on the system.

Finally, Chapter 6 we introduce ML-PWS, which combines recent machine learning models with PWS, to compute the mutual information directly from data. This idea significantly extends the range of applications for PWS, since we no longer require a mechanistic model of the system. Instead, the stochastic model is automatically learned from the data.

[3] M. Reinhardt, G. Tkačik, and P. R. ten Wolde, Path Weight Sampling: Exact Monte Carlo Computation of the Mutual Information between Stochastic Trajectories, *Phys. Rev. X* **13**, 041017 (2023) [152]

# 2 Path Weight Sampling

*Most natural and engineered information-processing systems transmit information via signals that vary in time. Computing the information transmission rate or the information encoded in the temporal characteristics of these signals requires the mutual information between the input and output signals as a function of time, i.e., between the input and output trajectories. Yet, this is notoriously difficult because of the high-dimensional nature of the trajectory space, and all existing techniques require approximations. We present an exact Monte Carlo technique called Path Weight Sampling (PWS) that, for the first time, makes it possible to compute the mutual information between input and output trajectories for any stochastic system that is described by a master equation. The principal idea is to use the master equation to evaluate the exact conditional probability of an individual output trajectory for a given input trajectory and average this via Monte Carlo sampling in trajectory space to obtain the mutual information. PWS also makes it possible to compute the mutual information between input and output trajectories for systems with hidden internal states as well as systems with feedback from output to input.*

The contents of this chapter have been published in *Phys. Rev. X* **13**, 041017 (2023) [152].

Quantifying information transmission is vital for understanding and designing natural and engineered information-processing systems, ranging from biochemical and neural networks, to electronic circuits and optical systems [189, 188, 101]. Claude Shannon introduced the mutual information and the information rate as the central measures of Information Theory more than 70 years ago [164]. These measures quantify the fidelity by which a noisy system transmits information from its inputs to its outputs. Yet, computing these quantities exactly remains notoriously difficult, if not impossible. This is because the inputs and outputs are often not scalar values, but rather temporal trajectories.

Most, if not all, information-processing systems transmit signal that vary in time. The canonical measure for quantifying information transmission via time-varying signals is the mutual information rate [164, 30, 194, 52]. It quantifies the speed at which distinct messages are transmitted through the system, and it depends not only on the accuracy of the input-output mapping but also on the correlations within the input and output signals. Computing the mutual information rate thus requires computing the mutual information between the input and output trajectories, not between their signal values at given time points. The rate at which this trajectory mutual information increases with the trajectory duration in the long-time limit defines the mutual information rate, see Eq. (1.3). In the absence of feedback this rate also equals the multi-step transfer entropy [106, 160].

More generally, useful information is often contained in the temporal dynamics of the signal. A prime example is bacterial chemotaxis, where the response does not depend on the current ligand concentration, but rather on whether it has changed in the recent past [18, 162]. Moreover, the information from the input may be encoded in the temporal dynamics of the output [104, 146, 163, 68]. Quantifying information encoded in these temporal features of the signals requires the mutual information not between two time points, i.e. the instantaneous mutual information, but rather between input and output trajectories [194].

Unfortunately, computing the mutual information between trajectories is exceptionally difficult. The conventional approach requires non-parametric distribution estimates of the input and output distributions, e.g. via histograms of data obtained through simulations or experiments [177, 137, 27, 190, 192, 112]. These non-parametric distribution estimates are necessary because the mutual information cannot generally be computed from summary statistics like the mean or variance of the data alone. However, the high-dimensional nature of trajectories makes it infeasible to obtain enough empirical data to accurately estimate the required probability distributions. Moreover, this approach requires the discretization of time, which becomes problematic when the information is encoded in the precise timing of signal spikes, as, e.g., in neuronal systems [154]. Except for the simplest systems with a binary state space [112], the conventional approach to estimate the mutual

information via histograms therefore cannot be transposed to trajectories.

Because there are currently no general schemes available to compute the mutual information between trajectories exactly, approximate methods or simplified models are typically used. While empirical distribution estimates can be avoided by employing the K-nearest-neighbors entropy estimator [81, 94], this method depends on a choice of metric in trajectory space and can become unreliable for long trajectories [25]. Alternative, decoding-based information estimates can be developed for trajectories [59], but merely provide a lower bound of the mutual information, and it remains unclear how tight these lower bounds are [20, 25, 76]. Analytical results are avaiable for simple systems [183], and for linear systems that obey Gaussian statistics, the mutual information between trajectories can be obtained from the covariance matrix [194]. However, many information processing systems are complex and non-linear such that the Gaussian approximation does not hold, and analytical solutions do not exist. A more promising approach to estimate the trajectory mutual information for chemical reaction networks has been developed by Duso and Zechner [46] and generalized in Ref. [119]. However, the scheme relies on a moment closure approximation and has so far only been applied to very simple networks, seemingly being difficult to extend to complex systems.

Here, we present *Path Weight Sampling* (PWS), an exact technique to compute the trajectory mutual information for any system described by a master equation. Master equations are widely used to model chemical reaction networks [37, 109, 110, 48], biological population growth [50, 140, 32], economic processes [209, 100], and a large variety of other systems [74, 24], making our scheme of interest to a broad class of problems.

PWS is an exact Monte Carlo scheme, in the sense that it provides an unbiased statistical estimate of the trajectory mutual information. In PWS, the mutual information is computed as the difference between the marginal output entropy associated with the marginal distribution $\mathcal{P}[\boldsymbol{x}]$ of the output trajectories $\boldsymbol{x}$, and the conditional output entropy associated with the output distribution $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ conditioned on the input trajectory $\boldsymbol{s}$. Our scheme is inspired by the observation of Cepeda-Humerez et al. [25] that the path likelihood, i.e., the probability $\mathcal{P}[\boldsymbol{x}|s]$, can be computed exactly from the master equation for a *static* input signal $s$. This makes it possible to compute the mutual information between a discrete input and a time-varying output via a Monte Carlo averaging procedure of the likelihoods, rather than from an empirical estimate of the intractable high-dimensional probability distribution functions. The scheme of Cepeda-Humerez et al. [25] is however limited to discrete input signals that do not vary in time. Here we show that the path likelihood $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ can also be computed for a dynamical input *trajectory* $\boldsymbol{s}$, which allows us to compute the conditional output entropy also for time-varying inputs. While this solves the problem in part, the marginal output entropy associated with $\mathcal{P}[\boldsymbol{x}]$ cannot be com-

puted with the approach of Cepeda-Humerez et al., and thus requires a different scheme.

In Section 2.1 we show how, for time-varying input signals, the marginal probability $\mathcal{P}[\boldsymbol{x}]$ can be obtained as a Monte Carlo average of $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ over a large number of input trajectories. We then use the Monte Carlo estimate for $\mathcal{P}[\boldsymbol{x}]$ to compute the marginal output entropy.

In Section 2.2 we show that, surprisingly, our PWS methods additionally make it possible to compute the mutual information between input and output trajectories of systems with hidden internal states. Hidden states correspond, for example, to network components that merely relay, process or transform the signal from the input to the output. Indeed, the downstream system typically responds to the information that is encoded in this output, and not the other internal system components. Most information processing systems contain such hidden states, and generally we want to integrate out these latent network components. In addition, we can generalize PWS to systems with feedback from the output to the input as shown in Section 2.3.

## 2.1 Monte Carlo Estimate of the Mutual Information

In this section we present the fundamental ideas of PWS. These ideas lie at the heart of Direct PWS (DPWS) and also form the foundation of the other two more advanced PWS variants which will be explained in Chapter 3.

### 2.1.1 Statement of the Problem

All information processing systems repeatedly take an input value $s$ and produce a corresponding output $x$. Due to noise, the output produced for the same input can be different every time, such that the system samples outputs from the distribution $\mathrm{P}(x|s)$. In the information theoretic sense, the device's capabilities are fully specified by its output distributions for all possible inputs. We consider the inputs as being distributed according to a probability density $\mathrm{P}(s)$ such that the whole setup of signal and device is completely described by the joint probability density $\mathrm{P}(s, x) = \mathrm{P}(s)\,\mathrm{P}(x|s)$.

When the conditional output distributions $\mathrm{P}(x|s)$ overlap with each other, information is lost because the input can not always be inferred uniquely from the output (see Fig. 2.1). The remaining information that the output carries about the signal on average is quantified by the mutual information between input and output.

Mathematically, the mutual information between a random variable $\mathcal{S}$, representing the input, and a second random variable $\mathcal{X}$, representing the output, is defined

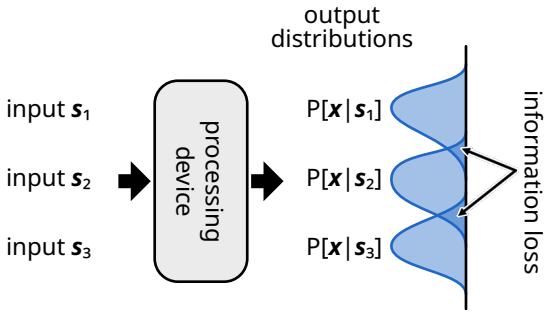



Figure 2.1: Schematic of information processing under the influence of noise. Overlapping output distributions for different inputs lead to information loss, because the input cannot always be uniquely inferred from the output. The mutual information $I(\mathcal{S},\mathcal{X})$ quantifies how much information the observation of the output typically retains about the input signal.

as

$$I(\mathcal{S},\mathcal{X}) = \iint ds\,dx\; P(s,x) \ln \frac{P(s,x)}{P(s)P(x)}\,, \tag{2.1}$$

where the marginal output distribution is given by $P(x) = \int ds\; P(s,x)$. The quantity $I(\mathcal{S},\mathcal{X})$ as defined above is a non-negative real number, representing the mutual information between $\mathcal{S}$ and $\mathcal{X}$ in nats. The integrals in Eq. (2.1) run over all possible realizations of the random variables $\mathcal{S}$ and $\mathcal{X}$. In our case, $\mathcal{S}$ and $\mathcal{X}$ represent stochastic trajectories and so the integrals become path integrals.

In general, the mutual information can be decomposed into two terms, a conditional and marginal entropy. Due to the symmetry of Eq. (2.1) with respect to exchange of $\mathcal{S}$ and $\mathcal{X}$, this decomposition can be written as

$$I(\mathcal{S},\mathcal{X}) = H(\mathcal{S}) - H(\mathcal{S}|\mathcal{X}) = H(\mathcal{X}) - H(\mathcal{X}|\mathcal{S})\,. \tag{2.2}$$

The (marginal) input entropy $H(\mathcal{S})$ represents the total uncertainty about the input, and the conditional input entropy $H(\mathcal{S}|\mathcal{X})$ describes the remaining uncertainty of the input after having observed the output. Thus, the mutual information $I(\mathcal{S},\mathcal{X}) = H(\mathcal{S}) - H(\mathcal{S}|\mathcal{X})$ naturally quantifies the reduction in uncertainty about the input through the observation of the output.

When analyzing data from experiments or simulations however, the mutual information is generally estimated via $I(\mathcal{S},\mathcal{X}) = H(\mathcal{X}) - H(\mathcal{X}|\mathcal{S})$. This is because simulation or experimental data generally provide information about the distribution of outputs for a given input, rather than vice versa. The accessible entropies are thus the marginal output entropy $H(\mathcal{X})$ and the conditional output entropy $H(\mathcal{X}|\mathcal{S})$,

which are defined as

$$\mathrm{H}(\mathcal{X}) = -\int dx\, \mathrm{P}(x) \ln \mathrm{P}(x) \tag{2.3}$$

$$\mathrm{H}(\mathcal{X}|\mathcal{S}) = -\int ds\, \mathrm{P}(s) \int dx\, \mathrm{P}(x|s) \ln \mathrm{P}(x|s)\,. \tag{2.4}$$

The conventional way of computing the mutual information involves generating many samples to obtain empirical distribution estimates for $\mathrm{P}(x|s)$ and $\mathrm{P}(x)$ via histograms. However, the number of samples needs to be substantially larger than the number of histogram bins to reduce the noise in the bin counts. Obtaining enough samples is effectively impossible for high-dimensional data, like signal trajectories. Moreover, any nonzero bin size leads to a systematic bias in the entropy estimates, even in one dimension [137]. These limitations of the conventional method make it impractical for high-dimensional data, highlighting the need for alternative approaches to accurately compute mutual information for trajectories.

### 2.1.2 Direct PWS

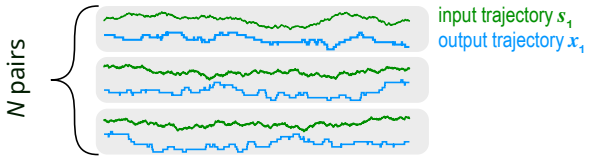
The central idea of PWS is to compute probability densities for trajectories exactly, sidestepping the problem having to estimate them via histograms. We exploit that for systems described by a master equation, the conditional probability of an output trajectory for a given input trajectory can be computed analytically. With this insight we can derive a procedure to compute the mutual information. Specifically, we will show that

- for a system described by a master equation, the trajectory likelihood $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ is a quantity that can be computed on the fly in a stochastic simulation;
- input trajectories can be generated from $\mathcal{P}[\boldsymbol{s}]$, output trajectories for a given input $\boldsymbol{s}$ can be generated according to $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ using standard SSA (Gillespie) simulations;
- by combining the two ideas above, we can derive a direct Monte Carlo estimate for the mutual information $\mathrm{I}(\mathcal{S},\mathcal{X})$, as illustrated in Fig. 2.2.

Note that we denote trajectories by bold symbols to distinguish them from scalar quantities.

Our technique is conceptually straightforward. Using Monte Carlo simulations we can compute averages over the configuration space of trajectories. Suppose we have a function $f[\boldsymbol{z}]$ that takes a trajectory $\boldsymbol{z}$ and produces a scalar value. The mean

1. **Simulate input output pairs** $(s_i, x_i)$




2. **Compute trajectory likelihoods** $P[x_i \mid s_i]$

$$P[x \mid s] = \mathrm{P}(x(t_0)) \prod_{i=1}^{n} \mathrm{P}(x(t_{i-1}) \to x(t_i) \mid s)$$

3. **Compute marginal probabilities** $P[x_i]$

| *Direct PWS* | *RR-PWS* | *TI-PWS* |
|---|---|---|
| Brute force estimate of the marginal probability $P[x]$ using direct samples from $P[s]$ | Estimate of the marginal probability $P[x]$ using biased sampling, inspired by polymer simulations | MCMC estimate of the marginal probability $P[x]$ using thermodynamic integration |

4. **Compute the mutual information**

$$\mathrm{I}(S,X) = \frac{1}{N} \sum_{i=1}^{N} (\ln P[x_i|s_i] - \ln P[x_i])$$

Figure 2.2: The PWS framework to compute the mutual information between trajectories in 4 steps. **1**. Generate $N$ input-output pairs from $\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]$. **2**. For each input-output pair compute the trajectory likelihood $\mathcal{P}[\boldsymbol{x}_i|\boldsymbol{s}_i]$ using Eq. (2.15). **3**. Compute $\mathcal{P}[\boldsymbol{x}_i]$ for every output. This step differentiates the different variants of PWS from each other. Direct PWS is presented in this chapter, whereas RR-PWS and TI-PWS are described in Chapter 3. **4**. Using the likelihoods and the marginal probabilities from the previous steps we can estimate the mutual information using Eq. (2.10).

of $f[\boldsymbol{z}]$ with respect to the trajectory distribution $\mathcal{P}[\boldsymbol{z}]$ is then

$$\langle f[\boldsymbol{z}]\rangle_{\mathcal{P}[\boldsymbol{z}]} \equiv \int \mathcal{D}[\boldsymbol{z}]\, \mathcal{P}[\boldsymbol{z}] f(\boldsymbol{z})\,. \tag{2.5}$$

We write $\int \mathcal{D}[\boldsymbol{z}]$ to denote a path integral over all possible trajectories of a given duration. We estimate $\langle f[\boldsymbol{z}]\rangle_{\mathcal{P}[\boldsymbol{z}]}$, by generating a large number of trajectories $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N$ from $\mathcal{P}[\boldsymbol{z}]$ and evaluating the corresponding Monte Carlo average

$$\hat{f}_N = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{z}_i) \tag{2.6}$$

which converges to the true mean in the limit $N \to \infty$.

Specifically, we want to estimate the conditional and the marginal entropy to compute the mutual information. Let us imagine that we generate $N$ input trajectories $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N$ from the distribution $\mathcal{P}[\boldsymbol{s}]$. Next, for every input $\boldsymbol{s}_i$, we generate a set of $K$ outputs $\boldsymbol{x}_{i,1}, \ldots, \boldsymbol{x}_{i,K}$ from $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}_i]$. Then, the Monte Carlo estimate for the conditional entropy is

$$\begin{aligned} \mathrm{H}(\mathcal{X}|\mathcal{S}) &= -\int \mathcal{D}[\boldsymbol{s}]\, \mathcal{P}[\boldsymbol{s}] \int \mathcal{D}[\boldsymbol{x}]\, \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] \ln \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] \\ &= -\Big\langle \langle \ln \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]\rangle_{\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]} \Big\rangle_{\mathcal{P}[\boldsymbol{s}]} \\ &\approx -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{K}\sum_{j=1}^{K} \ln \mathcal{P}[\boldsymbol{x}_{i,j}|\boldsymbol{s}_i]\,. \end{aligned} \tag{2.7}$$

Secondly, for a given output $\boldsymbol{x}$ we generate $M$ inputs $\boldsymbol{s}'_1, \ldots, \boldsymbol{s}'_M$ according to $\mathcal{P}[\boldsymbol{s}]$, then we can obtain a Monte Carlo estimate for the marginal probability of the output trajectory $\mathcal{P}[\boldsymbol{x}]$:

$$\begin{aligned} \mathcal{P}[\boldsymbol{x}] &= \int \mathcal{D}[\boldsymbol{s}]\, \mathcal{P}[\boldsymbol{s}] \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] \\ &= \langle \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]\rangle_{\mathcal{P}[\boldsymbol{s}]} \\ &\approx \frac{1}{M}\sum_{j=1}^{M} \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}'_j]\,. \end{aligned} \tag{2.8}$$

The estimate for the marginal entropy is then given by

$$\begin{aligned} \mathrm{H}(\mathcal{X}) &= -\int \mathcal{D}[\boldsymbol{x}]\, \mathcal{P}[\boldsymbol{x}] \ln \mathcal{P}[\boldsymbol{x}] \\ &= -\langle \ln \mathcal{P}[\boldsymbol{x}] \rangle_{\mathcal{P}[\boldsymbol{x}]} \\ &\approx -\frac{1}{N} \sum_{i=1}^{N} \ln \mathcal{P}[\boldsymbol{x}_i] \\ &\approx -\frac{1}{N} \sum_{i=1}^{N} \ln \left[ \frac{1}{M} \sum_{j=1}^{M} \mathcal{P}[\boldsymbol{x}_i | \boldsymbol{s}'_{i,j}] \right] . \end{aligned} \tag{2.9}$$

In the last step we inserted the result from Eq. (2.8). In this estimate, the trajectories $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ are sampled from $\mathcal{P}[\boldsymbol{x}]$, i.e., by first sampling from $\mathcal{P}[\boldsymbol{s}]$ and then from $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$. Finally, the mutual information is obtained by taking the entropy difference, i.e., $\mathrm{I}(\mathcal{S}, \mathcal{X}) = \mathrm{H}(\mathcal{X}) - \mathrm{H}(\mathcal{X}|\mathcal{S})$.

While this is the main idea behind PWS, it is computationally advantageous to change the order of operations in the estimate. Specifically, computing the difference of two averages, leads to large statistical errors. We can obtain an improved estimate by reformulating the mutual information as a single average of differences:

$$\begin{aligned} \mathrm{I}(\mathcal{S}, \mathcal{X}) &= \int \mathcal{D}[\boldsymbol{s}] \int \mathcal{D}[\boldsymbol{x}]\, \mathcal{P}[\boldsymbol{s}, \boldsymbol{x}] \ln \frac{\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]}{\mathcal{P}[\boldsymbol{x}]} \\ &= \langle \ln \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] - \ln \mathcal{P}[\boldsymbol{x}] \rangle_{\mathcal{P}[\boldsymbol{s},\boldsymbol{x}]} . \end{aligned} \tag{2.10}$$

This equation applies to all variants of PWS. They differ, however, in the way $\mathcal{P}[\boldsymbol{x}]$ is computed. In the brute-force version of PWS, called *Direct PWS* (DPWS), we use Eq. (2.8) to evaluate the marginal probability $\mathcal{P}[\boldsymbol{x}]$. DPWS indeed involves two nested Monte Carlo computations, in which $N$ pairs $(\boldsymbol{s}_i, \boldsymbol{x}_i)$ are generated, and for each output $\boldsymbol{x}_i$, $M$ input trajectories $\{\boldsymbol{s}\}$ are generated from scratch to compute $\mathcal{P}[\boldsymbol{x}]$. In Chapter 3, we present two additional variants of PWS where the brute-force estimate of the marginal probability $\mathcal{P}[\boldsymbol{x}]$ is replaced by more elaborate schemes. That said, DPWS is a conceptually simple, straightforward to implement, and exact scheme to compute the mutual information.

Having explained the core ideas of our technique above, we will continue this section with a review of the necessary concepts of master equations to implement PWS. First, in Section 2.1.3, we derive the formula for the conditional probability $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ which lies at the heart of our technique. In Sections 2.1.3 and 2.1.4, we discuss how trajectories are generated according to $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ and $\mathcal{P}[\boldsymbol{s}]$, which are the remaining ingredients required for using DPWS. Then, in Chapter 3, we will present the two other variants of PWS that improve on DPWS.

### 2.1.3 Driven Markov Jump Process

In this chapter, we consider systems that can be modeled by a master equation and are being driven by a stochastic signal. The master equation specifies the time evolution of the conditional probability distribution $\mathrm{P}(x,t|x_0,t_0)$ which is the probability for the process to reach the discrete state $x \in \Omega$ at time $t$, given that it was at state $x_0 \in \Omega$ at the previous time $t_0$. The state space $\Omega$ is multi-dimensional if the system is made up of multiple components and therefore $x$ and $x_0$ can be vectors rather than scalar values. Denoting the transition rate at time $t$ from state $x$ to another state $x' \neq x$ by $w_t(x',x)$, the master equation reads

$$\frac{\partial \mathrm{P}(x,t)}{\partial t} = \sum_{x' \in \Omega \setminus \{x\}} \left[ w_t(x,x')\mathrm{P}(x',t) - w_t(x',x)\mathrm{P}(x,t) \right], \tag{2.11}$$

where, for brevity, we suppress the dependence on the initial condition, i.e., $\mathrm{P}(x,t) = \mathrm{P}(x,t|x_0,t_0)$. By defining $Q_t(x',x) = w_t(x',x)$ for $x \neq x'$ and

$$Q_t(x,x) = -\sum_{x' \in \Omega \setminus \{x\}} w_t(x',x) \tag{2.12}$$

the master equation simplifies to

$$\frac{\partial \mathrm{P}(x,t)}{\partial t} = \sum_{x' \in \Omega} Q_t(x,x')\mathrm{P}(x',t)\,. \tag{2.13}$$

Note that by definition the diagonal matrix element $Q_t(x,x)$ is the negative exit rate from state $x$, i.e. the total rate at which probability flows away from state $x$.

Using the master equation we can compute the probability of any trajectory. A trajectory $\boldsymbol{x}$ is defined by a list of jump times $t_1, \ldots, t_{n-1}$, together with a sequence of system states $x_0, \ldots, x_{n-1}$. The trajectory starts at time $t_0$ in state $x_0$ and ends at time $t_n$ in state $x_{n-1}$, such that its duration is $T = t_n - t_0$. At each time $t_i$ (for $i = 1, \ldots, n-1$) the trajectory describes an instantaneous jump $x_{i-1} \to x_i$. The probability density of $\boldsymbol{x}$ is

$$\begin{aligned} \mathcal{P}[\boldsymbol{x}] = \mathrm{P}(x_0) \times &\left( \prod_{i=1}^{n-1} Q_{t_i}(x_i, x_{i-1}) \right) \\ &\times \left( \prod_{i=1}^{n} \exp \int_{t_{i-1}}^{t_i} dt\; Q_t(x_{i-1}, x_{i-1}) \right), \end{aligned} \tag{2.14}$$

a product of the probability of the initial state $\mathrm{P}(x_0)$, the rates of the $n-1$ transitions $Q_{t_i}(x_i, x_{i-1})$, and the survival probabilities for the waiting times between jumps, given by $\exp \int_{t_{i-1}}^{t_i} dt\; Q_t(x_{i-1}, x_{i-1})$ for $i = 1, \ldots, n$.

### Computing the Likelihood $\mathcal{P}[x|s]$

To compute the likelihood or conditional probability $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ of an output trajectory $\boldsymbol{x}$ for a given input trajectory $\boldsymbol{s}$, we note that the input determines the time-dependent stochastic dynamics of the jump process. Indeed, the transition rates at time $t$, given by $Q_t(x', x; \boldsymbol{s})$, depend explicitly on the input $s(t)$ at time $t$ and may even depend on the entire history of $\boldsymbol{s}$ prior to $t$.

In the common case that every input trajectory $\boldsymbol{s}$ leads to a unique transition rate matrix $Q_t(x', x; \boldsymbol{s})$, i.e. the map $\boldsymbol{s} \mapsto Q_t(\cdot, \cdot; \boldsymbol{s})$ is injective, the likelihood is directly given by Eq. (2.14):

$$\begin{aligned}\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] = \mathrm{P}(x_0|s_0) \times \left( \prod_{i=1}^{n-1} Q_{t_i}\left(x_i, x_{i-1}; \boldsymbol{s}\right) \right) \\ \times \left( \prod_{i=1}^{n} \exp \int_{t_{i-1}}^{t_i} dt\; Q_t(x_{i-1}, x_{i-1}; \boldsymbol{s}) \right)\end{aligned} \tag{2.15}$$

where $\mathrm{P}(x_0|s_0)$ is the probability of the initial state $x_0$ of the output given the initial state of the input $s_0 = s(t_0)$.

The evaluation of the trajectory likelihood is at the heart of our Monte Carlo scheme. However, numerically computing a large product like Eq. (2.15) very quickly reaches the limits of floating point arithmetic since the result is often either too large or too close to zero to be representable as a floating point number. Thus, to avoid numerical issues, it is vital to perform the computations in log-space, i.e. to compute

$$\ln \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] = \ln \mathrm{P}(x_0|s_0) + \int_{t_0}^{T} dt\; \mathcal{L}_t[\boldsymbol{s}, \boldsymbol{x}] \tag{2.16}$$

where

$$\mathcal{L}_t[\boldsymbol{s}, \boldsymbol{x}] = Q_t(x(t), x(t); \boldsymbol{s}) + \sum_{i=1}^{n-1} \delta(t - t_i) \ln Q_t(x_i, x_{i-1}; \boldsymbol{s})\,. \tag{2.17}$$

The computation of the log-likelihood $\ln \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ for given trajectories $\boldsymbol{s}$ and $\boldsymbol{x}$ according to Eqs. (2.16) and (2.17) proceeds as follows:

- At the start of the trajectory we compute the log-probability of the initial condition $\ln \mathrm{P}(x_0|s_0)$,
- for every jump $x_{i-1} \to x_i$ in $\boldsymbol{x}$ (at time $t_i$) compute the log jump propensity $\ln Q_{t_i}(x_i, x_{i-1}; \boldsymbol{s})$, and

- for every interval $(t_{i-1}, t_i)$ of constant output value $x(t) = x_{i-1}$ between two jumps of $\boldsymbol{x}$, we compute $\int_{t_{i-1}}^{t_i} dt\ Q_t(x_{i-1}, x_{i-1}; \boldsymbol{s})$. This integral can be performed using standard numerical methods such as the trapezoidal rule, which is also exact if $Q_t(x(t), x(t); \boldsymbol{s})$ is a piecewise linear function of $t$.

The sum of the three contributions above yields the exact log-likelihood $\ln \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ as given in Eq. (2.16).

The algorithm to compute the log-likelihood $\ln \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ is both efficient and straightforward to implement, being closely related to the standard Gillespie algorithm. The only term in Eq. (2.16) that cannot be directly obtained from the master equation is $\ln \mathrm{P}(x_0|s_0)$. This quantity depends on the initial distribution of the system of interest.

Our scheme can be applied to any system with a well-defined (non-equilibrium) initial distribution $\mathrm{P}(s_0, x_0)$ as specified by, e.g. the experimental setup. Most commonly though, one is interested in studying information transmission for systems in steady state. Then, the initial condition $\mathrm{P}(s_0, x_0)$ is the stationary distribution of the Markov process. Depending on the complexity of the system, this distribution can be found either analytically from the master equation [201, 208] (possibly using simplifying approximations [204, 85]), or computationally from stochastic simulations [65].

### Sampling from $\mathcal{P}[x|s]$

Standard kinetic Monte Carlo simulations naturally produce exact samples of the probability distribution $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ as defined in Eq. (2.15). That is, for any signal trajectory $\boldsymbol{s}$ and initial state $x_0$ drawn from $\mathrm{P}(x_0|s_0)$ we can use the Stochastic Simulation Algorithm (SSA) or variants thereof to generate a corresponding trajectory $\boldsymbol{x}$. The SSA propagates the initial condition $x_0, t_0$ forward in time according to the transition rate matrix $Q_t(\cdot; \boldsymbol{s})$. In the standard Direct SSA algorithm [65] this is done by alternatingly sampling the waiting time before the next transition, and then selecting the actual transition.

The transition rates $Q_t(x', x; \boldsymbol{s})$ of a driven master equation are necessarily time-dependent since they include the coupling of the jump process to the input trajectory $\boldsymbol{s}$, which itself varies in time. These time-varying transition rates need to be accounted for in the SSA. While most treatments of the SSA assume that the transition rates are constant in time, this restriction is easily lifted. Consider step $i$ of the Direct SSA which generates the next transition time $t_{i+1} = t_i + \Delta t_i$. For time-varying transition rates the distribution of the stochastic waiting time $\Delta t_i$ is characterized

by the survival function

$$S_i(\tau) = \mathrm{P}(\Delta t_i > \tau) = \exp \int_{t_i}^{t_i+\tau} dt\; Q_t(x_i, x_i; \boldsymbol{s})\,. \tag{2.18}$$

The waiting time can be sampled using inverse transform sampling, i.e. by generating a uniformly distributed random number $u \in [0, 1]$ and computing the waiting time using the inverse survival function $\Delta t_i = S_i^{-1}(u)$. Numerically, computing the inverse of the survival function requires solving the equation

$$\ln u = \int_{t_i}^{t_i+\Delta t_i} dt\; Q_t(x_i, x_i; \boldsymbol{s}) \tag{2.19}$$

for the waiting time $\Delta t_i$. Depending on the complexity of $Q_t(x_i, x_i|\boldsymbol{s})$, this equation can either be solved analytically or numerically, e.g. using Newton's method. Hence, this method to generate stochastic trajectories is only truly exact if we can solve Eq. (2.19) analytically. Additionally, in some cases more efficient variants of the SSA with time dependent rates could be used [144, 182].

### 2.1.4 Input Statistics

For our mutual information estimate, we need to be able to draw samples from the input distribution $\mathcal{P}[\boldsymbol{s}]$. Our algorithm poses no restrictions on $\mathcal{P}[\boldsymbol{s}]$ other than the possibility to generate sample trajectories.

For example, the input signal may be described by a continuous-time jump process. One benefit is that it is possible to generate exact realizations of such a process (using the SSA) and to exactly compute the likelihood $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ using Eq. (2.16). Specifically, the likelihood can be exactly evaluated because the transition rates $Q_t(\cdot, \cdot; \boldsymbol{s})$ for any input trajectory $\boldsymbol{s}$, while time-dependent, are *piece-wise constant*. This implies that the integral in Eq. (2.16) can be evaluated analytically without approximations. Similarly, for piece-wise constant transition rates, the inverse function of Eq. (2.19) can be evaluated directly such that we can sample exact trajectories from the driven jump process. As a result, when both input and output are described by a master equation, PWS is a completely exact Monte Carlo scheme to compute the mutual information.

However, the techniques described here do *not* require the input signal $\boldsymbol{s}$ to be described by a continuous-time jump process, or even to be Markovian. The input signal can be any stochastic process for which trajectories can be generated numerically. This includes continuous stochastic processes that are found as solutions to stochastic differential equations [91].

## 2.2 Integrating Out Internal Components

So far the output trajectory $\boldsymbol{x}$ has been considered to correspond to the trajectory of the system in the *full* state space $\Omega$. Concomitantly, the method presented is a scheme for computing the mutual information between the input signal $\boldsymbol{s}$ and the trajectory $\boldsymbol{x}$, comprising the time evolution of all the $n$ components in the system, $X^1, X^2, \dots, X^n$. Each component $X^i$ itself has a corresponding trajectory $\boldsymbol{x}^i$, such that the full trajectory can be represented as a vector $\boldsymbol{x} = (\boldsymbol{x}^1, \dots, \boldsymbol{x}^n)$. It is indeed also the conditional probability $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] = \mathcal{P}[\boldsymbol{x}^1, \dots, \boldsymbol{x}^n|\boldsymbol{s}]$ and the marginal probability $\mathcal{P}[\boldsymbol{x}] = \mathcal{P}[\boldsymbol{x}^1, \dots, \boldsymbol{x}^n]$ of this vector in the full state space that can be directly computed from the master equation. In fact, it is this vector, which captures the states of all the components in the system, that carries the most information on the input signal $\boldsymbol{s}$, and thus has the largest mutual information. However, typically the downstream system cannot read out the states of all the components $X^1, \dots, X^n$. Often, the downstream system reads out only a few components or often even just one component, the "output component" $X^r$. The other components then mainly serve to transmit the information from the input $\boldsymbol{s}$ to this readout $X^r$. From the perspective of the downstream system, the other components are hidden. The natural quantity to measure the precision of information processing is then the mutual information $\mathrm{I}(\mathcal{S}; \mathcal{X}^r)$ between the input $\boldsymbol{s}$ and the output component's trajectory $\boldsymbol{x}^r$, not $\mathrm{I}(\mathcal{S}; \mathcal{X})$. The question then becomes how to compute $\mathcal{P}[\boldsymbol{x}^r]$ and $\mathcal{P}[\boldsymbol{x}^r|\boldsymbol{s}]$, from which $\mathrm{I}(\mathcal{S}; \mathcal{X}^r)$ can be obtained. Here, we present a scheme to achieve this.

As an example, consider a chemical reaction network with species $X^1, \dots, X^n$. Without loss of generality, we will assume that the $n$-th component is the output component, $X^r = X^n$. The other species $X^1, \dots, X^{n-1}$ are thus not part of the output, but only relay information from the input signal $\boldsymbol{s}$ to the output signal $\boldsymbol{x}^n$. To determine the mutual information $\mathrm{I}(\mathcal{S}, \mathcal{X})$ we need $\mathcal{P}[\boldsymbol{x}^n|\boldsymbol{s}]$, where $\boldsymbol{x}^n$ is the trajectory of only the readout component $X^n$. However, from the master equation we can only obtain an expression for the full conditional probability $\mathcal{P}[\boldsymbol{x}^1, \dots, \boldsymbol{x}^n|\boldsymbol{s}]$ of all components. To compute the value of $\mathcal{P}[\boldsymbol{x}^n|\boldsymbol{s}]$, we must perform the marginalization integral

$$\mathcal{P}[\boldsymbol{x}^n|\boldsymbol{s}] = \int \mathcal{D}[\boldsymbol{x}^1] \cdots \int \mathcal{D}[\boldsymbol{x}^{n-1}]\, \mathcal{P}[\boldsymbol{x}^1, \dots, \boldsymbol{x}^n|\boldsymbol{s}]\,. \tag{2.20}$$

We can compute this integral using a Monte Carlo scheme as described below and use the resulting estimate for $\mathcal{P}[\boldsymbol{x}^n|\boldsymbol{s}]$ to compute the mutual information using our technique presented in Section 2.1.2.

The marginalization of Eq. (2.20) entails integrating out degrees of freedom from a known joint probability distribution. In Eq. (2.8) we solved the analogous problem of obtaining the marginal probability $\mathcal{P}[\boldsymbol{x}]$ by integrating out the input trajectories

through the integral $\mathcal{P}[\boldsymbol{x}] = \int d\boldsymbol{s}\, \mathcal{P}[\boldsymbol{s}, \boldsymbol{x}] = \int d\boldsymbol{s}\, \mathcal{P}[\boldsymbol{s}]\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$. As described in Section 2.1.2, the integral from Eq. (2.8) can be computed via a Monte Carlo estimate by sampling many input trajectories from $\mathcal{P}[\boldsymbol{s}]$ and taking the average of the corresponding conditional probabilities $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}_i]$. We will show that in the case where there is no feedback from the readout component back to the other components, a completely analogous Monte Carlo estimate can be derived for Eq. (2.20).

More specifically, we can evaluate Eq. (2.20) via a direct Monte Carlo estimate under the condition that the stochastic dynamics of the other components $X^1, \dots, X^{n-1}$ are not influenced by $X^n$ (i.e., no feedback from the readout). Using the identity

$$\mathcal{P}[\boldsymbol{x}^1, \dots, \boldsymbol{x}^n|\boldsymbol{s}] = \mathcal{P}[\boldsymbol{x}^1, \dots, \boldsymbol{x}^{n-1}|\boldsymbol{s}]\, \mathcal{P}[\boldsymbol{x}^n|\boldsymbol{x}_i^1, \dots, \boldsymbol{x}_i^{n-1}, \boldsymbol{s}] \tag{2.21}$$

to rewrite the integrand in Eq. (2.20), we are able to represent the conditional probability $\mathcal{P}[\boldsymbol{x}^n|\boldsymbol{s}]$ as an average over the readout component's trajectory probability

$$\mathcal{P}[\boldsymbol{x}^n|\boldsymbol{s}] = \left\langle \mathcal{P}[\boldsymbol{x}^n|\boldsymbol{x}_i^1, \dots, \boldsymbol{x}_i^{n-1}, \boldsymbol{s}] \right\rangle_{\mathcal{P}[\boldsymbol{x}^1, \dots, \boldsymbol{x}^{n-1}|\boldsymbol{s}]} . \tag{2.22}$$

Thus, assuming that we can evaluate the conditional probability of the readout given all the other components, $\mathcal{P}[\boldsymbol{x}^n|\boldsymbol{x}_i^1, \dots, \boldsymbol{x}_i^{n-1}, \boldsymbol{s}]$, we arrive at the estimate

$$\mathcal{P}[\boldsymbol{x}^n|\boldsymbol{s}] \approx \frac{1}{M} \sum_{i=1}^{M} \mathcal{P}[\boldsymbol{x}^n|\boldsymbol{x}_i^1, \dots, \boldsymbol{x}_i^{n-1}, \boldsymbol{s}] \tag{2.23}$$

where the samples $\boldsymbol{x}_i^1, \dots, \boldsymbol{x}_i^{n-1}$ for $i = 1, \dots, M$ are drawn from $\mathcal{P}[\boldsymbol{x}^1, \dots, \boldsymbol{x}^{n-1}|\boldsymbol{s}]$. Notice that the derivation of this Monte Carlo estimate is fully analogous to the estimate in Eq. (2.8), but instead of integrating out the input trajectory $\boldsymbol{s}$ we integrate out the component trajectories $\boldsymbol{x}^1, \dots, \boldsymbol{x}^{n-1}$.

To obtain $\mathcal{P}[\boldsymbol{x}^n|\boldsymbol{x}_i^1, \dots, \boldsymbol{x}_i^{n-1}, \boldsymbol{s}]$ in Eqs. (2.22) and (2.23), we note that, in absence of feedback, we can describe the stochastic dynamics of the readout component $X^n$ as a jump process with time-dependent transition rates whose time-dependence arises from the trajectories of the other components $\boldsymbol{x}^1, \dots, \boldsymbol{x}^{n-1}$ and the input input $\boldsymbol{s}$. In effect, this is a driven jump process for $X^n$, driven by all upstream components $X^1, \dots, X^{n-1}$ and the input signal. Specifically, denoting $\boldsymbol{u} = (\boldsymbol{x}^1, \dots, \boldsymbol{x}^{n-1}, \boldsymbol{s})$ as the joint trajectory representing the history of all upstream components as well as the input signal, we can, as explained in Section 2.1.3, write the time dependent transition rate matrix $Q_t(\cdot|\boldsymbol{u})$ for the stochastic dynamics of $X^n$ and use Eq. (2.15) to compute $\mathcal{P}[\boldsymbol{x}^n|\boldsymbol{u}] = \mathcal{P}[\boldsymbol{x}^n|\boldsymbol{x}_i^1, \dots, \boldsymbol{x}_i^{n-1}, \boldsymbol{s}]$. Using Eq. (2.23), this then allows us to compute $\mathcal{P}[\boldsymbol{x}^n|\boldsymbol{s}]$.

Finally, to compute the mutual information $\mathrm{I}(\mathcal{S}; \mathcal{X}^n)$, e.g. using the estimate in Eq. (2.10), we additionally need to evaluate the marginal output probability $\mathcal{P}[\boldsymbol{x}^n]$.

This requires us to perform one additional integration over the space of input trajectories $\boldsymbol{s}$:

$$\begin{aligned}\mathcal{P}[\boldsymbol{x}^n] &= \int \mathcal{D}[\boldsymbol{s}]\, \mathcal{P}[\boldsymbol{s}]\mathcal{P}[\boldsymbol{x}^n|\boldsymbol{s}] \\ &= \langle \mathcal{P}[\boldsymbol{x}^n|\boldsymbol{s}]\rangle_{\mathcal{P}[\boldsymbol{s}]} \,.\end{aligned} \tag{2.24}$$

The corresponding Monte Carlo estimate is

$$\begin{aligned}\mathcal{P}[\boldsymbol{x}^n] &\approx \frac{1}{N}\sum_{i=1}^{N} \mathcal{P}[\boldsymbol{x}^n|\boldsymbol{s}_i] \\ &\approx \frac{1}{N}\sum_{i=1}^{N}\frac{1}{M}\sum_{j=1}^{M} \mathcal{P}[\boldsymbol{x}^n|\boldsymbol{x}_{ij}^1,\dots,\boldsymbol{x}_{ij}^{n-1},\boldsymbol{s}_i]\end{aligned} \tag{2.25}$$

where the input trajectories $\boldsymbol{s}_i$ follow $\mathcal{P}[\boldsymbol{s}]$ and the intermediate components $(\boldsymbol{x}_{ij}^1,\dots,\boldsymbol{x}_{ij}^{n-1})$, for $i = 1,\dots,N$ and $j = 1,\dots,M$, follow $\mathcal{P}[\boldsymbol{x}^1,\dots,\boldsymbol{x}^{n-1}|\boldsymbol{s}_i]$.

In summary, the scheme to obtain $\mathcal{P}[\boldsymbol{x}^n|\boldsymbol{u}]$ in the presence of hidden intermediate components is analogous to that used for computing $\mathcal{P}[\boldsymbol{x}]$ from $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$. In both cases, one needs to marginalize a distribution function by integrating out components. Indeed, the schemes presented here and in Section 2.1.2 are bona fide schemes to compute the mutual information between the input $\boldsymbol{s}$ and either the trajectory of the output component $\boldsymbol{x}^n$ or the full output $\boldsymbol{x}$. However, when the trajectories are sufficiently long or the stochastic dynamics are sufficiently complex, then the free-energy schemes of Chapter 3 may be necessary to enhance the efficiency of computing the marginalized distribution, $\mathcal{P}[\boldsymbol{x}]$ or $\mathcal{P}[\boldsymbol{x}^n|\boldsymbol{s}]$.

## 2.3 Dealing with Feedback

In principle all physical information processing systems exhibit feedback. The physical interaction needed to measure the input signal necessarily affects the incoming signal, and indeed, it follows that no information can be extracted from the input signal without any perturbation of the input dynamics. Often, it is assumed that the amplitude of such perturbations is comparatively small and thus that the feedback can safely be ignored.

Indeed, the PWS scheme was derived with the assumption of no feedback. In this section, we drop the assumption and will explicitly consider systems where the produced output perturbs the input, i.e. systems where the output feeds back onto the input. We will first discuss the additional problems that arise when computing the

mutual information for a system with feedback, and subsequently present a modified version of PWS that can be used to compute the trajectory mutual information for these systems.

### 2.3.1 Computing the Mutual Information with Feedback between Input and Output

PWS requires the computation of the trajectory likelihood $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$, a quantity that is not readily available for systems with feedback. Indeed, as already mentioned in Section 2.1.3, for a given input trajectory $\boldsymbol{s}$, the output dynamics are no longer described by a Markov process in a system with feedback, and therefore we cannot find an expression for $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ based on the master equation. This implies that for systems with feedback, PWS schemes cannot be used without modification. While it is generally not possible to derive an expression for the conditional probability $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ in systems with feedback, we often still can compute the joint probability density $\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]$ instead. Based on this quantity, we will present a modified PWS scheme to compute the mutual information for systems with feedback.

Specifically, since PWS is a model-based approach to compute the mutual information, when there is feedback from the output back to the input, we require a complete model of the combined system. Specifically, such a model must provide an expression for the joint probability $\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]$, describing the input dynamics and the interaction between input and output, including the feedback.

An estimate of the mutual information that only relies on the computation of joint probability densities $\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]$ can be obtained by expressing the mutual information as

$$\mathrm{I}(\mathcal{S}, \mathcal{X}) = \int \mathcal{D}[\boldsymbol{s}] \int \mathcal{D}[\boldsymbol{x}]\, \mathcal{P}[\boldsymbol{s}, \boldsymbol{x}] \ln \frac{\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]}{\mathcal{P}[\boldsymbol{s}]\, \mathcal{P}[\boldsymbol{x}]}\,. \tag{2.26}$$

Thus, the PWS scheme with feedback consists of the computation of

$$\mathrm{I}(\mathcal{S}, \mathcal{X}) = \left\langle \ln \frac{\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]}{\mathcal{P}[\boldsymbol{s}]\, \mathcal{P}[\boldsymbol{x}]} \right\rangle_{\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]} \tag{2.27}$$

which we want to estimate via a Monte Carlo average using samples from $\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]$. We see that while we don't need to evaluate the likelihood $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$, we now need to explicitly compute the joint density $\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]$, and two marginal densities, $\mathcal{P}[\boldsymbol{s}]$ and $\mathcal{P}[\boldsymbol{x}]$, for each Monte Carlo sample $(\boldsymbol{s}, \boldsymbol{x}) \sim \mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]$. While the joint density can be evaluated directly by assumption, each of the marginalized densities can only be computed using a nested Monte Carlo estimate.

Specifically, for PWS with feedback, we need to compute *two* marginalization integrals per Monte Carlo sample:

$$\mathcal{P}[\boldsymbol{s}] = \int \mathcal{D}[\boldsymbol{x}]\, \mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]\,, \tag{2.28}$$

and

$$\mathcal{P}[\boldsymbol{x}] = \int \mathcal{D}[\boldsymbol{s}]\, \mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]\,. \tag{2.29}$$

However, these marginalization integrals cannot be directly computed with the techniques described so far. Therefore, in the following subsection, we discuss how to compute marginalization integrals for systems with feedback.

Additionally, as discussed in Section 2.2, we may also need to integrate out internal components of the master equation even when the output feeds back onto these internal components. The technique discussed below can also be used in this case as a way to compute the marginalization integral in Eq. (2.20).

### 2.3.2 Marginalization Integrals for Systems with Feedback

Computing marginalization integrals in systems with feedback is harder than it is in the case without feedback. Specifically, we will show that it is not obvious how apply the Monte Carlo estimate from Eq. (2.8) to systems with feedback. Nevertheless, if the system with feedback can be decomposed into a non-interacting part and an interacting part that includes the feedback, it is often still possible to compute marginalization integrals. Below, we sketch the steps that are necessary in order to compute marginalization integrals for systems with feedback using such a decomposition.

For concreteness, we discuss how to compute

$$\mathcal{P}[\boldsymbol{x}] = \int \mathcal{D}[\boldsymbol{s}]\, \mathcal{P}[\boldsymbol{s}, \boldsymbol{x}] \tag{2.30}$$

as the prototype for a marginalization integral we want to compute. Unlike before, we now assume that $\boldsymbol{x}$ feeds back onto $\boldsymbol{s}$. That means that we have access to the joint distribution's density $\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]$, but not to the marginal density $\mathcal{P}[\boldsymbol{s}]$ or the conditional density $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$.

Formulated in the language of statistical physics, marginalization is equivalent to the computation of the free-energy difference $\Delta\mathcal{F}[\boldsymbol{x}] = \mathcal{F}[\boldsymbol{x}] - \mathcal{F}_0[\boldsymbol{x}]$ between two ensembles described by potentials $\mathcal{U}[\boldsymbol{s}, \boldsymbol{x}]$ and $\mathcal{U}_0[\boldsymbol{s}, \boldsymbol{x}]$. Previously, for systems without feedback, we chose these potentials to be $\mathcal{U}_0[\boldsymbol{s}, \boldsymbol{x}] = -\ln \mathcal{P}[\boldsymbol{s}]$ and $\mathcal{U}[\boldsymbol{s}, \boldsymbol{x}] = -\ln \mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]$ with the idea that $\mathcal{U}$ is the potential corresponding to the actual system and $\mathcal{U}_0$ is the potential of a reference system with known free energy.

Then, by computing the free-energy difference between the reference system and the actual system, we could compute the marginal probability $\mathcal{P}[\boldsymbol{x}]$.

However, in systems with feedback we face a problem. Note that the actual system is still described by the potential $\mathcal{U}[\boldsymbol{s},\boldsymbol{x}] = -\ln \mathcal{P}[\boldsymbol{s},\boldsymbol{x}]$, even with feedback. Yet, for the reference system described by $\mathcal{U}_0[\boldsymbol{s},\boldsymbol{x}]$ we cannot make the same choice as before, because the previous choice involved the marginal probability $\mathcal{P}[\boldsymbol{s}]$ which is not available with feedback.

Instead, we have to find an alternative expression for $\mathcal{U}_0[\boldsymbol{s},\boldsymbol{x}]$. To construct a suitable reference potential, we can use a decomposition of the full potential into three parts

$$\mathcal{U}[\boldsymbol{s},\boldsymbol{x}] = \mathcal{U}_S[\boldsymbol{s}] + \mathcal{U}_X[\boldsymbol{x}] + \Delta\mathcal{U}[\boldsymbol{s},\boldsymbol{x}] \tag{2.31}$$

where $\Delta\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]$ describes the features of the system that induce interaction, or correlation, between $\boldsymbol{s}$ and $\boldsymbol{x}$. The first two terms of the potential above, $\mathcal{U}_S[\boldsymbol{s}]+\mathcal{U}_X[\boldsymbol{x}]$, therefore describe a *non-interacting* version of the system, where the input and output are fully independent of each other. We want to use the potential of that non-interacting version as our expression for $\mathcal{U}_0$, i.e. $\mathcal{U}_0[\boldsymbol{s},\boldsymbol{x}] = \mathcal{U}_S[\boldsymbol{s}] + \mathcal{U}_X[\boldsymbol{x}]$. To be able to do so, we require that the partition function (normalization constant)

$$\mathcal{Z}_0[\boldsymbol{x}] = \int \mathcal{D}[\boldsymbol{s}]\, e^{-\mathcal{U}_0[\boldsymbol{s},\boldsymbol{x}]} \tag{2.32}$$

is known. In other words, we need to choose the decomposition in Eq. (2.31) such that the partition function Eq. (2.32) is known either analytically or numerically. If such a decomposition is found, we can compute the marginal probability $\mathcal{P}[\boldsymbol{x}]$ from the difference in free energy $\Delta\mathcal{F}[\boldsymbol{x}]$ between $\mathcal{U}$ and $\mathcal{U}_0$:

$$-\ln \mathcal{P}[\boldsymbol{x}] = \mathcal{F}[\boldsymbol{x}] = \mathcal{F}_0[\boldsymbol{x}] + \Delta\mathcal{F}[\boldsymbol{x}] \tag{2.33}$$

where $\mathcal{F}_0 = -\ln \mathcal{Z}_0[\boldsymbol{x}]$ is known. Because we have a known expression for $\mathcal{U}_0[\boldsymbol{s},\boldsymbol{x}]$, the free-energy difference $\Delta\mathcal{F}[\boldsymbol{x}]$ can now be computed using any of the techniques described in Section 3.1.

As an example for finding a decomposition like Eq. (2.31), let us consider the case where the joint system of input and output is described by a single master equation, i.e. we have a master equation with two components, $S$ which represents the input, and $X$ which represents the output. In such a system, information is transmitted if there exist transitions that change the copy number of $X$ with a rate that depends on the copy number of $S$. In terms of chemical reactions, $S \to S + X$ is an example for such a transition. In turn, this system exhibits feedback if at least one of the transitions that change the copy number of $S$ has a rate that depends on $X$, as for example with the reaction $S + X \to X$. Note that with such reactions, the dynamics of $S$ depend on the current copy number of $X$, and therefore we cannot evolve $S$

trajectories independently of $X$ trajectories, a consequence of feedback. Both of the reactions $S \to S + X$ and $S + X \to X$ introduce correlations between the $S$ and $X$ trajectories.

In a non-interacting system, such interactions between the input and output must be absent. Thus, a non-interacting version of the reaction system contains no single reaction that involves both $S$ and $X$. We will now describe how we can use that non-interacting version of the reaction system, to obtain the reference potential $\mathcal{U}_0[\boldsymbol{s}, \boldsymbol{x}]$. Since the input and output trajectories are completely independent in the non-interacting system, we can express the joint distribution's probability density as the product of the individual component's trajectory densities, $\mathcal{P}_0[\boldsymbol{s}, \boldsymbol{x}] = \mathcal{P}_0[\boldsymbol{s}]\, \mathcal{P}_0[\boldsymbol{x}]$. Note that $\mathcal{P}_0[\boldsymbol{s}]$ and $\mathcal{P}_0[\boldsymbol{x}]$ should not be confused with the marginal probabilities $\mathcal{P}[\boldsymbol{s}]$ and $\mathcal{P}[\boldsymbol{x}]$ of the *interacting* version of the reaction system, which must be computed using a marginalization integral. Since in the non-interacting version both $S$ and $X$ obey independent dynamics which are characterized by individual master equations, both $\mathcal{P}_0[\boldsymbol{s}]$ and $\mathcal{P}_0[\boldsymbol{x}]$ can be individually computed using Eq. (2.14). Thus, in this case, the non-interacting potential is $\mathcal{U}_0[\boldsymbol{s}, \boldsymbol{x}] = -\ln \mathcal{P}_0[\boldsymbol{s}] - \ln \mathcal{P}_0[\boldsymbol{x}]$ and, since the probability densities $\mathcal{P}_0[\boldsymbol{s}]$ and $\mathcal{P}_0[\boldsymbol{x}]$ are normalized, the corresponding partition function is $\mathcal{Z}_0 = 1$. Hence, for this reaction system, we can straightforwardly define a non-interacting version that can be used to obtain the reference potential $\mathcal{U}_0[\boldsymbol{s}, \boldsymbol{x}]$. Using the techniques described in Section 3.1, we can then compute the free-energy difference between $\mathcal{U}_0[\boldsymbol{s}, \boldsymbol{x}]$ and $\mathcal{U}[\boldsymbol{s}, \boldsymbol{x}] = -\ln \mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]$, where the latter potential describes the dynamics of the fully interacting system. Specifically, we can compute the marginal probabilities $\mathcal{P}[\boldsymbol{s}]$, $\mathcal{P}[\boldsymbol{x}]$ pertaining to the interacting system which are required for the mutual information estimate in Eq. (2.27).

In summary, for systems with feedback, we can compute marginalization integrals by specifying a reference potential $\mathcal{U}_0[\boldsymbol{s}, \boldsymbol{x}]$ by finding a non-interacting version of the system. However, barring a decomposition into interacting and non-interacting potentials, there is generally no unambiguous choice of the reference potential $\mathcal{U}_0[\boldsymbol{s}, \boldsymbol{x}]$ to compute the marginal probability $\mathcal{P}[\boldsymbol{x}]$. In fact, the optimal reference potential $\mathcal{U}_0[\boldsymbol{s}, \boldsymbol{x}]$ is likely to be system-specific. Still, if a suitable expression for $\mathcal{U}_0[\boldsymbol{s}, \boldsymbol{x}]$ can be found, we can make use of the techniques developed in Section 3.1 to compute marginal probability $\mathcal{P}[\boldsymbol{x}]$.

## 2.4 Discussion

In this chapter, we have described a general, practical, and flexible method that makes it possible to compute the mutual information between trajectories exactly. PWS is a Monte Carlo scheme based on the exact computation of trajectory probabilities. We showed how to compute exact trajectory probabilities from the master

equation and thus how to use PWS for any system described by a master equation. Since the master equation is employed in many fields and in particular provides an exact stochastic model for well-mixed chemical reaction dynamics, PWS is very broadly applicable.

However, it must be noted that PWS cannot be used to directly obtain the mutual information between trajectories from experimental data, in contrast to model-free (yet approximate) methods such as K-nearest-neighbors estimators [81, 94], decoding-based information estimates [59], or schemes that compute the mutual information from the data within the Gaussian framework [107]. PWS requires a (generative) model based on a master equation or Langevin equation. Yet, in Chapter 6, we will show how PWS can be combined with machine learning to obtain the rate directly from time-series data.

We have applied PWS to compute the mutual information rate in steady state, but PWS can be used equally well to study systems out of steady state. For such systems a (non-)equilibrium initial condition $P(s_0, x_0)$ must be specified in addition to the stochastic dynamics of input trajectories $\mathcal{P}[\boldsymbol{s}]$. These distributions are defined by the (experimental) setup and lead to a well-defined output distribution $\mathcal{P}[\boldsymbol{x}]$ when the system is coupled to the input. Thus, a steady state is no prerequisite for the application of PWS to study the trajectory mutual information.

Overall, PWS is a general framework for computing the mutual information between trajectories. Because of its flexibility and simplicity, we envision that it will become an important and reliable tool for studying information transmission in dynamic stochastic systems.

# 3 More Efficient Variants of PWS

*In the previous chapter, we introduced Path Weight Sampling (PWS), a computational approach capable of providing exact information rate estimates for any stochastic system. However, the direct implementation of PWS becomes inefficient for complex systems and long trajectories due to the high dimensionality of trajectory space. To overcome these limitations, we present two improved PWS variants in this chapter, inspired by free-energy estimation techniques from statistical physics. First, Rosenbluth-Rosenbluth PWS (RR-PWS) leverages computational strategies developed for polymer chemical potential calculations, enhancing efficiency for sampling in trajectory spaces. Second, Thermodynamic Integration PWS (TI-PWS) applies thermodynamic integration combined with trajectory space MCMC sampling, inspired by transition path sampling. We benchmark these methods using a simple coupled birth-death model, comparing the effectiveness of all three PWS variants against analytical results and the Gaussian approximation.*

The contents of this chapter have been published in *Phys. Rev. X* **13**, 041017 (2023) [152].

To quantify information transmission, be it in a natural or engineered information processing system, we need to be able to compute the mutual information between input and output trajectories, from which the information transmission rate can be obtained. However, because of the high dimensionality of the trajectory space, computing the mutual information between trajectories is exceedingly difficult, if not impossible, because the conventional non-parametric binning approach to estimate the required trajectory probability distributions cannot be used. Indeed, except for very simple models, the mutual information between trajectories is typically computed using approximations that are often uncontrolled. In the previous chapter, we introduced a computational scheme called Path Weight Sampling (PWS), which, for the first time, makes it possible to compute the information rate exactly, for any stochastic system.

Yet, the scheme presented in that chapter, Direct PWS, becomes inefficient for more complex systems and longer trajectories. The reason is that the number of possible trajectories increases exponentially with trajectory length, leading to a corresponding increase in the variance of the estimate. Hence, for long trajectories the PWS estimate may prove to be computationally infeasible. To address this issue, we describe two improved variants of PWS in this section, both based on free-energy estimators from statistical physics.

Specifically, in Section 3.2 we present *Rosenbluth-Rosenbluth PWS* (RR-PWS) which exploits the observation that the computation of $\mathcal{P}[\boldsymbol{x}]$ is analogous to the calculation of the (excess) chemical potential of a polymer, for which efficient methods have been developed [167, 69, 55]. In Section 3.3, we present *Thermodynamic Integration PWS* (TI-PWS) which is based on the classic free energy estimation technique of thermodynamic integration [54, 62, 127] in conjunction with a trajectory space MCMC sampler using ideas from transition path sampling [19].

In Section 3.4 we apply PWS to a well-known model system. It consists a simple pair of coupled birth-death processes which allows us to test the efficiency of the three PWS variants, as well as to compare the PWS results with analytical results from the Gaussian approximation [194] and the technique by Duso and Zechner [46].

## 3.1 Marginalizing in Trajectory Space

PWS evaluates the mutual information $I(\mathcal{S}, \mathcal{X})$ from the marginal entropy $H(\mathcal{X})$ and the conditional entropy $H(\mathcal{X}|\mathcal{S})$, see Eq. (2.2). Of these two entropies, the conditional one can be efficiently computed using the scheme described in the previous chapter, and as used in DPWS. However, obtaining the marginal entropy $H(\mathcal{X}) = -\int \mathcal{D}[\boldsymbol{x}]\ \mathcal{P}[\boldsymbol{x}] \ln \mathcal{P}[\boldsymbol{x}]$ is much more challenging. Indeed, the compu-

tationally most expensive part of Direct PWS is the evaluation of the marginalization integral $\mathcal{P}[\boldsymbol{x}_i] = \int \mathcal{D}[\boldsymbol{s}]\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}_i]$ which needs to be performed for every sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$. Consequently, the computational efficiency of this marginalization is essential for the overall performance.

Marginalization is a general term to denote an operation where one or more variables are integrated out of a joint probability distribution. For instance, we obtain the marginal probability distribution $\mathcal{P}[\boldsymbol{x}]$ from $\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]$ by computing the integral

$$\mathcal{P}[\boldsymbol{x}] = \int \mathcal{D}[\boldsymbol{s}]\, \mathcal{P}[\boldsymbol{s}, \boldsymbol{x}] = \int \mathcal{D}[\boldsymbol{s}]\, \mathcal{P}[\boldsymbol{s}]\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]\,. \tag{3.1}$$

In DPWS, we use Eq. (2.8) to compute $\mathcal{P}[\boldsymbol{x}]$ which involves generating independent input trajectories from $\mathcal{P}[\boldsymbol{s}]$. However, this this is not the optimal Monte Carlo technique to perform the marginalization. The generated input trajectories are independent from the output trajectory $\boldsymbol{x}$. Thus, we ignore the causal connection between $\boldsymbol{s}$ and $\boldsymbol{x}$, and we typically end up sampling trajectories $\boldsymbol{s}^\star$ whose likelihoods $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}^\star]$ are very small. Then, most sampled trajectories have small integral weights, and only very few samples provide a significant contribution to the average. The variance of the result is then very large because the effective sample size is much smaller than the total sample size. The use of $\mathcal{P}[\boldsymbol{s}]$ as the sampling distribution is thus only practical in cases where the dependence of the output on the input is not too strong. It follows that this sampling scheme works best when the mutual information is not too large [1].

This is a well known Monte Carlo sampling problem and a large number of techniques have been developed to solve it. The two variants of our scheme, RR-PWS and TI-PWS, both make use of ideas from statistical physics for the efficient computation of free energies.

[1] Indeed, the mutual information $\mathrm{I}(\mathcal{S}, \mathcal{X})$ precisely quantifies how strong the statistical dependence is between the trajectory-valued random variables $\mathcal{S}$ and $\mathcal{X}$. From its definition $\mathrm{I}(\mathcal{S}, \mathcal{X}) = \mathrm{H}(\mathcal{S}) - \mathrm{H}(\mathcal{S}|\mathcal{X})$ we can understand more clearly how this affects the efficiency of the Monte Carlo estimate. Roughly speaking, $\mathrm{H}(\mathcal{S})$ is related to the number of distinct trajectories $\boldsymbol{s}$ that can arise from the dynamics given by $\mathcal{P}[\boldsymbol{s}]$, while $\mathrm{H}(\mathcal{S}|\mathcal{X})$ is related to the number of distinct trajectories $\boldsymbol{s}$ that could have lead to a specific output $\boldsymbol{x}$, on average. Therefore, if the mutual information is very large, the difference between these two numbers is very large, and consequently the number of overall distinct trajectories is much larger than the number of distinct trajectories compatible with output $\boldsymbol{x}$. Now, if we generate trajectories according to the dynamics given by $\mathcal{P}[\boldsymbol{s}]$, with overwhelming probability we generate a trajectory $\boldsymbol{s}$ which is not compatible with the output trajectory $\boldsymbol{x}$, and therefore $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] \approx 0$. Hence, the effective number of samples $M_{\text{eff}}$ is much smaller than the actual number of generated trajectories $M$, i.e. $M_{\text{eff}} \ll M$. We therefore only expect the estimate in Eq. (2.8) to be reliable when computing the mutual information for systems where it is not too high. Thus, strikingly, the difficulty of computing the mutual information is proportional to the magnitude of the mutual information itself.

| | |
|---|---|
| $\mathcal{P}[\boldsymbol{s},\boldsymbol{x}]$ | $e^{-\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]}$ |
| $\mathcal{P}[\boldsymbol{s}]$ | $\frac{1}{\mathcal{Z}_0[\boldsymbol{x}]}e^{-\mathcal{U}_0[\boldsymbol{s}]}$ |
| $\mathcal{P}[\boldsymbol{s}\|\boldsymbol{x}]$ | $\frac{1}{\mathcal{Z}[\boldsymbol{x}]}e^{-\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]}$ |
| 1 | $\mathcal{Z}_0[\boldsymbol{x}]$ |
| $\mathcal{P}[\boldsymbol{x}]$ | $\mathcal{Z}[\boldsymbol{x}]$ |
| $\mathcal{P}[\boldsymbol{x}\|\boldsymbol{s}]$ | $e^{-\Delta\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]}$ |

Table 3.1: Translation to the notation of statistical physics. The definitions of $\mathcal{U}$ and $\mathcal{U}_0$ that are used here are given in Eqs. (3.2) and (3.6).

To understand how these ideas can be applied to compute the marginal probability $\mathcal{P}[\boldsymbol{x}]$, it is helpful to rephrase the marginalization integral in Eq. (3.1) in the language of statistical physics. In this language, $\mathcal{P}[\boldsymbol{x}]$ corresponds to the normalization constant, or partition function, of the Boltzmann distribution for the potential[2]

$$\mathcal{U}[\boldsymbol{s},\boldsymbol{x}] = -\ln \mathcal{P}[\boldsymbol{s},\boldsymbol{x}]\,. \tag{3.2}$$

In Eq. (3.2), $\boldsymbol{s}$ is interpreted as a variable in the configuration space, while $\boldsymbol{x}$ acts as an auxiliary variable, i.e., a parameter. Note that both $\boldsymbol{s}$ and $\boldsymbol{x}$ still represent trajectories. For this potential, the partition function is given by

$$\mathcal{Z}[\boldsymbol{x}] = \int \mathcal{D}[\boldsymbol{s}]\; e^{-\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]}\,. \tag{3.3}$$

The integral only runs over the configuration space, i.e. we integrate only with respect to $\boldsymbol{s}$. By inserting the expression for $\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]$, we see that the partition function is exactly equal to the marginal probability of the output, i.e. $\mathcal{Z}[\boldsymbol{x}] = \mathcal{P}[\boldsymbol{x}]$. The free energy is given by

$$\mathcal{F}[\boldsymbol{x}] = -\ln \mathcal{Z}[\boldsymbol{x}] = -\ln \mathcal{P}[\boldsymbol{x}]\,. \tag{3.4}$$

[2]Note that while the distribution $\exp(-\mathcal{U}[\boldsymbol{s},\boldsymbol{x}])/Z[\boldsymbol{x}]$ resembles the equilibrium distribution of a canonical ensemble from statistical mechanics, this does not imply that we are studying systems in thermal equilibrium. Indeed, PWS is used to quantify information transmission in systems driven out of equilibrium by the input signal. Thus, the notation introduced in this section is merely a mathematical reformulation of the marginalization integral to make the analogy to statistical physics apparent. We assign no physical meaning to the potentials and free energies. Also note that we set $k_\mathrm{B}T = 1$ since temperature is irrelevant here.

In statistical physics it is well known that the free energy cannot be directly measured from a simulation. Instead, one estimates the free-energy difference

$$\Delta\mathcal{F}[\boldsymbol{x}] = \mathcal{F}[\boldsymbol{x}] - \mathcal{F}_0[\boldsymbol{x}] = -\ln\frac{\mathcal{Z}[\boldsymbol{x}]}{\mathcal{Z}_0[\boldsymbol{x}]} \tag{3.5}$$

between the system and a reference system with known free energy $\mathcal{F}_0[\boldsymbol{x}]$. The reference system can be freely chosen and is usually defined using a Boltzmann distribution for a convenient reference potential $\mathcal{U}_0[\boldsymbol{s},\boldsymbol{x}]$. In our case, a natural choice of reference potential is

$$\mathcal{U}_0[\boldsymbol{s},\boldsymbol{x}] = -\ln\mathcal{P}[\boldsymbol{s}] \tag{3.6}$$

with the corresponding partition function being simply

$$\mathcal{Z}_0[\boldsymbol{x}] = \int\mathcal{D}[\boldsymbol{s}]\,\mathcal{P}[\boldsymbol{s}] = 1\,. \tag{3.7}$$

The reference free energy therefore is zero ($\mathcal{F}_0[\boldsymbol{x}] = -\ln\mathcal{Z}_0[\boldsymbol{x}] = 0$). Hence, the free-energy difference is

$$\Delta\mathcal{F}[\boldsymbol{x}] = \mathcal{F}[\boldsymbol{x}] = -\ln\mathcal{P}[\boldsymbol{x}]\,. \tag{3.8}$$

Note that in our case the reference potential $\mathcal{U}_0[\boldsymbol{s},\boldsymbol{x}] = -\ln\mathcal{P}[\boldsymbol{s}]$ does not depend on the output trajectory $\boldsymbol{x}$, i.e. $\mathcal{U}_0[\boldsymbol{s},\boldsymbol{x}] \equiv \mathcal{U}_0[\boldsymbol{s}]$. It describes a *non-interacting* version of our input-output system where the input trajectories evolve independently of the fixed output trajectory $\boldsymbol{x}$.

What is the interaction between the output $\boldsymbol{x}$ and the input trajectory ensemble? We define the interaction potential $\Delta\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]$ through

$$\mathcal{U}[\boldsymbol{s},\boldsymbol{x}] = \mathcal{U}_0[\boldsymbol{s}] + \Delta\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]\,. \tag{3.9}$$

The interaction potential makes it apparent that the distribution of $\boldsymbol{s}$ corresponding to the potential $\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]$ is biased by $\boldsymbol{x}$ with respect to the distribution corresponding to the reference potential $\mathcal{U}_0[\boldsymbol{s}]$. By inserting the expressions for $\mathcal{U}_0[\boldsymbol{s}]$ and $\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]$ into Eq. (3.9) we see that

$$\begin{aligned}\Delta\mathcal{U}[\boldsymbol{s},\boldsymbol{x}] &= -\ln\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] \\ &= -\ln\mathrm{P}(x_0|s_0) - \int_0^T dt\,\mathcal{L}_t[\boldsymbol{s},\boldsymbol{x}]\end{aligned} \tag{3.10}$$

where $\mathcal{L}_t[\boldsymbol{s},\boldsymbol{x}]$ is given by Eq. (2.16) and can be directly computed from the master equation. This expression illustrates that the interaction of the output trajectory $\boldsymbol{x}$

with the ensemble of input trajectories is characterized by the trajectory likelihood $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$.

To summarize, in this section we have introduced notation (see Table 3.1) showing that computing a marginalization integral is equivalent to the computation of a free-energy difference. This picture allows us to distinguish between two ensembles for $\boldsymbol{s}$, the *non-interacting* ensemble distributed according to $\exp(-\mathcal{U}_0[\boldsymbol{s}]) = \mathcal{P}[\boldsymbol{s}]$, and the *interacting* ensemble distributed according to $\exp(-\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]) \propto \mathcal{P}[\boldsymbol{s}|\boldsymbol{x}]$. With these notions we can rewrite the brute force estimate in Direct PWS (Chapter 2) as

$$\mathcal{P}[\boldsymbol{x}] = \frac{\mathcal{Z}[\boldsymbol{x}]}{\mathcal{Z}_0[\boldsymbol{x}]} = \langle e^{-\Delta\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]}\rangle_0 \tag{3.11}$$

where the notation $\langle\cdots\rangle_0$ refers to an average with respect to the non-interacting ensemble. By inserting the expressions for $\mathcal{U}_0$ and $\Delta\mathcal{U}$, it is easy to verify that this estimate is equivalent to Eq. (2.8).

The more advanced variants of PWS introduced below leverage the analogy with statistical physics to improve the efficiency of marginalization. RR-PWS draws ideas from soft condensed matter simulations by recognizing that Eq. (3.5) has the same form as the excess chemical potential of a polymer for which efficient computation techniques have been developed [167, 122]. Meanwhile, TI-PWS is inspired by Transition Path Sampling (TPS) for sampling rare trajectories [19] and uses thermodynamic integration for free-energy estimation.

## 3.2 RR-PWS

In Rosenbluth-Rosenbluth PWS we compute the free-energy difference $\Delta\mathcal{F}$ between the ideal system $\mathcal{U}_0$ and $\mathcal{U}$ in a *single* simulation just like in the brute force method. However, instead of generating $\boldsymbol{s}$ trajectories in an uncorrelated fashion according to $\exp(-\mathcal{U}_0[\boldsymbol{s}]) = \mathcal{P}[\boldsymbol{s}]$, we bias our sampling distribution towards $\exp(-\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]) \propto \mathcal{P}[\boldsymbol{s}|\boldsymbol{x}]$ to reduce the sampling problems found in DPWS.

The classical scheme for biasing the sampling distribution in polymer physics is due to Rosenbluth and Rosenbluth [155] in their study of self-avoiding chains. A substantial improvement of the Rosenbluth algorithm was achieved by Grassberger, by generating polymers using pruning and enrichment steps, thereby eliminating configurations that do not significantly contribute to the average. This scheme is known as the pruned-enriched Rosenbluth method, or PERM [69]. While PERM is much more powerful than the standard Rosenbluth algorithm, its main drawback is that it requires careful tuning of the pruning and enrichment schedule to achieve optimal convergence. Therefore we have opted to use a technique that is similar in

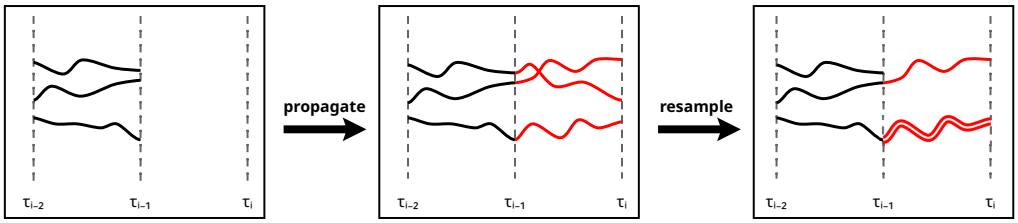



Figure 3.1: Illustration of one step of the bootstrap particle filter in RR-PWS. We start with a set of trajectories $\boldsymbol{s}^k_{[0,i-1]}$ with time span $[\tau_0, \tau_{i-1}]$ (left panel). In the next step we propagate these trajectories forward in time to $\tau_i$, according to $\mathcal{P}[\boldsymbol{s}]$ (central panel). Then we resample the trajectories according to the Boltzmann weights of their most recent segments, effectively eliminating or duplicating individual segments. An example outcome of the resampling step is shown in the right panel where the bottom trajectory was duplicated and one of the top trajectories was eliminated. These steps are repeated for each segment, until a set of input trajectories of the desired length is generated. The intermediate resampling steps bias the trajectory distribution from $\mathcal{P}[\boldsymbol{s}]$ towards $\mathcal{P}[\boldsymbol{s}|\boldsymbol{x}]$.

spirit to PERM but requires less tuning, the bootstrap particle filter [67]. We will describe how to use PWS with a particle filter below. That said, we want to stress that the particle filter can easily be replaced by PERM or other related methods [145]. Also schemes inspired by variants of Forward Flux Sampling [3, 12] could be developed.

### 3.2.1 Bootstrap Particle Filter

In the methods discussed above, a polymer is grown monomer by monomer. In a continuous-time Markov process this translates to trajectories being grown segment by segment. To define the segments, we choose a time discretization $0 < \tau_1 < \tau_2 < \cdots < \tau_{n-1} < T$. Thus, each trajectory $\boldsymbol{s}$ of duration $T$ consists of $n$ segments where we denote the segment between $\tau_i$ and $\tau_j$ by $\boldsymbol{s}_{[i,j]}$ (we define $\tau_0 = 0$ and $\tau_n = T$). The particle filter uses the following procedure to grow an ensemble of trajectories segment by segment:

1. Generate $M$ starting points $s_0^1, \ldots, s_0^M$ according to the initial condition of the input signal $\mathrm{P}(s_0)$.

2. Iterate for $i = 1, \ldots, n$:

a) Starting with an ensemble of $M$ partial trajectories of duration $\tau_{i-1}$ (if $i = 1$ an ensemble of starting points) which we label $\boldsymbol{s}^k_{[0,i-1]}$ for $k = 1, \dots, M$:

$$\left(\boldsymbol{s}^1_{[0,i-1]}, \dots, \boldsymbol{s}^M_{[0,i-1]}\right), \tag{3.12}$$

propagate each trajectory (or each starting point) forward in time from $\tau_{i-1}$ to $\tau_i$. Propagation is performed according to the natural dynamics of $\boldsymbol{s}$, i.e. generating a new segment $\boldsymbol{s}^k_{[i-1,i]}$ with probability

$$p_i^{\text{gen}}(k) = \mathcal{P}\left[\boldsymbol{s}^k_{[i-1,i]} | \boldsymbol{s}^k_{[0,i-1]}\right] = e^{-\mathcal{U}_0\left[\boldsymbol{s}^k_{[i-1,i]}\right]} \tag{3.13}$$

for $k = 1, \dots, M$.

b) Compute the Boltzmann weight

$$U_i^k = \Delta\mathcal{U}[\boldsymbol{s}^k_{[i-1,i]}, \boldsymbol{x}_{[i-1,i]}] \tag{3.14}$$

of each new segment. This Boltzmann weight of a segment from $\tau_{i-1}$ to $\tau_i$ can be expressed as

$$U_i^k = -\delta_{1i} \ln \mathrm{P}(x_0|s_0) - \int_{\tau_{i-1}}^{\tau_i} dt\, \mathcal{L}_t[\boldsymbol{s}^k_{[i-1,i]}, \boldsymbol{x}_{[i-1,i]}], \tag{3.15}$$

see Eq. (3.10), and is therefore straightforward to compute from the master equation.

c) Sample $M$ times from the distribution

$$p_i^{\text{select}}(k) = \frac{e^{-U_i^k}}{w_i} \tag{3.16}$$

where the Rosenbluth weight $w_i$ is defined as

$$w_i = \sum_{k=1}^{M} e^{-U_i^k}. \tag{3.17}$$

This sampling procedure yields $M$ randomly drawn indices $\ell_i^1, \dots, \ell_i^M$. Each $\ell_i^k$ is an index that lies in the range from $1, \dots, M$ and that points to one of the trajectories that have been generated up to $\tau_i$. To continue the sampling procedure, we relabel the indices such that the resampled set of trajectories is defined by $\tilde{\boldsymbol{s}}^k_{[0,i]} \leftarrow \boldsymbol{s}^{\ell_i^k}_{[0,i]}$ for $k = 1, \dots, M$. The list $\left(\tilde{\boldsymbol{s}}^1_{[0,i]}, \dots, \tilde{\boldsymbol{s}}^M_{[0,i]}\right)$ is subsequently used as the input for the next iteration of the algorithm.

The normalized Rosenbluth factor of the final ensemble is then given by

$$\mathcal{W} = \prod_{i=1}^{n} \frac{w_i}{M} \,. \tag{3.18}$$

As shown in Section 3.2.3, we can derive an *unbiased* estimate for the desired ratio $\mathcal{Z}[\boldsymbol{x}]/\mathcal{Z}_0[\boldsymbol{x}] = \mathcal{P}[\boldsymbol{x}]$ based on the Rosenbluth factor:

$$\hat{\mathcal{P}}[\boldsymbol{x}] = \mathrm{P}(x_0)\, \mathcal{W} \tag{3.19}$$

with $\mathrm{P}(x_0)$ being the probability of the initial output $x_0$. The particle filter can therefore be integrated into the DPWS algorithm to compute the marginal density $\mathcal{P}[\boldsymbol{x}]$, substituting the brute-force estimate given in Eq. (2.8). We call the resulting algorithm to compute the mutual information *RR-PWS*.

### 3.2.2 Intuitive Justification of the Algorithm

First note that steps 1 and 2(a) of the procedure above involve just propagating $M$ trajectories in parallel, according to $\mathcal{P}[\boldsymbol{s}] = \exp(-\mathcal{U}_0[\boldsymbol{s}])$. The interesting steps are 2(b-c) where we eliminate or duplicate some of the trajectories according to the Boltzmann weights of the most recent segment. Note, that in general the list of indices $(\ell_i^1, \dots, \ell_i^M)$ that are sampled in step 2(c) will contain duplicates ($\ell_i^k = \ell_i^{k'}$ for $k \neq k'$), thus cloning the corresponding trajectory. Concomitantly, the indices $\ell_i^1, \dots, \ell_i^M$ may not include every original index $1, \dots, M$, therefore eliminating some trajectories. Since indices of trajectories with high Boltzmann weight are more likely to be sampled from Eq. (3.16), this scheme biases the sampling distribution towards trajectories with large Boltzmann weight, ensuring that we are only spending computational effort on propagating trajectories which contribute significantly to the marginalization integral.

Hence, at its heart, the particle filter is an importance sampling scheme. It produces samples that are biased towards the ideal importance sampling distribution $\exp(-\mathcal{U}_0[\boldsymbol{s}])\exp(-\Delta\mathcal{U}[\boldsymbol{s}, \boldsymbol{x}])$, i.e., towards to the Boltzmann distribution of the interacting ensemble. The Rosenbluth factor $\mathcal{W}$ represents the importance sampling weight which would be required to correct for the sampling bias when computing averages using the sampled trajectories. Importantly for our case, the Rosenbluth factor can also be used to estimate the marginal probability $\mathcal{P}[\boldsymbol{x}]$. For illustration of the algorithm, one iteration of the particle filter is presented schematically in Fig. 3.1.

### 3.2.3 Detailed Justification

This subsection justifies the marginal probability estimate shown in Eq. (3.19) in greater detail, and may be skipped on first reading. We show that the bootstrap particle filter provides a consistent estimator for the marginal probability $\mathcal{P}[\boldsymbol{x}]$, or, equivalently, the ratio of partition functions $\mathcal{Z}[\boldsymbol{x}]/\mathcal{Z}_0[\boldsymbol{x}]$. The result that this estimate is also unbiased is more difficult to establish; a proof is given by Del Moral [36].

We structure our justification of the particle filter into three steps. We first give a brief description of how a resampling procedure can generally be used to generate samples approximating a target distribution when only samples from a different distribution are available. Secondly, we use these insights to explain how the resampling procedure used in the particle filter generates trajectories whose distribution is biased towards $\mathcal{P}[\boldsymbol{s}|\boldsymbol{x}]$, even though we only generate trajectories according to $\mathcal{P}[\boldsymbol{s}]$. Finally, we use this result to show that the particle filter provides a consistent estimate for $\mathcal{P}[\boldsymbol{x}]$.

#### Sampling and resampling

Sampling and then resampling is a strategy to use samples $\boldsymbol{s}^1, \dots, \boldsymbol{s}^M$ from a given prior distribution $f[\boldsymbol{s}]$ to generate *approximate* [3] samples from a different distribution of interest, with density proportional to the product $h[\boldsymbol{s}] = f[\boldsymbol{s}]g[\boldsymbol{s}]$. In general, $h[\boldsymbol{s}]$ is not normalized, and we denote the corresponding normalized probability density by $\hat{h}[\boldsymbol{s}] = h[\boldsymbol{s}]/\int \mathcal{D}[\boldsymbol{s}]h[\boldsymbol{s}]$. To generate samples from $\hat{h}[\boldsymbol{s}]$, we assign each of the existing samples from $f[\boldsymbol{s}]$ a normalized weight

$$W^k = \frac{g[\boldsymbol{s}^k]}{\sum_{j=1}^{M} g[\boldsymbol{s}^j]}\,. \tag{3.20}$$

Then, by sampling from the discrete set $\{\boldsymbol{s}^1, \dots, \boldsymbol{s}^M\}$ according to the assigned weights $W^1, \dots, W^M$, we select samples that are approximately distributed according to $\hat{h}[\boldsymbol{s}]$. Indeed, for $M \to \infty$ the distribution of the resulting samples approaches the density $\hat{h}[\boldsymbol{s}]$ [171]. We use resampling at each iteration of the algorithm of Section 3.2 to regularly prune those trajectories with low overall contribution to the marginalization integral.

[3] The samples generated through resampling are only approximate because they are limited to the discrete set $\{\boldsymbol{s}^1, \dots, \boldsymbol{s}^M\}$, which was originally drawn from the prior distribution $f[\boldsymbol{s}]$. The resampling process assigns weights based on the target distribution, which are used to select from that set, but it does not generate entirely new samples directly from the target. Therefore, the sampled points do not constitute draws from the target distribution unless $M \to \infty$.

### Particle filter

In the bootstrap particle filter, at each iteration, we start with a set of trajectories $\boldsymbol{s}^1_{[0,i-1]}, \dots, \boldsymbol{s}^M_{[0,i-1]}$. In each iteration of the particle filter, the goal is to produce a set of elongated trajectories (from time step $(i-1) \to i$) whose distribution tends towards $\mathcal{P}[\boldsymbol{s}_{[0,i]}|\boldsymbol{x}_{[0,i]}]$. By iterating such a procedure, we can generate a set of trajectories distributed approximately according to $\mathcal{P}[\boldsymbol{s}_{[0,n]}|\boldsymbol{x}_{[0,n]}]$ for any $n > 1$. Thus, the particle filter is a biased sampling scheme which provides an approximation of $\mathcal{P}[\boldsymbol{s}_{[0,n]}|\boldsymbol{x}_{[0,n]}]$. Moreover, using the particle filter we can also compute the corresponding importance weights which can be used to compute the marginal probability $\mathcal{P}[\boldsymbol{x}_{[0,n]}]$ for $n = 1, 2, \dots$.

We now take a closer look at one iteration of the particle filter. Start with a set of trajectories in $[\tau_0, \tau_{i-1}]$, denoted by $\left\{\boldsymbol{s}^1_{[0,i-1]}, \dots, \boldsymbol{s}^M_{[0,i-1]}\right\}$. These trajectories are then propagated forward to time $\tau_i$, by adding a new segment $\boldsymbol{s}^k_{[i-1,i]}$ to the trajectory $\boldsymbol{s}^k_{[0,i-1]}$ for $k = 1, \dots, M$. Each new segment is generated from the distribution $\mathcal{P}[\boldsymbol{s}^k_{[i-1,i]}|\boldsymbol{s}^k_{[0,i-1]}]$ such that the propagation step results in a set of trajectories $\{\boldsymbol{s}^1_{[0,i]}, \dots, \boldsymbol{s}^M_{[0,i]}\}$, distributed according to $f[\boldsymbol{s}_{[0,i]}] = \mathcal{P}[\boldsymbol{s}_{[0,i]}|\boldsymbol{x}_{[0,i-1]}]$.

Next, we resample from the set of trajectories, with the goal of producing a set of trajectories distributed according to the target density $\hat{h}[\boldsymbol{s}] = \mathcal{P}[\boldsymbol{s}_{[0,i]}|\boldsymbol{x}_{[0,i]}]$. Thus, we have to find the appropriate weighting function $g[\boldsymbol{s}_{[0,i]}]$ in order to approximately produce samples according to the target distribution. By choosing $g[\boldsymbol{s}_{[0,i]}] = \exp\left\{-\Delta\mathcal{U}[\boldsymbol{s}_{[i-1,i]}, \boldsymbol{x}_{[i-1,i]}]\right\} = \mathcal{P}[\boldsymbol{x}_{[i-1,i]}|\boldsymbol{x}_{[0,i-1]}, \boldsymbol{s}_{[0,i]}]$, we generate normalized weights

$$W_i^k = \frac{\mathcal{P}[\boldsymbol{x}_{[i-1,i]}|\boldsymbol{x}_{[0,i-1]}, \boldsymbol{s}^k_{[0,i]}]}{\sum_{j=1}^M \mathcal{P}[\boldsymbol{x}_{[i-1,i]}|\boldsymbol{x}_{[0,i-1]}, \boldsymbol{s}^j_{[0,i]}]}, \tag{3.21}$$

cf. Eq. (3.20). Note that this is the same choice of weighting function as in Section 3.2, Eq. (3.16). By comparison with the notation used there, we see that the Boltzmann factors $U_i^k$ and Rosenbluth weights $w_i$ were defined such that we can express the normalized weight equivalently as

$$W_i^k = \frac{e^{-U_i^k}}{w_i}. \tag{3.22}$$

Why is this choice of weighting function the correct one? First, observe that resampling with the normalized weights of Eq. (3.21) produces samples approximately distributed according to

$$\begin{aligned} h[\boldsymbol{s}_{[0,i]}] &= f[\boldsymbol{s}_{[0,i]}] g[\boldsymbol{s}_{[0,i]}] \\ &= \mathcal{P}[\boldsymbol{s}_{[0,i]}|\boldsymbol{x}_{[0,i-1]}]\, \mathcal{P}[\boldsymbol{x}_{[i-1,i]}|\boldsymbol{x}_{[0,i-1]}, \boldsymbol{s}_{[0,i]}]. \end{aligned} \tag{3.23}$$

What remains to be shown is that this density $h[\boldsymbol{s}_{[0,i]}]$, when normalized, becomes the desired target distribution $\mathcal{P}[\boldsymbol{s}_{[0,i]}|\boldsymbol{x}_{[0,i]}]$.

To do so, we need to rewrite the expression for $g[\boldsymbol{s}_{[0,i]}] = \mathcal{P}[\boldsymbol{x}_{[i-1,i]}|\boldsymbol{x}_{[0,i-1]}, \boldsymbol{s}_{[0,i]}]$ using Bayes' theorem

$$g[\boldsymbol{s}_{[0,i]}] = \frac{\mathcal{P}[\boldsymbol{s}_{[0,i]}|\boldsymbol{x}_{[0,i-1]}, \boldsymbol{x}_{[i-1,i]}]\, \mathcal{P}[\boldsymbol{x}_{[i-1,i]}|\boldsymbol{x}_{[0,i-1]}]}{\mathcal{P}[\boldsymbol{s}_{[0,i]}|\boldsymbol{x}_{[0,i-1]}]}\,. \tag{3.24}$$

Notice that the first term of the numerator can be written as $\mathcal{P}[\boldsymbol{s}_{[0,i]}|\boldsymbol{x}_{[0,i]}]$. After inserting this result into Eq. (3.23), we obtain

$$h[\boldsymbol{s}_{[0,i]}] = \mathcal{P}[\boldsymbol{s}_{[0,i]}|\boldsymbol{x}_{[0,i]}]\, \mathcal{P}[\boldsymbol{x}_{[i-1,i]}|\boldsymbol{x}_{[0,i-1]}]\,. \tag{3.25}$$

The second term in this product is a constant, since $\boldsymbol{x}$ is fixed. The first term is a normalized probability density for $\boldsymbol{s}_{[0,i]}$. Therefore we find that the normalized density corresponding to $h[\boldsymbol{s}_{[0,i]}]$ is

$$\hat{h}[\boldsymbol{s}_{[0,i]}] = \mathcal{P}[\boldsymbol{s}_{[0,i]}|\boldsymbol{x}_{[0,i]}]\,. \tag{3.26}$$

Consequently, this is the distribution that is approximated by the set of trajectories at the end of the $i$-th iteration of the particle filter, which is what we wanted to show. At its heart, the particle filter is therefore an algorithm to produce samples that are approximately distributed according to $\mathcal{P}[\boldsymbol{s}|\boldsymbol{x}]$.

### Marginal probability estimate

We now use these insights to derive an estimate of $\mathcal{P}[\boldsymbol{x}]$. We start by noting that the marginal density of the $i$-th output segment, $\mathcal{P}[\boldsymbol{x}_{[i-1,i]}|\boldsymbol{x}_{[0,i-1]}]$, is given by

$$\begin{aligned}
&\mathcal{P}[\boldsymbol{x}_{[i-1,i]}|\boldsymbol{x}_{[0,i-1]}] \\
&= \int \mathcal{D}[\boldsymbol{s}_{[0,i]}]\, \mathcal{P}[\boldsymbol{x}_{[i-1,i]}, \boldsymbol{s}_{[0,i]}|\boldsymbol{x}_{[0,i-1]}] \\
&= \int \mathcal{D}[\boldsymbol{s}_{[0,i]}]\, \mathcal{P}[\boldsymbol{s}_{[0,i]}|\boldsymbol{x}_{[0,i-1]}]\, g[\boldsymbol{s}_{[0,i]}]\,.
\end{aligned} \tag{3.27}$$

The third line follows from the definition of $g[\boldsymbol{s}_{[0,i]}] = \mathcal{P}[\boldsymbol{x}_{[i-1,i]}|\boldsymbol{x}_{[0,i-1]}, \boldsymbol{s}_{[0,i]}]$. Hence, we find that the probability $\mathcal{P}[\boldsymbol{x}_{[i-1,i]}|\boldsymbol{x}_{[0,i-1]}]$ can be expressed as the average

$$\mathcal{P}[\boldsymbol{x}_{[i-1,i]}|\boldsymbol{x}_{[0,i-1]}] = \left\langle g[\boldsymbol{s}_{[0,i]}] \right\rangle_{\mathcal{P}[\boldsymbol{s}_{[0,i]}|\boldsymbol{x}_{[0,i-1]}]}\,. \tag{3.28}$$

In principle, this average can be computed using a Monte Carlo scheme, using trajectories generated from $\mathcal{P}[\boldsymbol{s}_{[0,i]}|\boldsymbol{x}_{[0,i-1]}]$. Notice that at each iteration of the particle

filter, we *do* dispose of a set of trajectories $\boldsymbol{s}^1_{[0,i]}, \dots, \boldsymbol{s}^M_{[0,i]}$ which are approximately distributed according to $\mathcal{P}[\boldsymbol{s}_{[0,i]}|\boldsymbol{x}_{[0,i-1]}]$ above. Therefore, we can compute the average Eq. (3.28) directly from the trajectories that are present for each iteration of the particle filter. With the notation from Section 3.2, using $g[\boldsymbol{s}^k_{[0,i]}] = \exp(-U^k_i)$, we thus obtain the estimate

$$\mathcal{P}[\boldsymbol{x}_{[i-1,i]}|\boldsymbol{x}_{[0,i-1]}] \approx \frac{1}{M}\sum_{k=1}^{M} e^{-U_i^k} = \frac{w_i}{M}\,. \tag{3.29}$$

The probability of the entire output trajectory $\mathcal{P}[\boldsymbol{x}]$ is given by the product

$$\mathcal{P}[\boldsymbol{x}] = \mathrm{P}(x_0)\mathcal{P}[\boldsymbol{x}_{[0,1]}|x_0]\cdots\mathcal{P}[\boldsymbol{x}_{[n-1,n]}|\boldsymbol{x}_{[0,n-1]}] \tag{3.30}$$

where $\mathrm{P}(x_0)$ is the probability of the initial output state $x_0$ which is assumed to be known. In conclusion, we arrive at the following estimate for the marginal output probability

$$\hat{\mathcal{P}}[\boldsymbol{x}] = \mathrm{P}(x_0)\prod_{i=1}^{n}\frac{w_i}{M} \tag{3.31}$$

which is precisely Eq. (3.19).

### 3.2.4 Tuning the Particle Filter

For the efficiency of the particle filter, it is important to carefully choose the number of segments $n$. When segments are very short (i.e., when $n$ is large), the accumulated weights (Eq. (3.15)) tend to differ very little between newly generated segments $\boldsymbol{s}^k_{[i-1,i]}$. Hence, the pruning and enrichment of the segments is dominated by noise. In contrast, when the segments are very long, the distribution of Boltzmann weights $U^k_i$ becomes very wide. Then only a small number of segments contribute substantially to the corresponding Rosenbluth weight $w_i$. Hence, to correctly choose $n$, we need a measure that quantifies the variance in the trajectory weights of the $n$ particles. To this end, we follow Martino et al. [105] and introduce an effective sample size (ESS)

$$M_i^{(\mathrm{eff})} = \frac{w_i^2}{\sum_{k=1}^{M}\left(e^{-U_i^k}\right)^2}, \tag{3.32}$$

which lies in the range $1 \le M_i^{(\mathrm{eff})} \le M$; $M_i^{(\mathrm{eff})} = 1$ if one trajectory has a much higher weight than all the others and $M_i^{(\mathrm{eff})} = M$ if all trajectories have the same weight. As a rule of thumb, we resample only when the $M_i^{(\mathrm{eff})}$ drops below $M/2$. Additionally, as recommended in Ref. [44], we use the *systematic sampling* algorithm to randomly

draw the indices in step 2(c) which helps to reduce the variance; we find, however, the improvement over simple sampling is very minor. Using these techniques, the only parameter that needs to be chosen by hand for the particle filter is the ensemble size $M$.

## 3.3 TI-PWS

Thermodynamic integration PWS (TI-PWS), is based on the analogy of marginalization integrals with free-energy computations. As before, we view the problem of computing the marginal probability $\mathcal{P}[\boldsymbol{x}]$ as equivalent to that of computing the free-energy difference between ensembles defined by the potentials $\mathcal{U}_0[\boldsymbol{s},\boldsymbol{x}]$ and $\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]$, respectively. For TI-PWS, we define a potential $\mathcal{U}_\theta[\boldsymbol{s},\boldsymbol{x}]$ with a continuous parameter $\theta \in [0,1]$ that allows us to transform the ensemble from $\mathcal{U}_0$ to $\mathcal{U} = \mathcal{U}_1$. The corresponding partition function is

$$\mathcal{Z}_\theta[\boldsymbol{x}] = \int \mathcal{D}[\boldsymbol{s}]\, e^{-\mathcal{U}_\theta[\boldsymbol{s},\boldsymbol{x}]}\,. \tag{3.33}$$

For instance, for $0 \le \theta \le 1$, we can define our potential as

$$\mathcal{U}_\theta[\boldsymbol{s},\boldsymbol{x}] = \mathcal{U}_0[\boldsymbol{s},\boldsymbol{x}] + \theta\,\Delta\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]\,, \tag{3.34}$$

such that $e^{-\mathcal{U}_\theta[\boldsymbol{s},\boldsymbol{x}]} = \mathcal{P}[\boldsymbol{s}]\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]^\theta$. Note that this is the simplest choice for a continuous transformation between $\mathcal{U}_0$ and $\mathcal{U}_1$, but by no means the only one. For reasons of computational efficiency, it can be beneficial to choose a different path between $\mathcal{U}_0$ and $\mathcal{U}_1$, depending on the specific system [62]. Here we will not consider other paths however, and derive the thermodynamic integration estimate for the potential given in Eq. (3.34).

To derive the thermodynamic integration estimate for the free-energy difference, we first compute the derivative of $\ln \mathcal{Z}_\theta[\boldsymbol{x}]$ with respect to $\theta$:

$$\begin{aligned}\frac{\partial}{\partial\theta}\ln\mathcal{Z}_\theta[\boldsymbol{x}] &= \frac{1}{\mathcal{Z}_\theta[\boldsymbol{x}]}\frac{\partial}{\partial\theta}\int \mathcal{D}[\boldsymbol{s}]\, e^{-\mathcal{U}_\theta[\boldsymbol{s},\boldsymbol{x}]} \\ &= -\left\langle \frac{\partial\mathcal{U}_\theta[\boldsymbol{s},\boldsymbol{x}]}{\partial\theta}\right\rangle_\theta \\ &= -\langle\Delta\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]\rangle_\theta\,.\end{aligned} \tag{3.35}$$

Thus, the derivative of $\ln\mathcal{Z}_\theta[\boldsymbol{x}]$ is an average of the Boltzmann weight with respect to $\mathcal{P}_\theta[\boldsymbol{s}|\boldsymbol{x}]$ which is the ensemble distribution of $\boldsymbol{s}$ given by

$$\mathcal{P}_\theta[\boldsymbol{s}|\boldsymbol{x}] = \frac{1}{\mathcal{Z}_\theta[\boldsymbol{x}]}e^{-\mathcal{U}_\theta[\boldsymbol{s},\boldsymbol{x}]}\,. \tag{3.36}$$

Integrating Eq. (3.35) with respect to $\theta$ leads to the formula for the free-energy difference

$$\Delta\mathcal{F}[\boldsymbol{x}] = -\int_0^1 d\theta\ \langle \Delta\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]\rangle_\theta \tag{3.37}$$

which is the fundamental identity underlying thermodynamic integration.

To compute the free-energy difference using Eq. (3.37), we evaluate the integral with respect to $\theta$ numerically using Gaussian quadrature, while the inner average $\langle \Delta\mathcal{U}[\boldsymbol{s},\boldsymbol{x}]\rangle_\theta$ is computed using MCMC simulations. To perform MCMC simulations in trajectory space we use ideas from transition path sampling (TPS). Specifically, we define a MCMC proposal distribution for trajectories using forward shooting and backward shooting [19]. These proposals regrow either the end, or the beginning of a trajectory, respectively. A proposal is accepted according to the Metropolis criterion [114]. Since the efficiency of MCMC samplers strongly depends on the proposal moves that are employed, we are certain that better MCMC estimates are possible with more sophisticated proposal distributions.

### 3.3.1 MCMC Sampling in Trajectory Space

TI-PWS relies on the computation of averages with respect to the ensembles corresponding to the interaction parameter $\theta$, given by $\mathcal{P}_\theta[\boldsymbol{s}|\boldsymbol{x}] \propto \exp(-\mathcal{U}_\theta[\boldsymbol{s},\boldsymbol{x}])$. Sampling from this family of distributions using the SSA (Gillespie) algorithm is not possible. Instead, in this section, we show different ways of how to implement a Markov Chain Monte Carlo (MCMC) sampler in trajectory space to generate correctly distributed trajectories.

We can build an MCMC sampler in trajectory space using the Metropolis-Hastings algorithm. To create a Markov chain in trajectory space, we need to find a suitable proposal kernel, that generates a new trajectory $\boldsymbol{s}'$ from a given trajectory $\boldsymbol{s}$ with probability $T(\boldsymbol{s}\to\boldsymbol{s}')$. We accept the proposal using the Metropolis criterion with probability

$$A(\boldsymbol{s}',\boldsymbol{s}) = \min\left(1, e^{\mathcal{U}_\theta[\boldsymbol{s},\boldsymbol{x}]-\mathcal{U}_\theta[\boldsymbol{s}',\boldsymbol{x}]}\frac{T(\boldsymbol{s}'\to\boldsymbol{s})}{T(\boldsymbol{s}\to\boldsymbol{s}')}\right) \tag{3.38}$$

to create a chain of trajectories with stationary distribution given by $\mathcal{P}_\theta[\boldsymbol{s}|\boldsymbol{x}] = e^{-\mathcal{U}_\theta[\boldsymbol{s},\boldsymbol{x}]}/\mathcal{Z}_\theta[\boldsymbol{x}]$ for $0 \le \theta \le 1$. To ensure efficient convergence of the resulting Markov chain to its stationary distribution, the proposal kernel must balance two conflicting requirements. To efficiently explore the state space per unit amount of CPU time, the proposed trajectory $\boldsymbol{s}'$ must be sufficiently different from the original trajectory $\boldsymbol{s}$, while at the same time it should not be so different that the acceptance probability becomes too low. Thus, the design of the proposal kernel is crucial for an efficient MCMC sampler, and we will discuss various strategies to

create trial trajectories. Since different types of trial moves can easily be combined in a Metropolis-Hastings algorithm, the most efficient samplers often incorporate multiple complementary proposal strategies to improve the exploration speed of the trajectory space.

The simplest (and naïve) proposal kernel is to generate an entirely new trajectory $\boldsymbol{s}'$ independent of $\boldsymbol{s}$, by sampling directly from $\mathcal{P}[\boldsymbol{s}]$ using the SSA. Hence, the transition kernel is given by $T(\boldsymbol{s} \to \boldsymbol{s}') = \mathcal{P}[\boldsymbol{s}']$ and a proposal $\boldsymbol{s} \to \boldsymbol{s}'$ is accepted with probability

$$\begin{aligned} A(\boldsymbol{s}', \boldsymbol{s}) &= \min\left(1, e^{\mathcal{U}_\theta[\boldsymbol{s},\boldsymbol{x}]-\mathcal{U}_\theta[\boldsymbol{s}',\boldsymbol{x}]}\frac{\mathcal{P}[\boldsymbol{s}]}{\mathcal{P}[\boldsymbol{s}']}\right) \\ &= \min\left(1, \frac{\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}']^\theta}{\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]^\theta}\right) \end{aligned} \tag{3.39}$$

where the second line follows by using the definition of $\mathcal{U}_\theta[\boldsymbol{s}, \boldsymbol{x}]$ given in Eq. (3.34). Although this simple scheme is correct, it should not be used in practice to compute $\mathcal{P}[\boldsymbol{x}]$. Indeed, on would get a better estimate of $\mathcal{P}[\boldsymbol{x}]$ by just using the same number of independent sample trajectories from $\mathcal{P}[\boldsymbol{s}]$ and using the brute-force scheme in Eq. (2.8) without taking the detour of using MCMC to estimate the normalization constant.

Instead, an idea from transition path sampling is to only regenerate a part of the old trajectory as part of the proposal kernel [38]. By not regenerating the entire trajectory, the new trajectory $\boldsymbol{s}'$ is going to be correlated with the original trajectory $\boldsymbol{s}$, improving the acceptance rate. The simplest way to generate trial trajectories using a partial update is a move termed *forward shooting* in which a time point $\tau$ along the existing trajectory $\boldsymbol{s}$ is randomly selected, and a new trajectory segment is regrown from this point to the end, resulting in the proposal $\boldsymbol{s}'$. Since the new segment is generated according to the unbiased input statistics, the acceptance probability for the proposed trajectory is given by Eq. (3.39), i.e., the same as if the entire trajectory had been regenerated. If the input dynamics given by $\mathcal{P}[\boldsymbol{s}]$ are time-reversible, we can also perform a *backward shooting* move. Here, the beginning of $\boldsymbol{s}$ is replaced by a new segment that is generated backwards in time. Assuming that the initial condition is the input's steady state distribution, the corresponding acceptance probability of the backward shooting move is again given by Eq. (3.39). Using these two moves we create an MCMC sampler where both ends of the trajectory are flexible, and thus if the trajectory is not too long, the chain will quickly relax to its stationary distribution.

For long trajectories it can prove to be a problem that the middle section is too inflexible when the proposal moves only regenerate either the beginning or the end of a trajectory. Therefore, one could additionally try to incorporate mid-section regrowth to make sure that also the middle parts of the trajectory become flexible. To

regrow a middle segment with duration $\tau$ of a trajectory $\boldsymbol{s}$, we have to generate a new segment of duration $\tau$ according to the stochastic dynamics given by $\mathcal{P}[\boldsymbol{s}]$ but with the additional condition that we have to connect *both* endpoints of the new segment to the existing trajectory. Although the starting point of the segment can be freely chosen, the challenge is to ensure that the end point of the new segment satisfies the end-point constraint. Stochastic processes that generate trajectories under the condition of hitting a specific point after a given duration $\tau$ are called stochastic bridging processes.

The simplest way to generate trajectories from a bridging process is by generating a trajectory segment of length $\tau$ from the normal stochastic process and rejecting the segment if it does not hit the correct end point [79]. Clearly, this strategy is only feasible for very short segments and when the state space is discrete, as otherwise almost every generated segment will be rejected due to not hitting the correct end point. To avoid this problem, more efficient algorithms have been developed to simulate stochastic bridges for some types of stochastic processes. For diffusion processes, bridges can be simulated efficiently by introducing a guiding term into the corresponding Langevin equation [199]. For jump processes, bridges can be simulated using particle filters [66], by a weighted stochastic simulation algorithm (wSSA) [64], or using random time-discretization (uniformization) [79].

Further techniques to create a trajectory space MCMC samplers have been developed in the literature. Crooks [33] describes a scheme to create MCMC moves for trajectories evolving in non-equilibrium dynamics, by making MCMC moves to change the trajectories' noise histories. In the Particle Markov Chain Monte Carlo (PMCMC) algorithm, proposal trajectories are generated using a particle filter and accepted with an appropriate Metropolis criterion [4]. Another class of efficient samplers for Markov jump processes can be built using uniformization [149].

## 3.4 Simple Application and Benchmark

To demonstrate the power of our framework and illustrate how the techniques of the previous sections can be used in practice, we apply PWS to a simple chemical reaction network. We consider a linearly coupled birth-death process which has been studied previously using a Gaussian model [194], and by Duso and Zechner [46] using an approximate technique, and we compare our results with these studies. This simple birth-death system serves to illustrate the main ideas of our approach and also highlights that linear systems can be distinctly non-Gaussian.

The code used to produce the PWS estimates was written in the Julia programming language [16] and has been made freely available [150, 151]. For performing stochastic simulations we use the DifferentialEquations.jl package [147].

We consider a stochastic process $\varnothing \rightleftharpoons \mathrm{X}$ of species X which is created at rate $\rho(t)$ and decays with constant rate $\mu$ per copy of X. This system receives information from an input signal that modulates the birth rate $\rho(t)$. For simplicity, we assume it is given by

$$\rho(t) = \rho_0 s(t) \tag{3.40}$$

where $\rho_0$ is a constant and $s(t)$ is the input copy number at time $t$. This is a simple model for gene expression, where the rate of production of a protein X is controlled by a transcription factor S, and X itself has a characteristic decay rate. The input trajectories $s(t)$ themselves are generated via a separate birth-death process $\varnothing \rightleftharpoons \mathrm{S}$ with production rate $\kappa$ and decay rate $\lambda$.

We compute the trajectory mutual information for this system as a function of the trajectory duration $T$ of the input and output trajectories. For $T \to \infty$, the trajectory mutual information is expected to increase linearly with $T$, since, on average, every additional output segment contains the same additional amount of information on the input trajectory. Because we are interested in the mutual information in steady state, the initial states $(s_0, x_0)$ were drawn from the stationary distribution $\mathrm{P}(s_0, x_0)$. This distribution was obtained using a Gaussian approximation. This does not influence the asymptotic rate of increase of the mutual information, but leads to a nonzero mutual information already for $T = 0$.

Figure 3.2 shows the mutual information as a function of the trajectory duration $T$. We compare the three PWS variants and two approximate schemes. One is that of Duso and Zechner [46]. To apply it, we used the code publicly provided by the authors[4], and to avoid making modifications to this code, we chose a fixed initial condition ($s_0 = x_0 = 50$) which causes the mutual information to be zero for $T = 0$. The figure also shows the analytical result of a Gaussian model [194], obtained using the linear-noise approximation (see Section 3.6.1).

We find that the efficiency of the respective PWS variants depends on the duration of the input-output trajectories. For short trajectories all PWS variants yield very similar estimates for the mutual information. However, for longer trajectories the estimates of DPWS and, to a smaller degree, TI-PWS diverge, because of poor sampling of the trajectory space in the estimate of $\mathcal{P}[\boldsymbol{x}]$. For longer trajectories, the estimate becomes increasingly dominated by rare trajectories, which make an exceptionally large contribution to the average of $\mathcal{P}[\boldsymbol{x}]$. Missing these rare trajectories with a high weight tends to increase the marginal entropy $\mathrm{H}(\mathcal{X})$ [see Eq. (2.9)], and thereby the mutual information; indeed, the estimates of DPWS and TI-PWS are higher than that of RR-PWS. For brute-force DPWS, the error decreases as we increase the number $M$ of input trajectories per output trajectory used to estimate $\mathcal{P}[\boldsymbol{x}]$. Similarly, for TI-PWS the error decreases as we use more MCMC samples for

[4] https://github.com/zechnerlab/PathMI

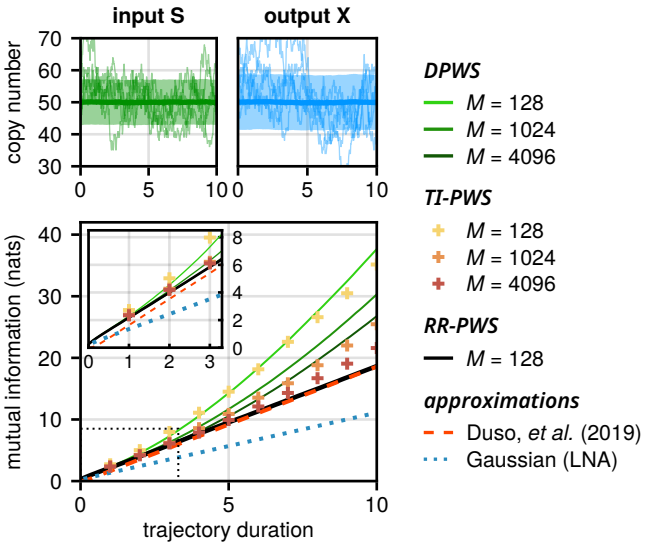



Figure 3.2: Comparing different schemes to compute the mutual information as a function of trajectory duration for a simple coupled birth-death process with rates $\kappa = 50, \lambda = 1, \rho_0 = 10, \mu = 10$ and steady-state initial condition. The top panels show example trajectories of input and output as well as the mean (solid line) and standard deviation (shaded region). Below, the mutual information is shown as a function of trajectory duration. The inset shows an enlarged version of the dotted rectangle near the origin. For short trajectories all PWS estimates agree. Yet, for longer trajectories, DPWS and TI-PWS require a much larger number of input trajectories $M$ for computing $\mathcal{P}[\boldsymbol{x}]$ than RR-PWS to converge. Results for the three PWS variants are compared with the Duso and Zechner [46] estimate, and with the linear noise approximation from Ref. [194]. We find excellent agreement between the Duso scheme and RR-PWS. The Gaussian linear noise approximation systematically underestimates the mutual information. All PWS estimates, as well as the Duso approximation were computed using $N = 10^4$ samples from $\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]$.

the marginalization scheme. For the RR-PWS, however, already for $M = 128$ the estimate has converged; we verified that a further increase of $M$ does not change the results.

We also find excellent agreement between the RR-PWS estimate and the approximate result of Duso and Zechner [46]. Only very small deviations are visible in Fig. 3.2. These deviations are mostly caused by the different choice for the initial conditions. In RR-PWS, the initial conditions are drawn from the stationary distribution, while in the Duso scheme they are fixed, such that the mutual information computed with RR-PWS is finite while that computed with the Duso scheme is zero. Yet, as the trajectory duration $T$ increases, the Duso estimate slowly "catches up" with the RR-PWS result.

Fig. 3.2 also shows that although the Gaussian model matches the PWS result for $T = 0$, it systematically underestimates the mutual information for trajectories of finite duration $T > 0$. Interestingly, this is not a consequence of small copy-number fluctuations: increasing the average copy number does not significantly improve the Gaussian estimate.[5]

The different approaches for computing the marginal probability $\mathcal{P}[\boldsymbol{x}]$ lead to different computational efficiencies of the respective PWS schemes. In Fig. 3.3, as a benchmark, we show the magnitude of the error of the different PWS estimates in relation to the required CPU time. Indeed, as expected, the computation of the marginal probability poses problems for long trajectories when using the brute force DPWS scheme. More interestingly, while TI-PWS improves the estimate of the mutual information, the improvement is not dramatic. Unlike the brute-force scheme, thermodynamic integration does make it possible to generate input trajectories $\boldsymbol{s}$ that are correlated with the output trajectories $\boldsymbol{x}$, but it still overestimates the mutual information for long trajectories unless a very large number of MCMC samples are used.

The RR-PWS implementation evidently outperforms the other estimates for this system. The regular resampling steps ensure that we mostly sample input trajectories $\boldsymbol{s}$ with non-vanishing likelihood $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$, thereby avoiding the sampling problem from DPWS. Moreover, sequential Monte Carlo techniques such as RR-PWS and FFS [3] have a considerable advantage over MCMC techniques in trajectory

[5] A detailed analysis of this observation was carried out in a (currently unpublished) collaborative work with Anne-Lena Moor and Christoph Zechner from the MPI-CBG in Dresden [120]. This work demonstrates that the discrepancy arises because all the information on the input signal is contained in the output species' production process, which is catalyzed by the input, rather than in the decay process of the output, which occurs independently of the input. The PWS result captures this distinction by using a fully discrete approach. In contrast, the Gaussian estimate of the linear-noise approximation misses this distinction because in the noise term for the output the contributions from the production and decay reactions are added together. We also comment further on this matter in Chapter 5.

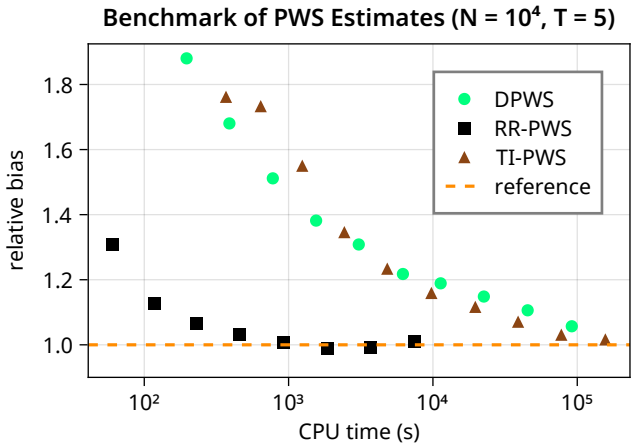



Figure 3.3: Comparing estimation bias for the different PWS variants in relation to their CPU time requirements. Each dot represents a single mutual information estimate with $N = 10^4$ samples for output trajectories of duration $T = 5$. Almost all the CPU time of a PWS estimate is spent on the computation of the marginal probability $\mathcal{P}[\boldsymbol{x}]$. The bias of the marginal probability estimate can be reduced by using a larger number $M$ of sampled input trajectories to compute the marginalization integral, which also increases the required CPU time. The RR-PWS estimate converges much faster than the estimate of DPWS and TI-PWS. For DPWS and TI-PWS, the dots represents estimates ranging from $M = 2^5$ to $M = 2^{14}$, for RR-PWS ranging from $M = 2^3$ to $M = 2^{10}$. As the baseline of zero bias we use the converged result from the RR-PWS estimates.

sampling. With MCMC path sampling, we frequently make small changes to an existing trajectory such that the system moves slowly in path space, leading to poor statistics. In contrast, in RR-PWS we generate new trajectories from scratch, segment by segment, and these explore the trajectory space much faster.

## 3.5 Discussion

Aside from Direct PWS introduced in Chapter 2, we developed two additional variants of PWS, capitalizing on the connection between information theory and statistical physics. Specifically, the computation of the mutual information requires the evaluation of the marginal probability of individual output trajectories $\mathcal{P}[\boldsymbol{x}]$. This corresponds to the computation of a partition function in statistical physics, $\mathcal{P}[\boldsymbol{x}] = \int \mathcal{P}[\boldsymbol{s}]\,\mathcal{P}[\boldsymbol{s}]\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$. RR-PWS and TI-PWS are based on techniques from polymer and rare-event simulations to make the computation of the marginal trajectory probability more efficient.

The different PWS variants share some characteristics yet also differ in others. DPWS and RR-PWS are static Monte Carlo schemes in which the trajectories are generated independently from the previous ones. These methods are similar to static polymer sampling schemes like PERM [69] and rare-event methods like DFFS or BG-FFS [3]. In contrast, TI-PWS is a dynamic Monte Carlo scheme, where a new trajectory is generated from the previous trajectory. In this regard, this method is similar to the CBMC scheme for polymer simulations [168] and the TPS [19], TIS [200], and RB-FFS [3] schemes to harvest transition paths. The benefit of static schemes is that the newly generated trajectories are uncorrelated from the previous ones, which means that they are less likely to get stuck in certain regions of path space. Concomitantly, they tend to diffuse faster through the configuration space. Indeed, TI-PWS suffers from a problem that is also often encountered in TPS or TIS, which is that the middle sections of the trajectories move only slowly in their perpendicular direction. Tricks that have been applied to TPS and TIS to solve this problem, such as parallel tempering, could also be of use here [203].

Another distinction is that RR-PWS generates all the trajectories in the ensemble simultaneously yet segment by segment, like DFFS, while DPWS and TI-PWS generate only one full trajectory at the time, similar to RB-FFS, BG-FFS, and also TPS and TIS. Consequently, RR-PWS, like DFFS, faces the risk of *genetic drift*, which means that, after sufficiently many resampling steps, most paths of the ensemble will originate from the same initial seed. Thus, when continuing to sample new segments, the old segments that are far in the past become essentially fixed, which makes it possible to miss important paths in the RR-PWS sampling procedure. As in DFFS, the risk of genetic drift in RR-PWS can be mitigated by increasing the initial

number of path segments. Although we did not employ this trick here, we found that RR-PWS was by far the most powerful scheme of the three variants studied.

Nonetheless, we expect that DPWS and TI-PWS become more efficient in systems that respond to the input signal with a significant delay $\tau$. In these cases, the weight of a particular output trajectory depends on the degree to which the dynamics of the output trajectory correlates with the dynamics of the intput trajectory a time $\tau$ earlier. Because in RR-PWS a new segment of an output trajectory is generated based on the corresponding segment of the input trajectory that spans the same time-interval, it may therefore miss these correlations between the dynamics of the output and that of the input a time $\tau$ earlier. In contrast, DPWS and TI-PWS generate full trajectories one at the time, and are therefore more likely to capture these correlations. Also the machine-learning based approach for determining the optimal importance sampling distribution $q[\boldsymbol{s}|\boldsymbol{x}]$ presented in Chapter 6 (Section 6.1.4) is likely to prove useful in these scenarios with complex temporal dependences between the input and output trajectories.

## 3.6 Supplementary Information

### 3.6.1 Gaussian Approximation of the Linear System

We derive the Gaussian approximation of the simple reaction system used in Section 3.4. We recall the elementary biochemical reaction motif consisting of four reactions

$$0 \xrightarrow{\kappa} \mathrm{S}, \quad \mathrm{S} \xrightarrow{\lambda} 0, \quad \mathrm{S} \xrightarrow{\rho} \mathrm{S} + \mathrm{X}, \quad \mathrm{X} \xrightarrow{\mu} 0 \tag{3.41}$$

with input $S$ and output $X$. This reaction motif is a simple model for gene expression, where the rate of production of a protein $X$ is controlled by a transcription factor $S$, and X itself has a characteristic decay rate. The dynamics of $S$ are given by a constant birth rate and a constant (per-molecule) decay rate.

We compute the covariance functions of this model which are then used to derive Gaussian signal statistics, and allow us to compute the Gaussian information rate. Specifically, we assume that the process is sampled at a sampling rate $\nu$, with $S_1, \ldots, S_n$ being the sequence of sampled inputs and $X_1, \ldots, X_n$ being the sequence of outputs in time. We can describe the dynamics of the input and output as fluctuations around the mean, i.e. we write $\delta S_i = S_i - \langle S_i \rangle$ and $\delta X_i = X_i - \langle X_i \rangle$. In the limit of large copy numbers, due to the central limit theorem, the distribution of these fluctuations become Gaussian [60].

In particular, let $\boldsymbol{Z} = (\delta S_1, \ldots, \delta S_n, \delta X_1, \ldots, \delta X_n)^T$ be the concatenation of the

input and output sequences. Then, the distribution of $\boldsymbol{Z}$ is multivariate normal, i.e.,

$$\mathrm{P}(\boldsymbol{Z}=\boldsymbol{z}) = \frac{1}{\sqrt{(2\pi)^{2n}|\Sigma|}} e^{-\frac{1}{2}(\boldsymbol{z}^T\Sigma^{-1}\boldsymbol{z})} \tag{3.42}$$

where the covariance Matrix $\Sigma \in \mathbb{R}^{2n\times 2n}$ has the following block structure

$$\Sigma = \begin{pmatrix} \Sigma_{SS} & \Sigma_{XS} \\ \Sigma_{SX} & \Sigma_{XX} \end{pmatrix}. \tag{3.43}$$

Here $\Sigma_{SS}$ and $\Sigma_{XX}$ are the (auto-)covariance matrices of the input and the output, respectively, whereas $\Sigma_{SX} = \Sigma_{XS}^T$ contain the cross-covariances. The matrix elements are given by $\Sigma_{AB}^{ij} = \langle \delta A_i \delta B_j \rangle = C_{AB}(t_i - t_j)$ where $C_{AB}(t)$ denote the (cross-)covariance functions and $t_1, \ldots, t_n$ are the sampling times. Thus, the full statistics of the trajectories are determined from the (cross-)covariance functions.

Since the reaction scheme in Eq. (3.41) features only first-order reactions, the covariance functions can be calculated explicitly using the regression theorem [206, 60]. For $t \geq 0$, we obtain the following expressions for the covariance functions:

$$C_{SS}(t) = \sigma_{SS}^2 \exp(-\lambda t) \tag{3.44}$$
$$C_{SX}(t) = \rho\sigma_{SS}^2 t \ \mathrm{exprel}[(\lambda-\mu)t] \ \exp(-\lambda t) + \sigma_{SX}^2 \exp(-\mu t) \tag{3.45}$$
$$C_{XS}(t) = \sigma_{SX}^2 \exp(-\lambda t) \tag{3.46}$$
$$C_{XX}(t) = \rho\sigma_{SX}^2 t \ \mathrm{exprel}[(\lambda-\mu)t] \ \exp(-\lambda t) + \sigma_{XX}^2 \exp(-\mu t). \tag{3.47}$$

In the expressions above we used the relative exponential function

$$\mathrm{exprel}(x) = \begin{cases} \frac{\exp(x)-1}{x}, & \text{if } x \neq 0 \\ 1, & \text{if } x = 0 \end{cases} \tag{3.48}$$

and the point (co-)variances

$$\sigma_{SS}^2 = \frac{\kappa}{\lambda} \tag{3.49}$$
$$\sigma_{SX}^2 = \frac{\rho\sigma_{SS}^2}{\lambda+\mu} \tag{3.50}$$
$$\sigma_{XX}^2 = \frac{\rho}{\mu}(\sigma_{SS}^2 + \sigma_{SX}^2). \tag{3.51}$$

Because the process is stationary, the values of the covariance functions for $t < 0$ can be obtained by applying the symmetry relation $C_{AB}(t) = C_{BA}(-t)$.

Now, we can directly compute the mutual information from the covariances. Using Eqs. (3.44) to (3.47) we obtain the matrix elements of the joint covariance matrix $\Sigma$ defined in Eq. (3.43) and then the mutual information is given by the expression

$$I(\mathcal{S},\mathcal{X}) = \frac{1}{2}\ln\left(\frac{|\Sigma_{SS}||\Sigma_{XX}|}{|\Sigma|}\right). \tag{3.52}$$

Note, that for discretized trajectories of length $n$, the matrix $\Sigma$ has dimensions of $2n \times 2n$. Thus, the computation of the trajectory mutual information requires the computation of a $2n \times 2n$ matrix, which can be computationally challenging for long trajectories (large $n$). In Appendix A of this thesis, we discuss some techniques to considerably accelerate the computation of the mutual information.

Using spectral analysis, Tostevin and ten Wolde [195] were able to derive an analytical expression for the Gaussian mutual information rate of this model in the continuous-time limit, given by

$$R(\mathcal{S},\mathcal{X}) = \frac{\lambda}{2}\left[\sqrt{1+\frac{\rho}{\lambda}} - 1\right]. \tag{3.53}$$

The information rate of the discretely sampled process converges to this value as the sampling rate approaches infinity [195]. Notably however, the model corresponding to Eq. (3.53) is a continuum description that assumes Gaussian statistics; indeed, this rate deviates from the mutual information rate of the exact model that is described by the chemical master equation, even in the limit of large copy numbers [119]. This finding is also further discussed in Chapter 5 (Section 5.2.1).

# 4 Application—Bacterial Chemotaxis

*The chemotaxis signaling network of the bacterium Escherichia coli is a sophisticated information processing system, enabling the bacterium to sense nutrient gradients and dynamically adjust its movement. The bacterium's ability to climb chemical gradients is constrained by the mutual information rate between the sensed nutrient concentration and the phosphorylated messenger protein CheYp. A recent study by Mattingly et al. (2021) used a Gaussian approximation to estimate this rate, based on the assumption that in shallow gradients the chemotactic response is approximately linear. However, the nonlinear nature of chemotaxis suggests that Gaussian methods may only approximate the true information rate. In this chapter, we apply an exact technique, Path Weight Sampling (PWS), to precisely quantify information transmission in E. coli chemotaxis and compare the results against the Gaussian approximation. We build a stochastic model based on literature data, which we use to simulate nonlinear chemotactic responses to time-dependent stimuli. Our PWS results for this model yield information rates 4–5 times higher than those obtained experimentally. While this finding can be viewed as surprisingly accurate for an ab initio prediction, the question remains whether the discrepancy is due to the limitations of the Gaussian framework used by Mattingly et al. or due to the assumptions of our stochastic model. Examining the latter question reveals that our initial model underestimates both the magnitude of the response and the biochemical noise. We refined the model by changing two key parameters that describe the receptor array, namely the number of clusters and their size. This leads to information rate estimates that closely align with experimental data, indicating that the number of receptor clusters is much smaller than hitherto believed, while their size is much larger. Finally, our analysis confirms the accuracy of the Gaussian framework for studying chemotaxis in shallow gradients, validating its use by Mattingly et al. a posteriori.*

The contents of this chapter have been published in *Phys. Rev. X* **13**, 041017 (2023) [152].

The chemotaxis system of the bacterium *Escherichia coli* is a complex information processing system. It is responsible for detecting nutrient gradients in the cell's environment and using that information to guide the bacterium's movement. *E. coli* navigates through its environment by performing a biased random walk, successively alternating between so-called *runs*, during which it swims with a nearly constant speed, and *tumbles*, during which it randomly chooses a new direction [15]. By adaptively varying the tumbling probability and thus adjusting the relative duration of runs and tumbles, the bacterium is able to climb a chemical gradient.

Recently, Mattingly et al. [107] found that the ability of the bacterium to climb chemical gradients is fundamentally limited by the information it can acquire via its receptors. However, to calculate the information rate from their experimental data, they relied on a Gaussian approximation which assumes linear Gaussian statistics for the trajectories. The use of the approximation is justified by the argument that in shallow concentration gradients the behavior of the chemotaxis network is approximately linear. Nevertheless, it is well known that chemotaxis generally exhibits a highly nonlinear response, which suggests limitations to the Gaussian framework's accuracy in capturing the full dynamics of the system. Moreover, we have seen in the previous chapter that the Gaussian approximation may fail in surprising ways.

In this chapter, we use Path Weight Sampling (PWS) to exactly quantify information transmission in *E. coli* chemotaxis, and to assess the accuracy of the Gaussian approximation. To study information transmission in *E. coli* chemotaxis using PWS, we first develop a stochastic model of the biochemical chemotaxis network based on the MWC model for receptor cooperativity [117, 8, 175]. This model accurately describes the nonlinear behavior of the chemotaxis network and allows us to simulate the chemotactic response to an arbitrary time-dependent stimulus. Furthermore, we can use PWS to compute the mutual information rate between arbitrary input signals and the response signal generated by the model. To obtain realistic time-dependent input signals, we assume a cell performing a random walk in a static exponential gradient, following Mattingly et al. [107]. By using the same input dynamics as those in Ref. [107], we can rigorously compare our results against the experiments.

One of our principal findings is that the information transmission rate computed by PWS for the model based on the literature data—which we refer to as the "literature-based model"—is approximately 4–5 times higher than the rate measured experimentally by Mattingly et al. [107]. Given that this model relies entirely on available literature data without any fitting to the results of Mattingly et al., an agreement within a factor of 4–5 is perhaps unexpectedly good. Still, the source of the discrepancy remains unclear: does it stem from the inaccuracy of the Gaussian framework used by Mattingly et al. or from the inaccuracy of our literature-based model?

To address this question, we examined the data of Mattingly et al. on the response

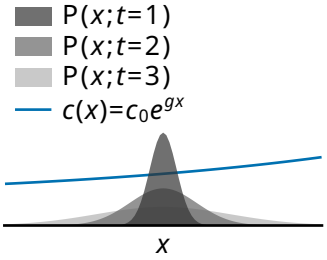



Figure 4.1: In a shallow exponential gradient $c(x)$, the bacterium diffuses nearly freely in the $x$-direction. The variance of the position increases with time, the hallmark of a random walk. The input signal is the concentration $c(t) = c(x(t))$ as experienced by the bacterium at time $t$.

and the noise amplitude. We found that the literature-based model underestimates not only the response strength but also the noise. By fitting the response kernel and the noise correlation function, we developed a "fitted model" in which receptor clusters are larger, enhancing the response amplitude, but significantly fewer in number, leading to much greater noise and thus a lower information transmission rate. Recomputing the information rate with PWS for this model, fitted to the linear response and the noise, yielded close agreement with the information rate measurements of Mattingly et al., suggesting that the receptor array composition differs substantially from previous assumptions. In particular, while the predicted cluster size is about a factor of 2 larger than previous estimates, the number of clusters is about tenfold lower. Lastly, we found that the Gaussian framework is highly accurate in the regime of shallow gradients as studied by Mattingly et al. [107], thus verifying their Gaussian approach *a posteriori*.

In Section 4.1 we describe the dynamics of the input signal. Section 4.2 introduces the stochastic model of the chemotaxis system that we developed based on available literature. In Section 4.3 we describe the Gaussian approximation used by Mattingly et al. [107] to compute the information transmission rate and in Section 4.4 we present the results. We conclude with a discussion of our findings, particularly regarding the number of clusters and their size.

## 4.1 Stochastic Dynamics of the Input Signal for Chemotaxis

The information transmission rate depends not only on the biochemical chemotaxis network, but also on the dynamics of the input signal. It is therefore important that the dynamics of this signal in our model agree with those in the experiments of Mattingly et al. [107]. For these experiments the input signal is the time-dependent ligand concentration $c(t)$ that is experienced by the swimming bacterium.

We consider an *Escherichia coli* bacterium that swims in a static nutrient concentration gradient $c(x)$. Following Mattingly et al. [107], the gradient is exponential:

$c(x) = c_0 e^{gx}$ with steepness $g$. In a shallow gradient, the speed $v_x(t)$ of *E. coli* along the gradient axis can be considered as a stochastic process that fluctuates around the net chemotactic drift velocity. Following Mattingly et al. [107], we assume that in a shallow gradient the bacterial swimming dynamics are, to a good approximation, the same as they are in the absence of a gradient. Their experimental evidence shows that the velocity fluctuations in absence of a gradient are described by an exponentially decaying auto-correlation function:

$$V(t) = \langle v_x(0) v_x(t) \rangle = a_v e^{-\lambda |t|} . \tag{4.1}$$

Therefore, in a shallow gradient, the gradient-climbing speed can be modeled as a zero-mean Ornstein-Uhlenbeck process

$$\frac{dv_x}{dt} = -\lambda v_x + \sigma \xi(t) \tag{4.2}$$

where $\sigma = \sqrt{2 a_v \lambda}$, and $\xi(t)$ is white noise with $\langle \xi(t) \xi(t') \rangle = \delta(t - t')$. The $x$-position of the bacterium is given by the integral of the velocity, i.e., $x(t) = \int_0^t d\tau \, v_x(\tau)$. Thus, when projected onto the gradient axis, the bacterium performs a 1D random walk described by the Langevin equation

$$\frac{d^2 x}{dt^2} = -\lambda \frac{dx}{dt} + \sigma \xi(t) . \tag{4.3}$$

Since the bacterium moves in a static concentration gradient described by $c(x)$, the concentration dynamics that the cell observes are generated directly from its own movement dynamics, see Fig. 4.1. At time $t$ the cell is at position $x(t)$ and thus measures the concentration $c(t) = c(x(t))$. We find the stochastic dynamics of $c$ by differentiating using the chain rule

$$\frac{dc}{dt} = \frac{\partial c}{\partial x} \frac{\partial x}{\partial t} = g c(t) \, v_x(t) . \tag{4.4}$$

The concentration dynamics are thus fully determined by the stochastic dynamics of the cell's swimming velocity $v_x(t)$ in the absence of a gradient and by the shape of the concentration gradient $c(x)$. The resulting stochastic dynamics are visualized in Fig. 4.2.

In the PWS simulations we use $c(t)$ directly as the input to our system. Yet, for the Gaussian approximation, to which we will compare the PWS result, we need to use a different input signal because the chemotaxis system does not respond linearly to $c(t)$. Instead, Mattingly et al. [107] show that the chemotaxis system responds approximately linear to an input $s(t)$ defined by

$$s(t) = \frac{d}{dt} \ln c(t) = g v_x(t) . \tag{4.5}$$

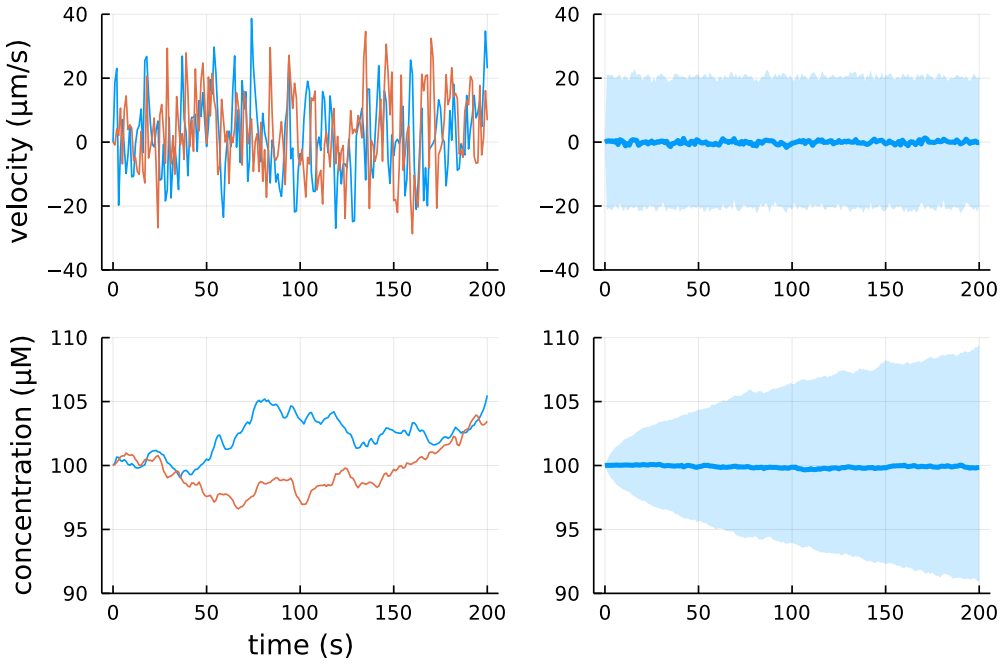



Figure 4.2: Input dynamics. *Left column:* two example time traces of the up-gradient velocity $v_x(t)$ and the observed concentration $c(t)$ obtained by integrating Eq. (4.4). *Right column:* averages of velocity and concentration traces obtained from 1000 simulated trajectories. The solid lines show the mean as a function of time and boundaries of the shaded regions indicate the 5% and 95% quantiles.

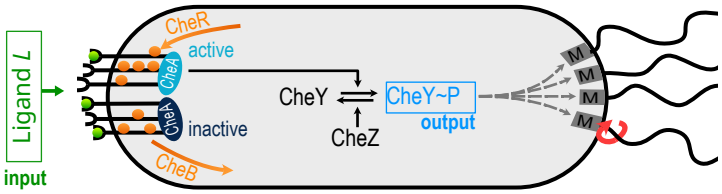



Figure 4.3: Cartoon of the chemotaxis network of *E. coli*. Receptors form clusters with an associated CheA kinase. A cluster can either be active or inactive, depending on the number of bound ligands (green dots) and methylated sites (orange dots). Active CheA can phosphorylate CheY; phosphorylated CheY controls the rotation direction of the flagellar motors and thereby the movement of the bacterium.

The correlation function of $s(t)$ is given by

$$C_{ss}(t) = \langle s(\tau)s(t+\tau)\rangle = g^2 V(t)\,. \tag{4.6}$$

The power spectral density of this signal is given by the Fourier transform of its correlation function:

$$P_{ss}(\omega) = g^2 V(\omega) = g^2 \frac{2a_v\lambda}{\omega^2+\lambda^2}\,. \tag{4.7}$$

We use this same input below in Section 4.3 to compute the Gaussian approximation of the mutual information rate. As discussed in more detail in the main text, we note that the mutual information between the output and the input trajectory $c(t)$, as measured in the PWS simulations, is identical to that between the output and the input trajectory $s(t)$, as computed in the Gaussian model because of the deterministic and monotonic mapping between $c(t)$ and $s(t)$.

## 4.2 Stochastic Chemotaxis Model

We apply PWS to a stochastic model of the chemotaxis network that describes individual reactions via a master equation. In this model, the receptors are grouped in clusters. Each receptor can switch between an active and an inactive conformational state, but, in the spirit of the Monod-Wyman-Changeux model [117], the energetic cost of having two different conformations in the same cluster is prohibitively large. We can then speak of each cluster as either being active or inactive. Each receptor in a cluster can bind ligand and be (de)methylated, which, together, control the probability that the cluster is active. In the simulations, receptor (de)methylation is modeled explicitly, because the (de)methylation reactions

are slow. In contrast, the timescale of receptor-ligand (un)binding is much faster than the other timescales in the system, i.e., those of the input dynamics, CheY (de)phosphorylation, and receptor (de)methylation. The receptor-ligand binding dynamics can therefore be integrated out without affecting information transmission, in order to avoid wasting CPU time. In addition, the receptor clusters can phosphorylate CheY, while phosphorylated CheY is dephosphorylated at a constant rate. The dynamics of the kinase CheA and the phosphatase CheZ which drive (de)phosphorylation are not modeled explicitly. Figure 4.3 shows a depiction of the bacterial chemotaxis network.

Table 4.1 shows the parameter values of our chemotaxis model, which are all based on values reported in the literature. For what follows below, the key parameters are the number of receptors per cluster, which is taken to be $N = 6$ based on Refs. [165, 82], while the number of clusters is $N_c = N_r/N = 400$, where $10^3 < N_r < 10^4$ is an estimate for the total number of receptors based on Ref. [96]. These are the numbers of the "literature-based model". The key parameters, the number of clusters and their size, change in the "fitted model", which is based on fitting the response kernel and noise correlation function to the data of Mattingly et al. [107].

### 4.2.1 MWC Model

In our model, each cluster consists of $N$ receptors. Shimizu et al. [165] report a typical value for the cluster size of $N = 6$. Detailed balance requires that the ligand binding affinity depends on whether a cluster is in the active or inactive state. Consequently, we have a dissociation constant $K_a$ for a ligand bound to an active receptor and another dissociation constant $K_i$ for a ligand bound to an inactive receptor. For chemotaxis, $K_a \gg K_i$, i.e. the ligand binding affinity is higher for the inactive state.

Additionally, each receptor monomer has $M$ methylation sites that can affect its conformation and therefore the kinase activity. The aspartate receptor Tar has $M = 4$ methylation sites [165]. We model the receptors' methylation dynamics such that CheB only demethylates active receptors while CheR only methylates inactive receptors, following previous models for chemotaxis [8, 121]. This approach represents arguably the simplest way to model methylation with exact receptor adaptation.

In an environment with ligand concentration $c$, the probability of a receptor cluster with $m$ methylated sites to be active, $p_a(c, m)$, is determined by the free-energy difference between the active and inactive receptor states

$$p_a(c, m) = \frac{1}{1 + e^{-f(c,m)}} \tag{4.8}$$

| parameter | value | | description |
|---|---|---|---|
| $a_v$ | 157.1 | $\mu m^2\,s^{-2}$ | variance of up-gradient velocity [107] |
| $\lambda$ | 0.86 | $s^{-1}$ | velocity correlation decay constant [107] |
| $c_0$ | 100 | µM | mean ligand concentration [107] |
| $N$ | 6 | | number of receptor units per cluster [165] |
| $N_c$ | 400 | | number of receptor clusters [96] |
| $M$ | 4 | | number of methylation sites per receptor [165] |
| $N_Y$ | 10 000 | | total copy number of CheY proteins (phosphorylated and unphosphorylated) [96] |
| $K_a$ | 2900 | µM | ligand dissociation constant of active receptors [82] |
| $K_i$ | 18 | µM | ligand dissociation constant of inactive receptors [82] |
| $k_R$ | 0.1 | $s^{-1}$ | methylation rate [165, 107] |
| $k_B$ | 0.2 | $s^{-1}$ | demethylation rate [165, 107] |
| $k_A$ | 0.015 | $s^{-1}$ | phosphorylation rate [173, 174] |
| $k_Z$ | 10.0 | $s^{-1}$ | dephosphorylation rate [173, 174] |
| $\phi_Y$ | 0.17 | | steady-state fraction of phosphorylated CheY [173] |
| $m_0/N$ | 0.5 | | receptor methylation level without ligands [165] |
| $\delta f_m$ | −2.0 | $k_B T$ | free energy change of active conformation from attachment of one methyl group [165] |

Table 4.1: The parameters required for the chemotaxis model, based on literature values. These are the parameters used in the so-called literature-based model. In the fitted model (see Section 4.4) the same parameter values are chosen, except for $N = 15$ and $N_c = 9$, which were obtained by fitting to the data of Mattingly et al. [107]; we note that changing $N$ and $N_c$ also requires updating $k_A$ to keep the fraction $\phi_Y$ of phosphorylated CheY constant.

where

$$f(c,m) = N \ln\left(\frac{1+c/K_i}{1+c/K_a}\right) + \delta f_m(m-m_0)\,. \tag{4.9}$$

Here, the number of methylated sites of a cluster (not receptor) is denoted by $m$, ranging from 0 to $NM$. The parameters are again taken from Shimizu et al. [165]. Their experimental results indicate that $\delta f_m = -2k_\mathrm{B}T$, $m_0 = -N/2$. Kamino et al. [82] report ligand dissociation constants of $K_a = 2900\,\mu\mathrm{M}$ for active receptors and $K_i = 18\,\mu\mathrm{M}$ for inactive Tar receptors (for MeASP). Note that in the equations we assume units such that $k_\mathrm{B}T = 1$.

The dynamics of methylation in our model are described by the following mean-field equation

$$\frac{dm}{dt} = (1 - p_a(c,m))k_R - p_a(c,m)k_B\,. \tag{4.10}$$

The system reaches a steady state for the adapted activity $p_a(c,m) = a_0$ where

$$a_0 = \frac{k_R}{k_R + k_B}\,. \tag{4.11}$$

The steady-state methylation $m^\star$ can be obtained from Eqs. (4.8) and (4.9) by solving $p_a(c,m^\star) = a_0$:

$$m^\star = m_0 + \frac{N \ln\left(\frac{1+c/K_i}{1+c/K_a}\right) + \ln\left(\frac{1-a_0}{a_0}\right)}{-\delta f_m}\,. \tag{4.12}$$

To characterize the methylation timescale, we linearize the dynamics of $m(t)$ around the steady state (at constant ligand concentration $c(t) = c_0$). To first order, we can write

$$\frac{dm}{dt} = -\frac{m(t) - m^\star}{\tau_m}\,. \tag{4.13}$$

where $\tau_m$ is the characteristic timescale of the methylation dynamics. We find $\tau_m$ by expanding $p_a$ [Eq. (4.8)] around $m = m^\star$:

$$\begin{aligned} p_a(c,m) &= p_a(c,m^\star) + \left.\frac{\partial p_a}{\partial m}\right|_{m^\star}(m-m^\star) + \mathcal{O}(m^2) \\ &= a_0[1 - \delta f_m(1-a_0)(m-m^\star)] + \mathcal{O}(m^2)\,, \end{aligned} \tag{4.14}$$

and then plugging this first-order expansion into Eq. (4.10) to get

$$\frac{dm}{dt} = \frac{\delta f_m(m-m^\star)}{k_R^{-1} + k_B^{-1}}\,. \tag{4.15}$$

By comparing with Eq. (4.13) we find that for small perturbations, the timescale for methylation to approach steady state is given by

$$\tau_m = \frac{k_R^{-1} + k_B^{-1}}{-\delta f_m} \,. \tag{4.16}$$

Thus, the parameters $k_R$ and $k_R$ determine two important characteristics of the methylation system: the adapted activity $a_0$ and the methylation time scale $\tau_m$. Shimizu et al. [165] report an adapted activity of $a_0 = 1/3$ and based on experimental data [107, 165] we assume a methylation time scale of $\tau_m = 10\,\mathrm{s}$. Our parameter choice, which is consistent with both of these observations, is $k_R = 0.075\,\mathrm{s}^{-1}$ and $k_B = 0.15\,\mathrm{s}^{-1}$.

CheY is phosphorylated by CheA, the receptor-associated kinase. The kinase activity is directly linked to the activity of a receptor cluster. Therefore, we assume that CheY is phosphorylated by active receptor clusters. Dephosphorylation of CheY-p is catalyzed by the phosphatase CheZ, which we assume to be present at a constant concentration. The CheZ-catalyzed dephosphorylation rate was reported to be $2.2\,\mathrm{s}^{-1}$ for attractant response and $22\,\mathrm{s}^{-1}$ for repellent response [174]. Based on this data, we use the approximate dephosphorylation rate $k_Z = 10\,\mathrm{s}^{-1}$ in our model. In the fully adapted state the fraction of active receptors is $a_0$ and therefore the mean fraction of phosphorylated CheY, $\phi_Y = [\mathrm{CheYp}]/([\mathrm{CheY}] + [\mathrm{CheYp}])$, is given by

$$\phi_Y = \frac{a_0 N_c k_A}{k_Z + a_0 N_c k_A} \,. \tag{4.17}$$

In the fully adapted state the phosphorylated fraction was found to be $\phi_Y \approx 0.16$ [173]. Hence, we infer a phosphorylation rate of $k_A = k_Z \phi_Y / (a_0 N_c (1 - \phi_Y)) = 0.015\,\mathrm{s}^{-1}$ for the literature-based model. Accordingly, for the "fitted model", based on fitting $K(t)$ and $N(t)$ to those measured by Mattingly et al. [107], we use a larger phosphorylation rate due to the smaller number of clusters $N_c$.

### 4.2.2 Reaction Kinetics

Since the timescale of conformational switching of active and inactive receptors and ligand binding is much faster [134] than the timescale of phosphorylation or methylation, we don't explicitly model ligand (un)binding and conformational switching. Each cluster is characterized by its methylation state $m$. This ranges from 0 to the total number of methylation sites, which equals the number of sites per receptor $M$ times the number of receptors per cluster $N$. In our Gillespie simulation, each possible state of a cluster is its own species, i.e., we have species $\mathrm{C}_m$ for $m = 0, \ldots, NM$. Overall, our chemotaxis model consists of four types of reactions that describe (a)

the methylation of a receptor $C_m \rightarrow C_{m+1}$, (b) the demethylation of a receptor $C_m \rightarrow C_{m-1}$, (c) the phosphorylation of CheY $C_m + Y \rightarrow C_m + Y_p$, and (d) the single dephosphorylation reaction $Y_p \rightarrow Y$. Thus, due to the combinatorial explosion of receptor states, the system has a total number of $3NM + 2$ elementary reactions (which amounts to 75 reactions in the literature-based model and 182 reactions in the fitted model).

The ligand-concentration dependent methylation rate for $C_m \rightarrow C_{m+1}$ is given by

$$k_{m+}(c, m) = (1 - p_a(c, m))k_R \,. \tag{4.18}$$

The term $1 - p_a(c, m)$ is needed because only inactive receptors can be methylated. The demethylation rate for $C_m \rightarrow C_{m-1}$ is given by

$$k_{m-}(c, m) = p_a(c, m)k_B \tag{4.19}$$

where only active receptors can be demethylated. These zero-order dynamics of (de)methylation of receptors lead to the adaptive behavior of the chemotaxis system as described above.

Only active receptors can phosphorylate the CheY protein using the receptor-associated kinase CheA. We therefore model phosphorylation as a reaction $C_m + Y \rightarrow C_m + Y_p$ with rate

$$k_{Y \rightarrow Y_p}(c, m) = p_a(c, m)k_A \tag{4.20}$$

where $k_A$ is a constant that represents the phosphorylation rate of an active cluster. The dephosphorylation $Y_p \rightarrow Y$ is carried out by the phosphatase CheZ at a constant rate $k_Z = 10\,\mathrm{s}^{-1}$.

## 4.3 Mutual Information Rate for the Chemotaxis System in the Gaussian Approximation

To test the validity of the Gaussian approach used by Mattingly et al. [107], we also compared the exact PWS results for our discrete, stochastic model to the prediction of the Gaussian approximation for this same model. In continuous time, the information transmission rate $R(\mathcal{S}, \mathcal{X})$ of a Gaussian system in steady state can be computed exactly from the power spectral density functions of the system [194]:

$$R(\mathcal{S}, \mathcal{X}) = -\frac{1}{4\pi} \int_{-\infty}^{\infty} d\omega \, \ln\left[1 - \frac{|P_{sx}(\omega)|^2}{P_{ss}(\omega)P_{xx}(\omega)}\right] . \tag{4.21}$$

Here, the power spectral density $P_{\alpha\beta}(\omega)$ is defined as

$$P_{\alpha\beta}(\omega) = \int_{-\infty}^{\infty} dt \, e^{-i\omega t} C_{\alpha\beta}(t) \tag{4.22}$$

where $C_{\alpha\beta}(t_i - t_j) = \langle \alpha(t_i)\beta(t_j)\rangle$ denote the stationary (cross-)correlation functions of the system.

Thus, we need to obtain the (cross-)correlation functions to compute the information rate in the Gaussian framework. In their experiments with *E. coli* bacteria, Mattingly et al. [107] don't obtain these correlation functions directly, however. Instead, they obtain three kernels, $V(t)$, $K(t)$ and $N(t)$, from which the correlation functions can be inferred. We follow this approach for calculating the Gaussian information transmission rate.

### 4.3.1 Computing the Gaussian information rate using linear response kernels *V(t), K(t),* and *N(t)*

$V(t)$ denotes the autocorrelation function of the swimming velocity of bacteria, i.e., $V(t) = \langle v_x(\tau)v_x(\tau + t)\rangle$. As explained in Section 4.1, the swimming dynamics of the bacteria determine the statistics of the input signal $s(t) = \frac{d}{dt}\ln c(t)$, where $c(t)$ is the ligand concentration as experienced by the bacterium and $g$ is the gradient steepness. The input signal correlation function, denoted by $C_{ss}(t)$, can then be expressed as

$$C_{ss}(t) = g^2 V(t)\,. \tag{4.23}$$

The response kernel, denoted by $K(t)$, represents the time evolution of the average activity of the receptors in response to an instantaneous step change in the input concentration. More precisely, $K(t)$ is defined as

$$K(t) = \theta(t)\langle a(t) - a_0\rangle \ln\frac{c_s}{c_0} \tag{4.24}$$

where we assume the input concentration jumps instantaneously from $c_0$ to $c_s$ at time $t = 0$. $\theta(t)$ is the Heaviside step function. Note that because the signal $s(t)$ is defined as the time-derivative of the concentration $c(t)$, a step-change in $c(t)$ corresponds to a delta impulse in $s(t)$. Thus, $K(t)$ describes the deterministic dynamics of the system after being subjected to a delta stimulus $s(t) = \delta(t)$, making $K(t)$ the Green's function of the system. The activity $a(t)$ resulting from arbitrary time-dependent signal $s(t)$ can be written as a convolution of $K(t)$ with $s(t)$

$$a(t) = a_0 + \int_{-\infty}^{t} dt'\, K(t - t')s(t') + \eta_a(t) \tag{4.25}$$

where $\eta_a(t)$ is the receptor activity noise. We define the response $x(t) = a(t) - a_0$. Assuming the input statistics are stationary and described by the correlation

function $C_{ss}(t)$, it is easy to show that the cross-correlation between $s(t)$ and $x(t)$ is given by

$$C_{sx}(t) = \langle s(\tau)x(\tau + t)\rangle = \int_{-\infty}^{t} dt'\, K(t - t')C_{ss}(t')\,. \tag{4.26}$$

In other words, the cross-correlation between $s(t)$ and $x(t)$ is given by the convolution of the response kernel with the input correlation function.

The noise kernel $N(t)$ describes the autocorrelation of the activity fluctuations in the absence of an input stimulus. In particular, $N(t) = \langle \eta_a(\tau)\eta_a(\tau + t)\rangle$.

We now rewrite Eq. (4.21) for the mutual information rate in terms of the three kernels described above. We express the power spectra $P_{\alpha\beta}(\omega)$ in terms of the Fourier-transformed kernels $V(\omega)$, $K(\omega)$, and $N(\omega)$. In Section 4.1 we already showed that $P_{ss}(\omega) = g^2V(\omega)$. The cross power spectrum is given by $P_{sx}(\omega) = K(\omega)P_{ss}(\omega)$ which follows from Eq. (4.26). Finally, from Ref. [194] we use the identity $P_{xx}(\omega) = P_{ss}(\omega)|K(\omega)|^2 + N(\omega)$ to express the output power spectrum. We insert these expressions into Eq. (4.21) which yields

$$R(\mathcal{S}, \mathcal{X}) = \frac{1}{4\pi}\int_{-\infty}^{\infty} d\omega\, \ln\left(1 + \frac{g^2V(\omega)|K(\omega)|^2}{N(\omega)}\right). \tag{4.27}$$

Then, for shallow gradients, we can make a Taylor approximation in $g$ to obtain

$$R(\mathcal{S}, \mathcal{X}) = \frac{g^2}{4\pi}\int_{-\infty}^{\infty} d\omega\, \frac{V(\omega)|K(\omega)|^2}{N(\omega)} + \mathcal{O}(g^4)\,. \tag{4.28}$$

This result shows that the information rate in shallow gradients is proportional to $g^2$ and the proportionality constant is determined by the measured kernels. Mattingly et al. [107] obtain the relevant kernels $V(\omega)$, $K(\omega)$, and $N(\omega)$ from experiments by fitting phenomenological models to their single-cell data. We obtain the kernels from the simulation outputs of our stochastic chemotaxis model.

### 4.3.2 Estimating the kernels from simulations

#### Response Kernel *K(t)*

To compute the response kernel from simulations of our model, we study how the system responds to a sudden increase in ligand concentration. First, we allow the system to reach steady state by adapting it to an initial ligand concentration of $c_0 = 100\,\mu\text{M}$ for $t_0 = 50\,\text{s}$. At time $t = t_0$, we instantaneously increase the concentration by 10% to $c_s = c_0 + 0.1c_0$. We then record the system's response over the next 200 s, sampling at intervals of 0.01 s.

Rather than directly obtaining the receptor activity from the simulations, we follow the experimental approach of inferring the receptor activity from the phosphorylated CheY levels. Specifically, we record the fraction $f(t) = [\mathrm{Y_p}]/[\mathrm{Y}]$ between phosphorylated and unphosphorylated CheY. Since the copy number of CheY is relatively large, this fraction serves as a good proxy for the activity $a(t)$. We relate the $f(t)$ to the activity $a(t)$ via the expression

$$a(t) = \frac{k_Z}{k_A N_c} f(t) \tag{4.29}$$

where $k_A$ and $k_Z$ are the phosphorylation and dephosphorylation rate, respectively, and $N_c$ is the number of receptor clusters.

Finally, we estimate the response Kernel $K(t)$ by averaging the changes in measured activity over $10^5$ simulated trajectories

$$K(t) = \ln\left(\frac{c_s}{c_0}\right)\langle a(t - t_0) - a(t_0)\rangle\,. \tag{4.30}$$

### Output Noise Kernel *N(t)*

We can similarly obtain the noise statistics of the output from simulations of our chemotaxis model. In this case, we stochastically evolve the chemotaxis model at constant ligand concentration $c_0 = 100\,\mu\mathrm{M}$ for a very long time of $1 \times 10^4$ s. The result is a time trace of the activity $a(t)$, which we again obtain from the fraction $f(t)$ using Eq. (4.29). We discretize this time trace at a resolution of 0.01 s. This results in a time series $\boldsymbol{a} = (a_1, \ldots, a_N)^T$ where $a_i = a(t_i)$. To estimate the correlations in the time series we subtract the overall average activity from each data point and thus obtain the data vector $\boldsymbol{x}$ where $x_i = a_i - \sum_{j=1}^{N} a_j/N$. From $\boldsymbol{x}$ we estimate the auto-correlation function $C_{xx}(t) = \langle x(\tau)x(\tau + t)\rangle$ of the activity. To obtain precise results we average the correlation function for $10^5$ trajectories.

### Obtaining the Fourier Kernels using the FFT

To compute the Gaussian information rate, we need the frequency-space representations of the kernels $V(t)$, $K(t)$, and $N(t)$. We already derived the analytical form of $V(\omega)$ in Section 4.1. We obtain $K(\omega)$ and $N(\omega)$ numerically via a discrete Fourier transform of the corresponding time-domain kernel.

As explained above, we compute time-discretized kernels $K_i = K(t_i)$ and $N_i = N(t_i)$ from time traces obtained via stochastic simulations of our model. We sample these functions at the instants $t_0, \ldots, t_{N-1}$, the sampling frequency being $f_s =$

100 Hz. Then, we use the discrete Fourier transform (DFT) to obtain approximations for $K(\omega)$ and $N(\omega)$ as follows. The DFT coefficients $\tilde{K}_k$ are given by

$$\tilde{K}_k = \sum_{n=0}^{N-1} K_n e^{-i2\pi nk/N} \tag{4.31}$$

where $k = 0, 1, \ldots, N-1$. These DFT coefficients can be computed efficiently using the Fast Fourier Transform (FFT) algorithm. The DFT provides point estimates for the Fourier-domain kernel $K(\omega)$ at discrete frequencies

$$\omega_k = \frac{2\pi f_s k}{N}, \quad k = 0, 1, ..., N-1\,, \tag{4.32}$$

i.e., $K(\omega_k) \approx \tilde{K}_k$. This approximation introduces some level of error, known as spectral leakage, due to the finite duration and sampling of the signal. This error can be reduced by multiplying the time-domain kernel with a window function. Thus, before computing the DFT, we multiply the kernel with a Hanning window, which is a smooth function that tapers at the edges of the kernel, reducing the effect of discontinuities at the beginning and end of the time series. The Hanning window is defined as:

$$h_n = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi n}{N-1}\right)\right], \quad n = 0, 1, ..., N-1\,. \tag{4.33}$$

The windowed kernel $k_n$ is obtained by multiplying the time-domain kernel $K_n$ with the Hanning window $h_n$:

$$k_n = K_n h_n, \quad n = 0, 1, ..., N-1 \tag{4.34}$$

Using the FFT algorithm we then compute the DFT coefficients $\tilde{k}_k$ of the windowed kernel.

The procedure described above to obtain the DFT coefficients $\tilde{k}_k$ from $K(t)$ is also applied to $N(t)$ to obtain the coefficients $\tilde{n}_k$.

We can then evaluate the information rate using Eq. (4.28) by discretizing the integral $\int d\omega\, F(\omega) \to \sum_k \Delta\omega\, F(\omega_k)$ with $\Delta\omega = 2\pi f_s/N$. More precisely, we compute the Gaussian information rate as

$$R(\mathcal{S}, \mathcal{X}) = \frac{g^2}{4\pi} \sum_{k=0}^{N-1} \Delta\omega\, \frac{V(\omega_k)|\tilde{k}_k|^2}{\tilde{n}_k}\,. \tag{4.35}$$

## 4.4 Results

We first asked whether our chemotaxis model based on the current literature can reproduce the information transmission rate as recently measured by Mattingly et al. [107]. In what follows, we call this model the "literature-based" model.

### 4.4.1 Discrepancy between experimental results and literature-based model

In our model, the output is the concentration of phosphorylated CheY, while in the experiments of Mattingly et al. [107] it is the average activity of the receptor clusters as obtained via FRET measurements. We argue that this difference does not significantly affect the obtained information rates, and thus, that it is valid to compare our results to the experiments. In particular, since the copy number of CheY is much larger than the number of receptor clusters, the fluctuations in CheY are dominated by the extrinsic fluctuations coming from the receptor activity noise rather than from the intrinsic fluctuations associated with CheY (de)phosphorylation. To a good approximation, the copy number of phosphorylated CheY, $\mathrm{Y}p(t)$, is thus a deterministic function of the average receptor activity $a(t)$. Mathematically, the mutual information $I(X;Y)$ between two stochastic variables $X$ and $Y$ is the same as the mutual information $I(f(X);g(Y))$ for deterministic and monotonic functions $f$ and $g$. It follows that the mutual information between $c(t)$ and $\mathrm{Y}p(t)$, is nearly the same as that between $c(t)$ and the receptor activity $a(t)$. It is therefore meaningful to compare the information transmission rates as predicted by our PWS simulations to those measured by Mattingly et al. [107].

We use RR-PWS to compute the mutual information for the literature-based model. Specifically, we measure the mutual information $I(\mathbf{C}, \mathbf{Y_p}; T)$ between the input trajectory of the ligand concentration $c(t)$ and the output trajectory of phosphorylated CheY, $y_p(t)$, and where each trajectory is of duration $T$. With RR-PWS it is possible to compute $I(\mathbf{C}, \mathbf{Y_p}; \tau)$ for all $\tau \leq T$ within a single PWS simulation of duration $T$ by saving intermediate results after each sampled segment, see Section 3.2. The receptor states are considered hidden internal states, and we use the technique described in Section 2.2 to integrate them out.

Figure 4.4a shows the PWS estimate of the information transmission rate for cells swimming in gradients of different steepnesses $g$. The information transmission rate is obtained from the PWS estimate of the trajectory mutual information $I(\mathbf{C}, \mathbf{Y_p}; T)$, different trajectory durations $T$. As seen in Fig. 4.4b, for short trajectories the mutual information increases non-linearly with trajectory duration $T$, but in the long-duration limit the slope becomes constant. This asymptotic rate of increase of the mutual information with $T$ is the desired information transmission rate $R(\mathbf{C}, \mathbf{Y_p})$. The precise definition is given by

$$R(\mathbf{C}, \mathbf{Y_p}) = \lim_{T\to\infty} \frac{I(\mathbf{C}, \mathbf{Y_p}; T)}{T}\,. \tag{4.36}$$

We then compared our results for the information transmission rate of the literature-based model to those of Mattingly et al. [107]. Figure 4.5c shows that the model pre-

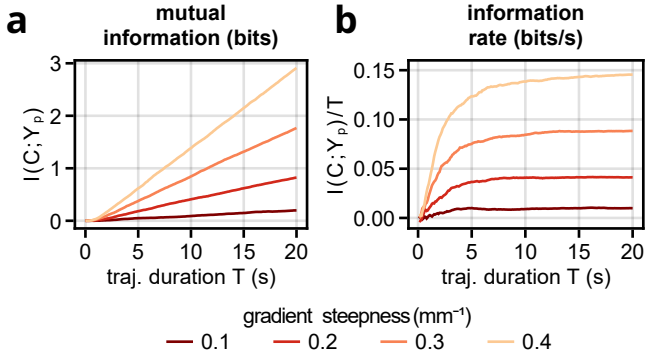



Figure 4.4: PWS simulations for the trajectory mutual information of chemotaxis in the shallow gradient regime. **a** Mutual information $I(\mathbf{C}_T, \mathbf{Y}_T)$ between input trajectories $c(t)$ and output trajectories $y_p(t)$ as a function of trajectory duration $T$. In each RR-PWS simulation, $N = 7200$ Monte Carlo samples were used ($M = 256$ for the particle filter). **b** The information transmission rate is defined as $I(\mathbf{C}_T, \mathbf{Y}_T)/T$ in the limit $T \to \infty$.

dictions differ from the experiments by a factor of $\approx 4$. Despite this discrepancy, we believe that the agreement between experiment and theory is, in fact, remarkable, because these predictions were made *ab initio*: the model was developed based on the existing literature and we did not fit our model to the data of Mattingly et al.

Yet, the question about the origin of the discrepancy remains. The difference between their measurements and our predictions could be attributed either to the inaccuracy of our model or to the approximation that Mattingly et al. had to employ to compute the information transmission rate from experimental data. Concerning the latter hypothesis, due to the curse of dimensionality and experimental constraints, Mattingly et al. could not directly obtain the information transmission rate from measured time traces of the input and output of the system. Instead, they measured three different kernels that describe the system in the linear regime. Specifically, they obtained the response $K(t)$ of the kinase activity to a step-change in input signal, the autocorrelation function of the input signal $V(t)$, and the autocorrelation $N(t)$ of the kinase activity in a constant background concentration. Then they used a Gaussian model to compute the information transmission rate from these measured functions $K(t)$, $V(t)$, and $N(t)$ [194, 107] (see also Section 4.3). This Gaussian model is based on a linear noise assumption and cannot perfectly capture the true non-linear dynamics of the biochemical network. This could be the cause for the observed discrepancies in the information rate. We have indeed already seen in Chapter 3 that there can be substantial differences between exact computations

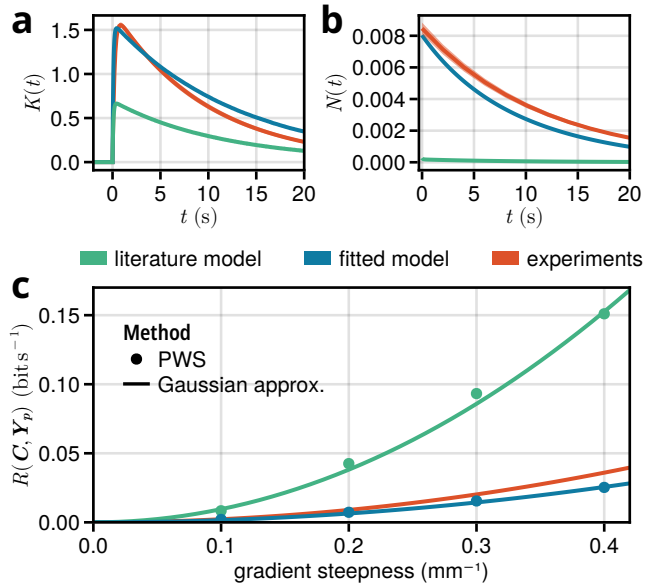



Figure 4.5: Comparison of theoretical models with experimental data for bacterial chemotaxis system. Panels **a** and **b** show the response and noise kernels, respectively, for the model based on literature parameters (green), parameters fitted to experiments (blue), and experiments from Mattingly et al. [107] (orange). In panel **c**, the information transmission rate is shown for each model as a function of gradient steepness, with results from the Gaussian approximation shown alongside exact PWS calculations. The fitted model closely matches the experiments, while the literature-based model over-estimates information transmission rate by a factor of $\approx 4$ despite having a lower response amplitude (panel **a**). This is because the literature-based model has a large number of independents receptor clusters $N_c$, resulting in much lower noise in the output (panel **b**). In all cases, the Gaussian approximation matches the exact PWS results, providing support for the accuracy of the measurements of Mattingly et al. [107].

and the Gaussian approximation for the trajectory mutual information.

### 4.4.2 Revising the literature-based model

To uncover the reason for the discrepancy we first tested whether our literature-based model reproduces the experimentally measured kernels. If the kernels do not match, then, clearly, the discrepancy in the information rate may be caused by the difference between our model and the experimental system, as opposed to the inaccuracy of the Gaussian framework. Our input correlation function, $V(t)$, is, by construction, the same as that of Mattingly et al. [107]. The simulation protocol we used for measuring the other kernels was directly modeled after the experimental protocol [107].

We find that the response kernel $K(t)$ and the autocorrelation function of the noise $N(t)$ of our system are different. Figure 4.5a, b shows that our model reproduces the timescales of $N(t)$ and $K(t)$ as measured experimentally. This is perhaps not surprising, because the decay of both $N(t)$ and $K(t)$ is set by the (de)methylation rate, which has been well-characterized experimentally. Yet, the figure also shows that our model significantly underestimates the amplitudes of both $N(t)$ and $K(t)$.

This raises the question of whether other parameter values would allow our model to better reproduce the measured kernels $K(t)$ and $N(t)$, and, secondly, whether this would resolve the discrepancy in information rate between our simulations and the experiments.

The amplitude $\sigma_N^2$ of the output noise correlation function $N(t)$ is bounded by the number of receptor clusters $N_c$. In particular, the variance of the receptor activity is $\sigma_N^2 = \sigma_a^2/N_c \leq 1/4N_c$, where $\sigma_a^2 \leq 1/4$ is the variance of the activity of a single receptor cluster. Comparing this bound to the measured receptor noise strength $\sigma_N^2$ reveals that $N_c$ needs to be much smaller than our original model assumes: the number of clusters needs to be as small as $N_c \lesssim 10$. Indeed, Fig. 4.5b shows that with $N_c = 9$, our model quantitatively fits the correlation function $N(t)$ of the receptor activity in a constant background concentration, as measured experimentally [107].

The amplitude of $K(t)$, i.e. the gain, depends on the ratio $K_D^A/K_D^I$ of the dissociation constants of the receptor for ligand binding in its active or inactive state, respectively, as well as on the number of receptors per cluster, $N$. Both dissociation constants have been well characterized experimentally [95, 165], but the number of receptors per cluster has only been inferred indirectly from experiments [82, 165]. The higher gain as measured experimentally by Mattingly et al. [107] indicates that $N$ is larger than assumed in our model: with $N = 15$ our model can quantitatively fit $K(t)$ (Fig. 4.5a).

We thus find that by reducing the number of clusters from $N_c = 400$ to $N_c = 9$ while simultaneously increasing their size from $N = 6$ to $N = 15$, our model is able

**PWS Estimate for the Fitted Chemotaxis Model**

In the main text, we described that a chemotaxis model with $N_c = 9$ receptor clusters, each containing $N = 15$ receptors, matches the experimental kernels of Mattingly et al. [107]. We then computed the information rate for this model using both the exact PWS method and a Gaussian approximation. How the rate in the Gaussian model is computed is described in Section 4.3. Here, we describe briefly how we compute the exact rate using PWS.
While in principle the rate could be computed directly via PWS for the model with $N_c = 9$ and $N = 15$, the receptor activity noise was so large that obtaining this estimate directly in a single PWS simulation proved to be inefficient. Instead, we computed the rate via an extrapolation procedure. In particular, we computed the rate for a series of models with $N = 15$, yet with $N_c$ going down from 400 to 50. The rate for the model of interest, with $N = 15$ and $N_c = 9$, was then obtained by fitting this data to a simple polynomial and then extrapolating to $N_c = 9$.
In Fig. 4.7 we show the information rate for a range of values of $N_c$, and for different gradient steepnesses $g$. We see that the information rate increases nonlinearly with the number $N_c$ of independent clusters. Based on the assumption that the information rate is zero in the limit $N_c \to 0$, we fit a quadratic function $R(N_c) = aN_c - bN_c^2$ with positive coefficients $a, b$ to the data. We provide the fit coefficients for different gradient steepnesses $g$ in Table 4.2. From these fits we can obtain the extrapolated information rates for $N_c = 9$ that are shown in the main text.

to quantitatively fit both $N(t)$ and $K(t)$ [107], see Fig. 4.5 and Fig. 4.6 for their Fourier representations. This suggests that the number of independent receptor clusters is smaller than hitherto believed, while their size is larger.

### 4.4.3 Comparing the chemotaxis information rate of the models against experiments

How accurately can our revised model reproduce the measured information rate, and how accurate is the Gaussian framework for the experimental system in the regime studied by Mattingly et al. [107]? In the revised model, called the "fitted model", with $N_c = 9$ and $N = 15$, all key quantities for computing the information transmission rate within the Gaussian framework, $V(t)$, $N(t)$ and $K(t)$, are nearly identical to the experiments of Mattingly et al. [107], see Fig. 4.5. Within the Gaussian framework (see Section 4.3), the information transmission rate in our model is

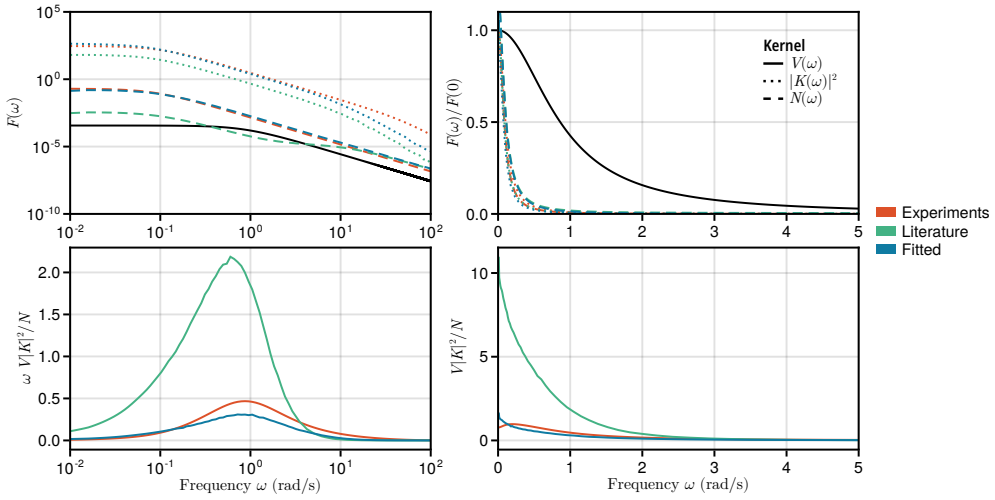



Figure 4.6: Fourier representation of the kernels for computing the information transmission rate in the Gaussian approximation, the velocity power spectrum $V(\omega)$ $[(\mathrm{mm\,s^{-1}})^2]$, the squared frequency response $|K(\omega)|^2$, and the noise power spectrum $N(\omega)$. The top-left panel shows the individual Fourier kernels as a function of frequency $\omega$ for the different models. On the top-right the normalized kernels are shown with linear axis scales. In the bottom panels the integrand for computing the mutual information rate in the Gaussian approximation is shown. In the bottom right, the area under the curves represents the proportionality between the squared gradient steepness $g^2$ and the information rate (units $\mathrm{bit\,s^{-1}\,mm^{-2}}$). In the bottom left plot, the integrand is multiplied by $\omega$, so that with log scaling of the axes the area under the curve is equal to the integral.

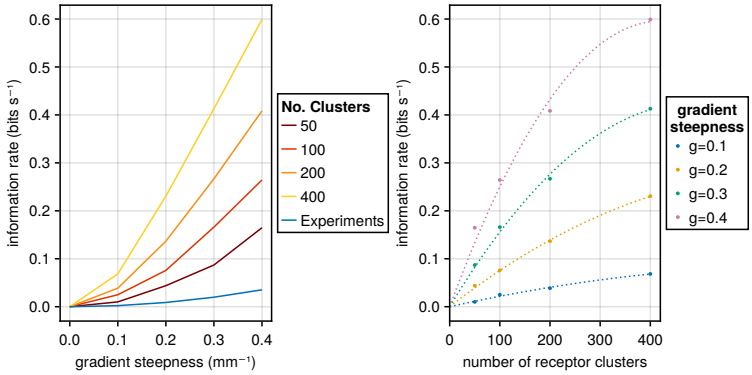



Figure 4.7: The information rate as a function of the number of receptor clusters $N_c$. The cluster size is fixed at $N = 15$. The left panel shows the increase of information rate as a function of gradient steepness for different values of $N_c$, including a line for the experimental data from Mattingly et al. [107]. The right panel shows the same data but highlights the increase of the information rate and when increasing the number of receptor clusters. A quadratic fit (shown as dotted lines) is used to extrapolate the information rate. All results were obtained using RR-PWS.

| $g\,(\mathrm{mm}^{-1})$ | $a\,(\mathrm{bit\,s}^{-1})$ | $b\,(\mathrm{bit\,s}^{-1})$ |
|---|---|---|
| 0.1 | $0.234 \times 10^{-3}$ | $0.160 \times 10^{-6}$ |
| 0.2 | $0.814 \times 10^{-3}$ | $0.598 \times 10^{-6}$ |
| 0.3 | $1.74 \times 10^{-3}$ | $1.77 \times 10^{-6}$ |
| 0.4 | $2.84 \times 10^{-3}$ | $3.39 \times 10^{-6}$ |

Table 4.2: Fit coefficients for the information rate as a function of the number of clusters $N_c$. These coefficients are for a quadratic function $R(N_c) = aN_c - bN_c^2$.

thus expected to be very similar to the experimentally measured one, and Fig. 4.5c shows that this is indeed the case. To quantify the accuracy of the Gaussian framework, we then recomputed the information transmission rate for the revised model, using exact PWS. We found that the result matches the Gaussian prediction very well. For these shallow and static chemical gradients, the Gaussian model is thus highly accurate. Our analysis validates *a posteriori* the Gaussian framework adopted by Mattingly et al. [107].

## 4.5 Discussion

The application of PWS to the bacterial chemotaxis system shows how crucial it is to have a simulation technique that is exact. Without the latter it would be impossible to determine whether the difference between our predictions and the Mattingly data [107] is due to the inaccuracy of the model, the inaccuracy of the numerical technique to simulate the model, or the approximations used by Mattingly and coworkers in analyzing the data. In contrast, because PWS is exact, we knew the difference between theory and experiment is either due to the inaccuracy of the model or the approximations used to analyze the data. By then employing the same Gaussian framework to analyze the behavior of the model and the experimental system, we were able to establish that the difference is due to the inaccuracy of our original model.

Our analysis indicates that the size of the receptor clusters in the *E. coli* chemotaxis system, $N \approx 15$, is larger than that based on previous estimates, $N \sim 6$ [113, 165, 82]. The early estimates of the cluster size were based on bulk dose-response measurements with a relatively slow ligand exchange, yielding $N \approx 6$ [113, 165]. More recent dose-response measurements, at the single cell level and with faster ligand exchange, yield an average that is higher, $\langle N \rangle \approx 8$, and with a broad distribution around it, arising from cell-to-cell variability [82]. Our estimate, $N \approx 15$, based on fitting the response kernel $K(t)$ to that measured by Mattingly et al. [107], therefore appears reasonable. At the same time, the number of clusters, obtained by fitting the noise correlation function $N(t)$ to the data of Mattingly et al. [107] is surprisingly low, $N_c \sim 10$, given the total number of receptors, $N_r \sim 10^3$–$10^4$ [96]. Interestingly, recent experiments indicate that the receptor array is poised near the critical point [83], where receptor switching becomes correlated over large distances. This effectively partitions receptors into a few large domains, which may explain our fitted values for $N$ and $N_c$.

It has been suggested that information processing systems are positioned close to a critical point to maximize information transmission [193, 112], although it has been argued that the sensing error of the *E. coli* chemotaxis system is minimized for

independent receptors [170]. Mattingly et al. have demonstrated that the chemotactic drift speed in shallow exponential gradients is limited by the information transmission rate [107], but whether the system has been optimized for information transmission, and how the latter affects chemotactic performance in other spatio-temporal concentration profiles, remain interesting questions for future work.

# 5 The Accuracy of the Gaussian Approximation

**Age Tjalma and Manuel Reinhardt**

*Efficient information processing is crucial for both living organisms and engineered systems. The mutual information rate, a core concept of information theory, quantifies the amount of information shared between the trajectories of input and output signals, and allows to quantification of information flow in dynamic systems. A common approach for estimating the mutual information rate is the Gaussian approximation, which assumes that the input and output trajectories follow Gaussian statistics. However, this method is limited to linear systems, and its accuracy in nonlinear or discrete systems remains unclear. In this work, we assess the accuracy of the Gaussian approximation for non-Gaussian systems by leveraging Path Weight Sampling (PWS), a recent technique for exactly computing the mutual information rate. In two case studies, we examine the limitations of the Gaussian approximation. First, we focus on discrete linear systems and demonstrate that, even when the system's statistics are nearly Gaussian, the Gaussian approximation fails to accurately estimate the mutual information rate. Second, we explore a continuous diffusive system with a nonlinear transfer function, revealing significant deviations between the Gaussian approximation and the exact mutual information rate as nonlinearity increases. Our results provide a quantitative evaluation of the Gaussian approximation's performance across different stochastic models and highlight when more computationally intensive methods, such as PWS, are necessary.*

This chapter was written by Age Tjalma and Manuel Reinhardt as shared first authors, in collaboration with Anne-Lena Moor (MPI-CBG Dresden), and Pieter Rein ten Wolde.

For the functioning of both living and engineered systems it is paramount that they collect and process information effectively. Increasingly, it has become evident that beyond instantaneous properties, the dynamic features of an input signal or system output often encode valuable information [194, 195, 107, 119, 152, 72]. Prime examples in biology include bacterial chemotaxis, which responds to temporal changes in concentration [162], the transcription factor NF-$\kappa$B, which encodes information about input signals in its dynamic response [31], and neuronal information processing, where information is encoded in the sequence and timing of spikes [177]. Beyond biology, dynamic input signals are critical for various sensing systems, such as those used in automated factories or self-driving cars.

To understand and evaluate the performance, potential improvements, and limitations of these systems in processing information, we need appropriate metrics that capture their full information processing capability. Information theory, introduced by Shannon [164], provides the most general mathematical framework for such metrics. The mutual information and mutual information rate measure how much one random variable reduces uncertainty about another, quantified in bits. It is relatively straightforward to quantify the information shared between scalar properties of the input and output, as has been done in various forms [210, 27, 45, 135, 26, 10, 157, 187, 186]. However, capturing all information in dynamical properties of the input and the output, is much more challenging. To do so, one must consider the information encoded time-varying trajectories of the variables of interest. Yet, due to the high dimensionality of the trajectory space, computing the mutual information between such trajectories is notoriously difficult.

A major advancement in this area has been the Gaussian approximation of the mutual information rate [194, 195], based on the assumption of input and output trajectories following jointly Gaussian statistics. This assumption makes it possible to compute the mutual information rate directly from the two-point correlation functions of the input and output. It is thus straightforward to apply the Gaussian approximation to experimental data. Moreover, given a mechanistic model of the underlying dynamics, the Gaussian approximation can be used to derive analytical expressions for the information rate [194, 195, 107]. Crucially however, the assumption of Gaussian statistics restricts the method to linear systems, as Gaussian statistics can only arise in such systems [30].

Understanding when the Gaussian approximation is accurate is critical because many real-world systems, such as biological and engineered sensory systems, exhibit nonlinear dynamics. This includes features such as bimodality, discrete jumps, or heavy tails, all of which deviate from purely Gaussian dynamics. Such non-Gaussian behavior typically results from intrinsic nonlinearities in the system, but determining the degree of a system's deviation from linearity is difficult [190, 35, 115], and the extent to which the approximation loses accuracy in nonlinear sys-

tems is unclear. Thus, although the Gaussian approximation offers a computationally simple framework to estimate information transmission, it remains an open question under what conditions this approximation is sufficiently accurate.

Until recently, addressing this question has been hard because there was no reliable benchmark for the exact information rate. Without a method to compute the true information rate of a non-Gaussian system, it is impossible to rigorously assess the accuracy of the Gaussian approximation. This gap was filled by the development of two independent methods [152, 119] for computing the information rate accurately even in systems that significantly deviate from Gaussian behavior. Here we leverage one of these methods: Path Weight Sampling (PWS) [152]. This is a Monte Carlo technique which is an exact method for calculating the mutual information rate in a wide range of stochastic models.

Using PWS, we can directly evaluate the accuracy of the Gaussian approximation in models that exhibit explicit non-Gaussian features, and study the approximation's robustness in typical applications.

In this article, we investigate the accuracy of the approximate Gaussian information rate through two case studies. The first focuses on Markov jump processes, where the statistics are non-Gaussian due to the discrete nature of the processes. Perhaps surprisingly, the Gaussian approximation fails to accurately estimate the mutual information rate in this case, even when the statistics are nearly Gaussian [119, 152]. We show that a recently developed reaction-based "discrete approximation" by Moor and Zechner [119] is much more accurate. This suggests that the Gaussian approximation fails because it cannot distinguish the individual reaction events.

The second case study examines a continuous diffusive process with a nonlinear transfer function. We demonstrate how intrinsic nonlinearity can cause significant deviations between the Gaussian approximation and the true mutual information rate. By varying the degree of nonlinearity as well as the system's response timescale, we provide a comprehensive quantitative understanding of the Gaussian approximation's limitations in nonlinear systems. Additionally, we show that for such systems, the Gaussian approximation differs significantly when derived from empirical correlation functions compared to when it is analytically obtained from the nonlinear model, highlighting that the correct application of the approximation is important.

Our work translates into concrete recommendations on when to use which method for the computation of the information rate. It therefore enables researchers to more confidently determine when a simpler approximate method is sufficient, or when a more sophisticated method like PWS [152] or the method developed by Moor and Zechner [119] should be used.

## 5.1 Methods

### 5.1.1 The mutual information rate

The mutual information between two random variables $S$ and $X$ is defined as

$$I(S,X) = \iint \mathrm{P}(s,x) \ln \frac{\mathrm{P}(s,x)}{\mathrm{P}(s)\,\mathrm{P}(x)}\, ds\, dx\,, \tag{5.1}$$

or, equivalently, using Shannon entropies

$$\begin{aligned} I(S,X) &= H(S) + H(X) - H(S,X) \\ &= H(S) - H(S|X) \\ &= H(X) - H(X|S)\,. \end{aligned} \tag{5.2}$$

In the context of a noisy communication channel, $S$ and $X$ represent the messages at the sending and receiving end, respectively. Then, $I(S,X)$ is the amount of information about $S$ that is communicated when only $X$ is received. If $S$ can be perfectly reconstructed from $X$, then $I(S,X) = H(S)$. On the contrary, if $S$ and $X$ are independent, $I(S,X) = 0$. The mutual information thus is always non-negative and quantifies the degree of statistical dependence between two random variables.

For systems that continuously transmit information over time, this concept must be extended to trajectories $\boldsymbol{S}_T = \{S(t) \mid t \in [0,T]\}$ and $\boldsymbol{X}_T = \{X(t) \mid t \in [0,T]\}$. The mutual information between trajectories is defined analogously as

$$I(\boldsymbol{S}_T, \boldsymbol{X}_T) = \left\langle \ln \frac{\mathrm{P}(\boldsymbol{s}_T, \boldsymbol{x}_T)}{\mathrm{P}(\boldsymbol{s}_T)\,\mathrm{P}(\boldsymbol{x}_T)} \right\rangle \tag{5.3}$$

where the expected value is taken with respect to the full joint probability of both trajectories. This quantity can be interpreted as the total information that is communicated over the time interval $[0,T]$.

Note that the total amount of information communicated over the time-interval $[0,T]$ is not directly related to the instantaneous mutual information $I(S(t),X(t))$ at any instant $t \in [0,T]$. This is because auto-correlations within the input or output sequences reduce the amount of new information transmitted in subsequent measurements. Moreover, information can be encoded in temporal features of the trajectories, which cannot be captured by an instantaneous information measure. Therefore, as previously pointed out [112, 49], the instantaneous mutual information $I(S(t),X(t))$ for any given $t$ does not provide a meaningful measure of information transmission. To correctly quantify the amount of information transmitted per unit time we must consider entire trajectories.

For that reason, the *mutual information rate* is defined via the trajectory mutual information. Let the input and output of a system be given by two continuous-time

stochastic processes $\mathcal{S} = \{S(t) \mid t \in \mathbb{R}\}$ and $\mathcal{X} = \{X(t) \mid t \in \mathbb{R}\}$. Then, the mutual information rate between $\mathcal{S}$ and $\mathcal{X}$ is

$$R(\mathcal{S}, \mathcal{X}) = \lim_{T\to\infty} \frac{1}{T} I(\boldsymbol{S}_T, \boldsymbol{X}_T)\,, \tag{5.4}$$

and quantifies the amount of information that can reliably be transmitted per unit time. The mutual information rate therefore represents an excellent performance measure for information processing systems.

In summary, the mutual information rate is *the* crucial performance metric for stochastic information processing systems. However, its information-theoretic definition does not translate into an obvious scheme for computing it. As a result, various methods have been developed to compute or approximate the mutual information rate.

### 5.1.2 Gaussian Approximation

One way to significantly simplify the computation of the information rate, is to assume that the input and output trajectories obey stationary Gaussian statistics. Under this assumption Eq. (5.3) simplifies to,

$$I(\boldsymbol{S}_T, \boldsymbol{X}_T) = \frac{1}{2} \ln \frac{|\boldsymbol{C}_{ss}||\boldsymbol{C}_{xx}|}{|\boldsymbol{Z}|}, \tag{5.5}$$

where $|\boldsymbol{C}_{ss}|$ and $|\boldsymbol{C}_{xx}|$ are the determinants of the covariance matrices of the respective trajectories $S_{[0,T]}$ and $X_{[0,T]}$, and

$$\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{C}_{ss} & \boldsymbol{C}_{sx} \\ \boldsymbol{C}_{xs} & \boldsymbol{C}_{xx} \end{pmatrix} \tag{5.6}$$

is the covariance matrix of their joint distribution.

In the limit that the trajectory length $N = T/\Delta$, with the discretization $\Delta$, becomes infinitely long ($N \to \infty$) and continuous ($\Delta \to 0$), the information rate as defined in Eq. (5.4) can be expressed in terms of the power spectral densities, or power spectra, of the processes $\mathcal{S}$ and $\mathcal{X}$ [194, 195]:

$$R(\mathcal{S}, \mathcal{X}) = -\frac{1}{4\pi} \int_{-\infty}^{\infty} d\omega \ln\left(1 - \frac{|S_{sx}(\omega)|^2}{S_{ss}(\omega) S_{xx}(\omega)}\right). \tag{5.7}$$

Here, $S_{ss}(\omega)$ and $S_{xx}(\omega)$ respectively are the power spectra of trajectories generated by $\mathcal{S}$ and $\mathcal{X}$, and $S_{sx}(\omega)$ is their cross-spectrum. The fraction

$$\phi_{sx}(\omega) = \frac{S_{sx}(\omega)|^2}{S_{ss}(\omega) S_{xx}(\omega)} \tag{5.8}$$

is known as the coherence, describing the distribution of power transfer between $\mathcal{S}$ and $\mathcal{X}$ over the frequency $\omega$.

For systems that are neither Gaussian nor linear, there are two ways to still obtain an approximate Gaussian information rate. The first is to directly measure two-point correlation functions from data or simulations, and use these to retrieve the power spectra in Eq. (5.7). The second is to use van Kampen's linear noise approximation (LNA) and approximate the dynamics of the system to first order around a fixed point [201], see also Section 5.4.1. In this work, we will analyze both of these methods.

### 5.1.3 Path Weight Sampling for diffusive systems

To evaluate the accuracy of the Gaussian information rate for non-Gaussian systems, an exact method for determining the true information rate is required. Recently, a method called Path Weight Sampling (PWS) was developed, which computes the exact mutual information rate using Monte Carlo techniques without relying on approximations [152].

In Ref. [152], PWS was introduced as a computational framework for calculating the mutual information rate in systems governed by master equations. Master equations provide an exact stochastic description of continuous-time processes with discrete state-spaces, commonly used in models ranging from biochemical signaling networks to population dynamics. However, many systems are not described by discrete state spaces and instead require a stochastic description based on diffusion processes or other stochastic models. Fortunately, PWS is not restricted to systems described by master equations and can be extended to a variety of stochastic models.

In general, PWS can be applied to any system that meets the following conditions: (i) sampling from the input distribution $\mathrm{P}(\boldsymbol{s}_T)$ is straightforward, (ii) sampling from the conditional output distribution $\mathrm{P}(\boldsymbol{x}_T \mid \boldsymbol{s}_T)$ is straightforward, and (iii) the logarithm of the conditional probability density $\ln \mathrm{P}(\boldsymbol{x}_T \mid \boldsymbol{s}_T)$, referred to as the path weight, can be evaluated efficiently. For any stochastic model that satisfies these three criteria, the PWS computation proceeds similarly to systems governed by master equations.

Briefly, PWS computes the trajectory Mutual Information using a Monte Carlo estimate of Eq. (5.3)

$$\frac{\sum_{i=1}^{N}\left[\ln \mathrm{P}\left(\boldsymbol{x}_T^i \,\middle|\, \boldsymbol{s}_T^i\right) - \ln \mathrm{P}\left(\boldsymbol{x}_T^i\right)\right]}{N} \tag{5.9}$$

where $\boldsymbol{s}_T^1, \ldots, \boldsymbol{s}_T^N$ are independently drawn from $\mathrm{P}(\boldsymbol{s}_T)$, and each $\boldsymbol{x}_T^i$ is drawn from $\mathrm{P}(\boldsymbol{x}_T \mid \boldsymbol{s}_T^i)$. As $N \to \infty$, this expression converges to the mutual information $I(\boldsymbol{S}_T, \boldsymbol{X}_T)$. In Eq. (5.9), the term $\ln \mathrm{P}(\boldsymbol{x}_T \mid \boldsymbol{s}_T)$ can be evaluated directly (per crite-

rion iii), but the marginal probability $\mathrm{P}(\boldsymbol{x}_T)$ has to computed in separately for each output trajectory $\boldsymbol{x}_T^i$. Typically, this has to be done numerically via marginalization, i.e., by computing the path integral

$$\mathrm{P}(\boldsymbol{x}_T) = \int d\boldsymbol{s}_T \;\; \mathrm{P}(\boldsymbol{s}_T)\, \mathrm{P}(\boldsymbol{x}_T \mid \boldsymbol{s}_T) \tag{5.10}$$

using Monte Carlo techniques. Evaluating the marginalization integral efficiently is essential for computing the mutual information using PWS and discussed in detail in Ref. [152]. In summary, PWS is a generic framework that can be used beyond systems defined by a master equation as long as a suitable generative model satisfying the three conditions above is available.

For this study, we extended PWS to compute the mutual information rate for systems with diffusive dynamics, described by Langevin equations. For such systems, the aforementioned conditions are inherently fulfilled and PWS can be applied. Specifically, in a Langevin system, both the input $S_t$ and the output $X_t$ are stochastic processes given by the solution to a stochastic differential equation (SDE). Using stochastic integration schemes like the Euler-Mayurama method, we can straightforwardly generate realizations $s(t)$ and $x(t)$ from the corresponding stochastic process. These realizations are naturally time-discretized with the integration time step $\Delta t$. For a time-discretized trajectory $\boldsymbol{x} = (x_1, \ldots, x_n)$, the path weight $\ln \mathrm{P}(\boldsymbol{x} \mid \boldsymbol{s})$ is—up to a Gaussian normalization constant—given by the Onsager-Machlup action [131]

$$\ln \mathrm{P}(\boldsymbol{x} \mid \boldsymbol{s}) = -\sum_{i=1}^{n-1} \frac{1}{2\Delta t} \left( \frac{\Delta x_i - v_i \Delta t}{\sigma(x_i)} \right)^2 + \text{const} \tag{5.11}$$

where we used $\Delta x_i = x_{i+1} - x_i$, and $v_i = f(x_i, s_i)$ is the deterministic drift, and $\sigma(x_i)$ represents the white noise amplitude. This expression captures the likelihood of a particular trajectory, given the stochastic dynamics of the system, and serves as the path weight in the PWS computation.

## 5.2 Case Studies

To investigate the conditions under which the Gaussian approximation deviates from the exact mutual information rate, we conducted two case studies. In both studies we compare the Gaussian approximation against the exact mutual information rate, computed via PWS. In the first case study we focus on a discrete linear system which is inspired by minimal motifs of cellular signaling.

### 5.2.1 Discrete reaction system

We consider a simple linear reaction system of two species, $S$ and $X$, whose dynamics are governed by 4 reactions

$$\varnothing \xrightarrow{\kappa} \mathrm{S} \tag{5.12}$$

$$\mathrm{S} \xrightarrow{\lambda} \varnothing \tag{5.13}$$

$$\mathrm{S} \xrightarrow{\rho} \mathrm{S} + \mathrm{X} \tag{5.14}$$

$$\mathrm{X} \xrightarrow{\mu} \varnothing\,. \tag{5.15}$$

The reaction system is linear because each reaction has at most one reactant. The trajectories of $S$ and $X$ are correlated because the production rate of $X$ depends on the copy number of $S$, and therefore information is transferred from $S$ to $X$. This set of reactions can be interpreted as a simple motif for gene expression where $S$ is a transcription factor and $X$ represents the expressed protein. In steady state, the mean copy numbers are given by $\bar{s} = \kappa\lambda^{-1}$ and $\bar{x} = \bar{s}\rho\mu^{-1}$.

The exact stochastic dynamics of this reaction system can be expressed by the chemical master equation [201]. This equation describes the time-evolution of the discrete probability distribution over the possible copy numbers of species $S$ and $X$, capturing the noise from the chemical reaction events. From this description we can obtain the mutual information rate from $S$ to $X$ without approximations using PWS [152].

While the chemical master equation is an exact representation of the reaction system, for large copy numbers the stochastic dynamics are well-approximated by a linearized model around the steady state. The resulting Langevin equations can be systematically derived from the master equation using the LNA which yields

$$\dot{s}(t) = \kappa - \lambda s(t) + \eta_s(t) \tag{5.16}$$

$$\dot{x}(t) = \rho s(t) - \mu x(t) + \eta_x(t) \tag{5.17}$$

where $s$ and $x$ are continuous variables representing the copy numbers of $S$ and $X$, and $\eta_s, \eta_x$ are independent delta-correlated white noise terms with $\langle \eta_s^2 \rangle = 2\lambda\bar{s}$ and $\langle \eta_x^2 \rangle = 2\mu\bar{x}$, see Section 5.4.1.

The Gaussian approximation of the mutual information rate is derived from the LNA description. Using this framework, Tostevin and ten Wolde [194] computed an analytical expression for the mutual information rate of the motif in units of nats $s^{-1}$:

$$R_{\mathrm{Gaussian}} = \frac{\lambda}{2}\left(\sqrt{1 + \frac{\rho}{\lambda}} - 1\right). \tag{5.18}$$

More recently, Moor and Zechner [119] have derived a different expression for the mutual information rate of this reaction system by analytically approximating the relevant filtering equation, which is derived from the master equation, thus recognizing the discreteness of molecules. This approach explicitly differentiates the contributions of individual reactions to the noise amplitude of each component, while the LNA lumps their contributions together. As we will discuss in more detail below, separately accounting for the noise from each reaction separately better captures the information transmitted via discrete systems, making this "discrete approximation" more accurate than the Gaussian approximation for this case study. Nevertheless, the result is still based on an approximation that is only accurate for large copy numbers. The expression for the mutual information rate in the discrete approximation appears remarkably similar to the expression obtained using the Gaussian framework:

$$R_{\text{discrete}} = \frac{\lambda}{2}\left(\sqrt{1+2\,\frac{\rho}{\lambda}}-1\right). \tag{5.19}$$

Note that this equation only differs from Eq. (5.18) by the additional factor 2 inside the square root.

The natural—but incorrect—expectation is that for large copy numbers both approximations converge to the true mutual information rate. However, the difference between Eqs. (5.18) and (5.19) already reveals that the two approximations do not converge. Indeed, previous work shows that even in the limit of infinite copy numbers, the Gaussian approximation only yields a lower bound to the information rate, which is not tight [152, 119].

We compare both approximations against exact PWS simulations for different parameters. In Fig. 5.1a, we vary the mean copy number of the readout, $\bar{x}$, by varying its synthesis rate $\rho$ and compute the mutual information rate using both approximations as well as PWS, while keeping the input copy number constant at $\bar{s} = 100$. We observe that the Gaussian approximation via the LNA [Eq. (5.18)] consistently underestimates the mutual information rate. This confirms that even when $\bar{s}$ and $\bar{x}$ are large, the Gaussian approximation only yields a lower bound to the information rate of the discrete linear system. In contrast, the discrete approximation [Eq. (5.19)] coincides with the true mutual information rate obtained from PWS simulations over all output copy numbers $\bar{x}$, even for $\bar{x} \ll 1$.

In Fig. 5.1b, instead of varying the copy number of the output, we vary the copy number of the input by varying the production rate $\kappa$. Note that both the Gaussian approximation and the discrete approximation are independent of $\kappa$. Yet, we observe that the true mutual information rate is not. For sufficiently large input copy numbers the discrete approximation coincides with the true information rate while

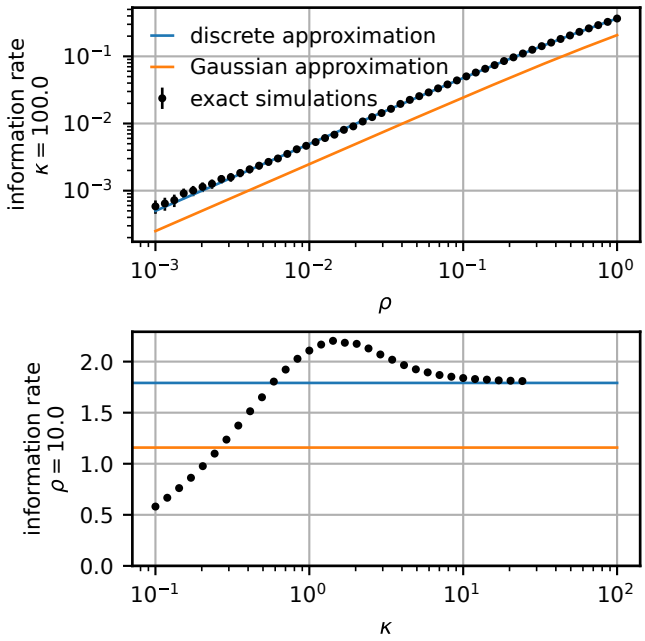



Figure 5.1: The mutual information rate of a simple linear reaction system defined by Eqs. (5.12) to (5.15). The black dots show the exact information rate, computed with PWS. We compare both, the Gaussian approximation of Tostevin and ten Wolde [194] and the discrete approximation of Moor and Zechner [119] against the exact result. In panel (a), we use parameters $\kappa = 100$, $\lambda = 1$, $\mu = 1$ while varying $\rho$. The mean output copy number is directly proportional to $\rho$ with proportionality factor $\kappa\lambda^{-1} = 100$. In panel (b), we fix $\rho = 10$, $\lambda = 1$, $\mu = 1$, and systematically vary $\kappa$. As a consequence, we vary the mean input copy number $\bar{s} = \kappa\lambda^{-1}$, and simultaneously also the mean output copy number $\bar{x} = \bar{s}\rho\mu^{-1} = 10\bar{s}$.

the Gaussian information rate remains only a lower bound. Thus, the discrete approximation is highly accurate for $\bar{s} \geq 10$. For small $\kappa$ where $\bar{s} < 10$, we find that the mutual information rate deviates from both, the LNA as well as the discrete approximation. Surprisingly, we find an optimal value of $\kappa$ for which the mutual information rate is maximized and exceeds both approximations. This implies that at low input copy numbers, the system is able to extract additional information from the discrete input trajectories, which is not accounted for by either of the approximations.

In all cases, we found that the Gaussian approximation deviates significantly from the true information rate for this discrete system. Seemingly paradoxically, the Gaussian approximation based on the LNA does not converge to the true information rate at high copy numbers, even though the LNA approximates the stochastic dynamics extremely well in this regime. In contrast, the discrete approximation from Moor and Zechner [119] does not suffer from this issue. It has been shown that, generally, the Gaussian approximation is a lower bound on the discrete approximation [119], prompting the question of which features of the discrete trajectories are not captured by the Gaussian approximation. [1]

### 5.2.2 Nonlinear continuous system

Next, we study a nonlinear variant of the reaction system above. In contrast to the previous case study, we deliberately avoid using discrete dynamics, as we already

[1] In still unpublished work together with Anne-Lena Moor and Christoph Zechner [120], we found that the root cause for the deviations of the Gaussian approximation lies in how the LNA approximates the reaction noise in the chemical master equation. While the dynamics of the chemical master equation give rise to discrete sample paths, i.e., piece-wise constant trajectories connected by instantaneous discontinuous jumps, the LNA approximation yields continuous stochastic trajectories. Our results imply that a discrete sample path of $X$ carries more information about $S$ than the corresponding continuous sample path $x(t)$ would carry about $s(t)$ in the LNA. In the collaborative effort [120] we found that this is ultimately due to the fact that in the discrete system, each reaction event is unambiguously recorded in the $X$ trajectories and thus different reactions modifying the same species can be distinguished. In contrast, in the continuous LNA description, all reactions that modify $X$ contribute to the noise term $\eta_x(t)$ in Eq. (5.17) but their contributions are lumped together and therefore cannot be distinguished from an observed $x(t)$-trajectory. Specifically, note that for the motif studied here, only the production reaction $S \to S + X$ conveys information. The decay reaction of the output $X \to \varnothing$ does not carry information on the input fluctuations since its propensity is independent of the input. Yet, it contributes to the overall fluctuations in the output. The Gaussian approximation only considers the total fluctuations in the output, while the discrete approximation correctly distinguishes between the fluctuations induced from production events and decay events. Therefore, the Gaussian approximation consistently underestimates the true information transmission, whereas the discrete approximation does not incur this systematic error. This subtle point is reflected in the difference between Eqs. (5.18) and (5.19).

observed that the Gaussian approximation is generally inaccurate in such systems. Instead, we focus solely on continuous Langevin dynamics to explore how an explicitly nonlinear input-output mapping affects the accuracy of the inherently linear Gaussian approximation. We hypothesize that the accuracy of the Gaussian approximation will deteriorate as the degree of nonlinearity increases. To test this hypothesis, we analyze a simple Langevin system with adjustable nonlinearity.

The system is defined by two coupled Langevin equations, one that describes the input, and one that describes the output. The stochastic dynamics of the input $s(t)$ are given by Eq. (5.16). The output dynamics of $x(t)$ are given by

$$\dot{x}(t) = \rho a(s) - \mu x(t) + \eta_x(t) \tag{5.20}$$

with the Hill function

$$a(s) = \begin{cases} \frac{s^n}{K^n + s^n} & \text{if } s \geq 0 \\ 0 & \text{if } s < 0 \end{cases} . \tag{5.21}$$

This function serves as a tuneable non-linearity with the Hill coefficient $n$. As $n \to 0$, the Hill function approaches a shallow linear mapping, while for large $n$, it becomes sigmoidal and highly non-linear. As $n \to \infty$, $a(s)$ approaches the unit step function centered at $s = K$. The so-called static input-output relation specifies the mean output $\bar{x}(s)$ for a given input signal $s$ and is given by $\bar{x}(s) = \rho a(s)/\mu$. The gain of this system is then defined as the slope of this relation at $s = \bar{s}$, i.e.,

$$g = \left.\frac{\partial \bar{x}(s)}{\partial s}\right|_{\bar{s}} = \frac{n a(\bar{s})[1 - a(\bar{s})]\rho}{\mu \bar{s}} , \tag{5.22}$$

as derived in Section 5.4.1. Importantly, for $\bar{s} = K$, the gain of the system is directly proportional to the Hill coefficient $n$, i.e., the gain is directly coupled to the degree of nonlinearity.

Figure 5.2 shows how, on average, the output $x(t)$ at a given time depends on the input $s(t)$ at that same time (solid colored curves). This is the so-called dynamical input-output relation of a system [102]. The solid black curve represents the static input-output relation. While the static input-output relation is purely determined by the instantaneous function $a(s)$, the dynamical input-output relation depends not only on this function, but also on the timescale of the response $\tau_x = \mu^{-1}$. The reason is that as the output responds more slowly to the input, the temporal input fluctuations are averaged out increasingly. Therefore, the response of the output becomes shallower for increasing $\tau_x$. Moreover, slower systems with more shallow responses react approximately linear to the input (Fig. 5.2).

While using the Langevin extension of PWS we can directly compute the mutual information rate of this nonlinear model, the Gaussian approximation can only be

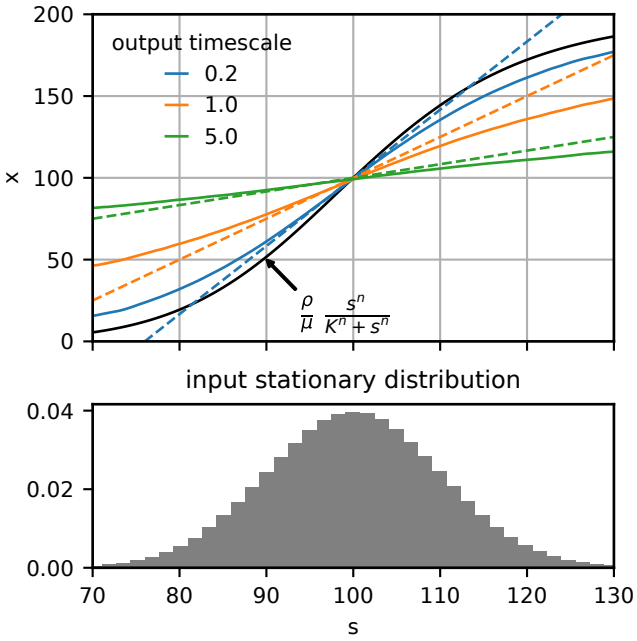



Figure 5.2: The dynamical input output relationship of the non-linear system with a fixed static gain of $g = 5$ [Eq. (5.22)]. The upper panel shows the static input-output relationship (solid black line) as well as the dynamical input output relationship, i.e., the effective mapping from input to output $S(t) \mapsto X(t)$ (at the same time) for different output timescales $\tau_x = \mu^{-1}$. The dynamical input-output mapping is defined as the conditional expectation $x_{\text{dyn}}(s) = \mathbb{E}[X(t) \mid S(t) = s]$ and was estimated non-parametrically from simulated trajectories of the system via Nadaraya–Watson kernel regression [125, 207] with a Gaussian kernel (bandwidth $h = 0.5$). Additionally, using the linear noise approximation we obtain linear input output mappings with a dynamical gain $\tilde{g} = g/(1 + \tau_x)$ (see Section 5.4.1) which are displayed as dashed lines. We observe that the linear mapping approximates the dynamical input output relation well for $s \approx \bar{s} = 100$ but cannot capture the nonlinear saturation effect. The lower panel shows the stationary distribution of $s(t)$.

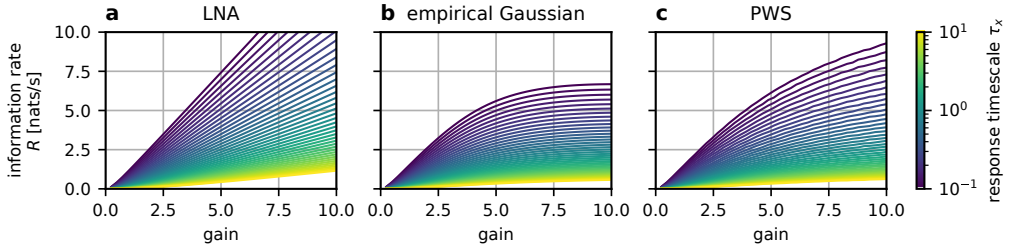



Figure 5.3: The information rate of a non-linear system as a function of its gain over a range of response timescales. We vary the **static** gain $g$ by varying the Hill coefficient $n$, see Eq. (5.22). A short response timescale corresponds to a fast system (purple) while a long response timescale corresponds to a slow system (yellow). The information rate was computed in three different ways: (**a**) via the Gaussian approximation using the LNA to estimate the required power spectra; (**b**) via the Gaussian approximation using simulations to numerically estimate the required power spectra; (**c**) exactly via PWS.

applied to linear systems. Therefore, to obtain the mutual information rate in the Gaussian approximation, we have to linearize the system. There are two approaches for linearizing the stochastic dynamics of this nonlinear system which result in different information estimates.

The first approach is to linearize Eq. (5.20) analytically via the LNA as shown in Section 5.4.1. Within this approach we can obtain an analytical expression for the information rate (see Section 5.4.1),

$$R_{\mathrm{LNA}} = \frac{\lambda}{2}\left(\sqrt{1 + g^2 \frac{\bar{s}\mu}{\bar{x}\lambda}} - 1\right). \tag{5.23}$$

This LNA based approach also yields a linearized dynamic input-output relation, shown as dashed lines in Fig. 5.2.

We observe that the linearized input-output relation closely matches the slope of the true nonlinear dynamical input-output relation at $s = \bar{s} = 100$, but overall it does not correspond to a (least-squares) linear fit of the nonlinear dynamical input-output relation. For all values of $s$, the linearized input-output relation has a slope greater than or equal to the slope of the dynamical input-output relation. Empirically, the LNA thus seems to over-estimate the dynamical gain of the system. The reason may be that the LNA approximates the static input-relation (Fig. 5.2 black curve), and estimates the linearized dynamical input-output relation based on this static approximation only.

The second "empirical Gaussian" approach to linearize the nonlinear system potentially avoids these issues. In this approach, we first numerically generate trajectories from the stochastic Eqs. (5.16) and (5.20) and use digital signal processing techniques to estimate the mutual information rate from the trajectories. We numerically estimate the (cross) power spectra of input and response using Welch's method [133, see Ch. 11]. From the estimated spectral densities $\hat{S}_{\alpha\beta}(\omega)$ we compute the coherence

$$\hat{\phi}_{sx}(\omega) = \frac{|\hat{S}_{sx}(\omega)|^2}{\hat{S}_{ss}(\omega)\hat{S}_{xx}(\omega)} \tag{5.24}$$

which we use to obtain the Gaussian approximation of the mutual information rate directly using Eq. (5.7).

The empirical power spectra characterize the linear response of a system, but not in the same way as the LNA. While for linear systems the power spectra obtained via the LNA match the empirical power spectra [206], for a nonlinear system, the empirical power spectra and the coherence can differ from the corresponding LNA calculations. The two linearization approaches are thus not equivalent. We tested the accuracy of the Gaussian mutual information rate estimates using both linearization approaches to elucidate the differences in these approaches.

Figure 5.3 displays the mutual information rate obtained via two linearized approximations as well as the exact PWS result. We vary the gain $g$ and the response time-scale $\tau_x$, both of which significantly affect the shape of the dynamical input-output relationship. As expected, a larger gain or a faster response time lead to an increase in the mutual information rate. At large gain, the information rate naturally saturates as $a(s)$ approaches a step function. The saturation effect is clearly seen in the PWS results, and is found to be even more pronounced in the empirical Gaussian approximation. The LNA-based Gaussian approximation, however, shows no saturation. This highlights that the LNA linearizes the system at the level of the input-output mapping $a(s)$ which results in an approximation that is unaffected by the sigmoidal shape of $a(s)$. In contrast, the empirical approximation is affected by nonlinear saturation effects because it is computed directly from simulated trajectories. We thus see that both approximations yield substantially different results at large gain.

In Fig. 5.4 we compare the absolute deviation between the approximations and the PWS result. For small gain we see that both approximations are accurate which is not surprising since the nonlinearity is very weak in this regime. Strikingly, for large gain, the LNA-based approximation always overestimates the mutual information rate while the empirical Gaussian approximation always underestimates the rate. In both cases the systematic error decreases as the response timescale becomes slower. This reflects the fact that for slow responders, the dynamic input-output re-

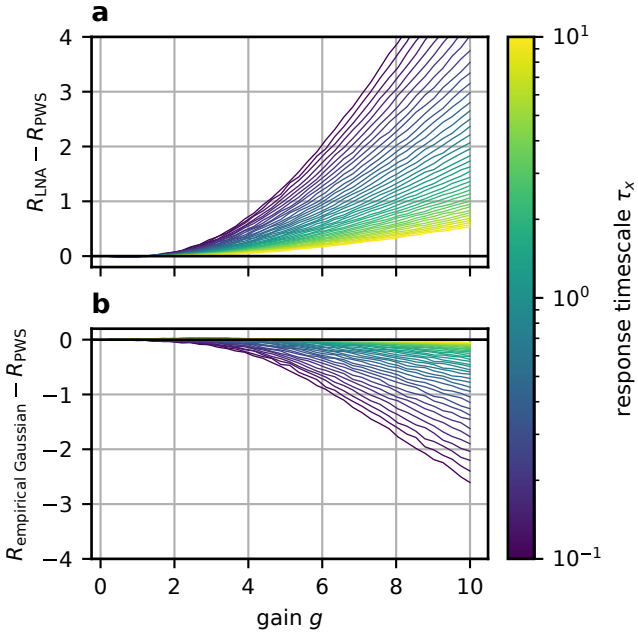



Figure 5.4: Deviation from the exact information rate for the approximate Gaussian information rate computed via LNA (top), and the approximate Gaussian information rate computed via empirical estimates of the power spectra (bottom). A deviation of 0 implies perfect accuracy. While the absolute deviation of both approximations increases with increasing gain and decreasing response timescale, the LNA based approach consistently overestimates the information rate whereas the empirical approach constitutes a lower bound. Moreover, in terms of absolute deviation, the empirical approach is more accurate across all parameter values.

lation is more linear (Fig. 5.2) than for fast responders.

Additionally, we computed the relative deviation, see Fig. 5.5 in Section 5.4.2. We find that in terms of relative error the curves for different response timescales largely overlap. In terms of relative approximation error, the gain, rather than response timescale, is the primary factor affecting the accuracy of the Gaussian approximation.

## 5.3 Discussion

We investigated the accuracy of the Gaussian approximation for the mutual information rate in two case studies, each highlighting a scenario where the approximation may be inaccurate. We were able to reliably quantify the inaccuracy in each case by computing the "ground truth" mutual information rate for these scenarios using a recently developed exact Monte Carlo technique called PWS [152].

We first considered linear discrete systems, which are relevant in biology due to the discrete nature of biochemical signaling networks. In our example, the Gaussian approximation cannot capture the full information rate, but only yields a lower bound. We show that a discrete approximation, developed by Moor and Zechner [119], is able to correctly estimate the mutual information rate of the network over a wide range of parameters. Since the Gaussian approximation captures the second moments of the discrete system, this finding demonstrates that a discrete system can transmit significantly more information than what would be inferred from its second moments alone. This perhaps surprising fact has been observed before [25, 119, 152] and it hinges on the use of a discrete reaction-based readout. As demonstrated in unpublished work [120], the increased mutual information rate found for a discrete readout stems from the ability of unambiguously distinguishing between individual reaction events in the readout's trajectory. However, it remains an open question whether biological (or other) signaling systems can effectively harness this additional information encoded in the discrete trajectories. For systems that cannot distinguish individual reaction events in downstream processing, the Gaussian framework might still accurately quantify the "accessible information".

A notable new observation in our first case study is the deviation between the discrete approximation of the mutual information rate derived by Moor and Zechner [119] and the exact result obtained using PWS [152] for inputs with low copy number $\bar{s}$. In the discrete approximation, the mutual information rate is independent of the input copy number $\bar{s}$, but the PWS simulations show that at low copy numbers there is an optimal $s^\star \approx 1$ which maximizes the mutual information rate. This surprising finding suggests that the information rate in discrete systems can be increased by reducing the copy number of the input sufficiently, such that it

only switches between a few discrete input levels. Notably, in the reverse case—low output copy number $\bar{x}$ but large $\bar{s}$—the discrete approximation always remains accurate. We leave a precise characterization of this finding for future work.

The second example focused on a continuous but nonlinear system, where we demonstrated that the accuracy of the Gaussian approximation depends on the linearization method. Linearizing the underlying system dynamics directly via the LNA leads to an overestimation of the information rate, while estimating the system's correlation functions empirically from data underestimates it. Regardless of the method, the Gaussian approximation is more accurate in terms of absolute deviation of the true information rate when the gain of the system is small and its response is slow compared to the timescale of the input fluctuations.

The result of our second case study—that the empirical Gaussian mutual information rate underestimates the true rate—is consistent with theoretical expectations. As shown by Mitra and Stark [115], and highlighted in Ref. [195], an empirical Gaussian estimate of the mutual information between a Gaussian input signal $S_G$ and a non-Gaussian output $X$ provides a lower bound on the channel capacity $C(S,X) = \max_{\mathrm{P}(s)} I(S,X)$ (subject to a power constraint on $S$). Specifically, they show that $C(S,X) \geq I(S_G;X) \geq I(S_G;X_G)$, where $(S_G,X_G)$ is a jointly Gaussian pair with the same covariance matrix as $(S_G,X)$. For purely Gaussian systems like $(S_G,X_G)$, the mutual information calculated using Eq. (5.7) is exact and equal to the channel capacity. However, for systems that have a Gaussian input but are otherwise non-Gaussian, the mutual information is larger or equal than the corresponding Gaussian model with matching second moments, as evidenced in Fig. 5.4. In general, the empirical Gaussian approximation yields a lower bound on the mutual information of the nonlinear system with a Gaussian input signal, as well as a lower bound on the channel capacity of the nonlinear system.[2]

We can distill several concrete recommendations for the computation of the information rate from our analysis. For linear discrete systems, the Gaussian approximation yields a lower bound on the true information rate which may accurately quantify the information available to systems that cannot distinguish individual discrete events. Alternatively, the reaction-based discrete approximation by Moor and Zechner [119] is highly accurate, even when the copy number of the output is extremely small. However, when the copy number of the input becomes small ($\lesssim 10$), both approximations break down and one must use an exact method. Exact methods for obtaining the information rate of any stochastic reaction-based systems are

[2]Note that this argument does not apply to the Linear Noise Approximation (LNA). The bound specifically requires the Gaussian model to use the covariance of the full, original system. When the system is first linearized using the LNA, the resulting linear model does not retain the same covariance as the original nonlinear system. As a result, the mutual information rate calculated with the LNA is generally not a lower (nor an upper) bound on the true mutual information rate.

PWS [152] or brute-force numerical integration of the stochastic filtering equation, as shown in [119]. For nonlinear continuous systems with small gain one can safely use the Gaussian approximation, either based on a linearization of the underlying dynamics or on empirically estimated correlation functions. Moreover, when the slowest input time-scale is more than a magnitude faster than the response time-scale, the non-linear response of the system is "averaged out" by the quick input fluctuations, and the Gaussian approximation yields accurate results. In this case, using a Gaussian approximation based on empirical correlation functions yields the most accurate result, and provides a rigorous lower bound for the mutual information. Finally, if the system is both highly nonlinear and has a fast response with respect to the input, one must resort to an exact method like PWS. We hope that our results will guide future research in determining the appropriate method for computing the mutual information rate.

Overall, our results greatly increase the usefulness of the Gaussian approximation for the information rate of non-Gaussian systems. The Gaussian approximation remains a useful method that can be applied directly and straightforwardly to experimental data. Here, we have quantified the prerequisites to safely use this approach. Moreover, we elucidate how an empirical Gaussian approximation constitutes a lower bound on the true information rate for systems with a sufficiently large input copy number.

## 5.4 Supplementary Information

### 5.4.1 Gaussian approximation

Here we derive the analytical expressions for the Gaussian information rate of the networks considered in the main text. To this end we first discuss the dynamics of the input signal $S$ and its power spectrum. Then, we perform a linear approximation of the dynamics of the readout species $X$ and derive the approximate Gaussian information rate between $S$ and $X$ for the nonlinear network. Finally, we derive the Gaussian information rate of the linear network from our expression of the Gaussian information rate of the nonlinear network.

#### Signal

The input signal is generated by a birth-death process,

$$\varnothing \underset{\lambda}{\overset{\kappa}{\rightleftharpoons}} \mathrm{S}, \tag{5.25}$$

Its dynamics in Langevin form are,

$$\dot{s} = \kappa - \lambda s(t) + \eta_s(t), \tag{5.26}$$

yielding the steady state signal concentration $\bar{s} = \kappa/\lambda$. The independent Gaussian white noise process $\eta_s(t)$ summarizes all reactions that contribute to fluctuations in $S$. The strength of the noise term in steady state is

$$\langle \eta_s^2 \rangle = \kappa + \lambda \bar{s} = 2\lambda \bar{s}. \tag{5.27}$$

The power spectral density, or power spectrum, of a stationary process $\mathcal{X}$ is defined as $S_{xx}(\omega) = \lim_{T\to\infty} \frac{1}{T}|\tilde{x}_T(\omega)|^2$, where $\tilde{x}(\omega)$ denotes the Fourier transform of $x(t)$. The power spectrum of a signal obeying Eq. (5.26) is thus given by

$$S_{ss}(\omega) = \frac{\langle \eta_s^2 \rangle}{\omega^2 + \lambda^2} = \frac{2\lambda \bar{s}}{\omega^2 + \lambda^2}. \tag{5.28}$$

## Linear approximation

We now consider the readout $X$, which is produced via a nonlinear activation function $a(s)$:

$$\begin{aligned} \mathrm{S} &\xrightarrow{\rho\, a(s)} \mathrm{S} + \mathrm{X}, \\ \mathrm{X} &\xrightarrow{\mu} \varnothing. \end{aligned} \tag{5.29}$$

We define the activation level $a(s)$ to be a Hill function,

$$a(s) = \frac{s(t)^n}{K^n + s(t)^n}. \tag{5.30}$$

Such a dependency, in which $K$ sets the concentration of $S$ at which the activation is half-maximal and $n$ sets the steepness, can for example arise from cooperativity between the signal molecules in activating the synthesis of $X$.

We have for the dynamics of $X$ in Langevin form

$$\dot{x} = \rho\, a(s) - \mu\, x(t) + \eta_x(t), \tag{5.31}$$

with $a(s)$ given by Eq. (5.30). The steady state concentration of $X$ is given by $\bar{x} = \bar{a}\rho/\mu$, where we have defined the steady state activation level $\bar{a} = a(\bar{s})$. It is useful to determine the static gain of the network, which is defined as the change in the steady state of the output upon a change in the steady state of the signal:

$$\begin{aligned} g &= \partial_{\bar{s}} \bar{x} = r/\mu, \\ &= n(1-\bar{a})\bar{x}/\bar{s}, \end{aligned} \tag{5.32}$$

where we have defined the approximate linear activation rate

$$r = n\bar{a}(1-\bar{a})\rho/\bar{s}, \tag{5.33}$$

and the steady state of the activation level is given by

$$\bar{a} = \frac{\bar{s}^n}{K^n + \bar{s}^n}. \tag{5.34}$$

Generally, we assume that $K = \bar{s}$, which entails that in steady state the network is tuned to $\bar{a} = 1/2$.

To compute the Gaussian information rate we approximate the dynamics of $X$ to first order around $\bar{x}$ via the classical linear noise approximation [201]. Within this approximation the dynamics of the deviation $\delta x(t) = x(t) - \bar{x}$ are,

$$\delta\dot{x} = r\,\delta s(t) - \mu\,\delta x(t) + \eta_x(t), \tag{5.35}$$

with the synthesis rate $r$ given by Eq. (5.33).

In the linear noise approximation the noise strength is a constant given by the noise strength at steady state,

$$\langle \eta_x^2 \rangle = \rho\bar{a} + \mu\bar{x} = 2\mu\bar{x}. \tag{5.36}$$

## Information rate

Following Tostevin & Ten Wolde [194, 195], we can express the Gaussian information rate as follows,

$$R(\mathcal{S};\mathcal{X}) = \frac{1}{4\pi}\int_{-\infty}^{\infty} d\omega \log\left(1 + \frac{|K(\omega)|^2}{|N(\omega)|^2} S_{ss}(\omega)\right), \tag{5.37}$$

where $|K(\omega)|^2$ is the frequency dependent gain and $|N(\omega)|^2$ is the frequency dependent noise of the output process $\mathcal{X}$. If the intrinsic noise of the network is not correlated to the process that drives the signal, the power spectrum of the network output obeys the spectral addition rule [180]. In this case the frequency dependent gain and noise can be identified directly from the power spectrum of the output, because it takes the following form:

$$S_{xx}(\omega) = |K(\omega)|^2 S_{ss}(\omega) + |N(\omega)|^2. \tag{5.38}$$

For a species $X$ obeying Eq. (5.35), we have

$$\begin{aligned} |K(\omega)|^2 &= \frac{r^2}{\mu^2 + \omega^2}, \\ |N(\omega)|^2 &= \frac{\langle \eta_x^2 \rangle}{\mu^2 + \omega^2} = \frac{2\mu\bar{x}}{\mu^2 + \omega^2}. \end{aligned} \tag{5.39}$$

The Wiener Khinchin theorem states that the power spectrum of a stochastic process and its auto-correlation function are a Fourier transform pair. We thus obtain for the variance in the readout, substituting the frequency dependent gain and noise [Eq. (5.39)] and the power spectrum of the signal [Eq. (5.28)] in Eq. (5.38) and taking the inverse Fourier transform at $t = 0$,

$$\sigma_x^2 = g\tilde{g}\sigma_s^2 + \sigma_{x|s}^2 = \frac{g^2\bar{s}}{1+\lambda/\mu} + \bar{x}, \tag{5.40}$$

where the signal variance equals its mean $\sigma_s^2 = \bar{s}$, and the mean readout concentration sets the intrinsic noise $\sigma_{x|s}^2 = \bar{x}$. We further have the static gain $g$ given by Eq. (5.32), and have defined the dynamical gain

$$\tilde{g} \equiv \frac{\langle \delta x(t) \delta s(t) \rangle}{\sigma_s^2} = \frac{g}{1+\lambda/\mu}, \tag{5.41}$$

which is the slope of the mapping from the time-varying signal value $s(t)$ to the time-varying readout $x(t)$; for Gaussian systems $\langle x(t)|s(t)\rangle = \tilde{g}s(t)$ [195, 102, 186].

To solve the integral in Eq. (5.37) we exploit that

$$\int_{-\infty}^{\infty} d\omega \log\left(\frac{\omega^2+a^2}{\omega^2+b^2}\right) = 2\pi(a-b). \tag{5.42}$$

Substituting the frequency dependent gain and noise given in Eq. (5.39) and the signal power spectrum of Eq. (5.28) in Eq. (5.37) and using Eq. (5.42) we obtain the information rate,

$$\begin{aligned} R(\mathcal{S};\mathcal{X}) &= \frac{\lambda}{2}\left(\sqrt{1+\frac{r^2\langle \eta_s^2\rangle}{\lambda^2\langle \eta_x^2\rangle}}-1\right), \\ &= \frac{\lambda}{2}\left(\sqrt{1+g^2\frac{\bar{s}\mu}{\bar{x}\lambda}}-1\right), \end{aligned} \tag{5.43}$$

where we used the noise strengths given in Eq. (5.27) and Eq. (5.36), we have the static gain $g$ of Eq. (5.32), and the synthesis rate $r$ of Eq. (5.33).

## Linear network

To disambiguate differences in the information rate caused by the linear approximation of our nonlinear reaction network on one hand and the Gaussian approximation of the underlying jump process on the other, we consider the information rate of a linear network. Any difference between the exact information rate and the Gaussian information rate must then be a result of the Gaussian approximation. To

this end we use the same input signal [Eq. (5.26)], but we consider a linear activation of the readout, i.e.

$$\begin{aligned} \mathrm{S} &\xrightarrow{\rho} \mathrm{S} + \mathrm{X}, \\ \mathrm{X} &\xrightarrow{\mu} \varnothing \end{aligned} \tag{5.44}$$

such that the Langevin dynamics of $X$ are

$$\dot{x} = \rho s(t) - \mu x(t) + \eta_x(t), \tag{5.45}$$

which yields the steady state concentration $\bar{x} = \bar{s}\rho/\mu$. For this linear readout, the static gain is simply set by the ratio of steady states of the input and the output, $g = \rho/\mu = \bar{x}/\bar{s}$. We can then obtain the information rate of this linear system by substitution of its static gain in Eq. (5.43), which yields

$$R(\mathcal{S};\mathcal{X}) = \frac{\lambda}{2}\left(\sqrt{1+\frac{\rho}{\lambda}} - 1\right). \tag{5.46}$$

This result is identical to that of Tostevin and ten Wolde [194, 195] (motif III).

### 5.4.2 Relative deviation of the Gaussian approximation for a nonlinear system

Due to the definition of the mutual information, an absolute difference in information maps to a relative difference in the reduction of uncertainty. For this reason, Fig. 5.4 in the main text (Section 5.2.2) focuses on the absolute deviation between the Gaussian approximation and the true mutual information. We compared two variants of the Gaussian approximation, the LNA-based approximation and the empirical Gaussian approximation. We found that in both cases the absolute deviation decreases with slower response timescales, reflecting the more linear input-output relationship in slow-responding systems.

However, the relative deviation of the Gaussian information rate from the true rate also offers valuable insights, which we explore here. In Fig. 5.5 we compare the relative deviation $R_{\mathrm{Gaussian}}/R_{\mathrm{PWS}}$ between the Gaussian approximation and the exact mutual information computed using PWS. We find that the relative deviation increases as the system gain increases, indicating that the Gaussian approximation also becomes relatively less accurate for larger gains. As already discussed above, the empirical Gaussian method consistently underestimates the true information rate, while the LNA-based approximation overestimates it.

Interestingly, we also observe that for the LNA approximation, at fast timescales the result is slightly more accurate, whereas the empirical Gaussian estimate is more accurate at slow timescales. We initially expected that in both cases slow timescales

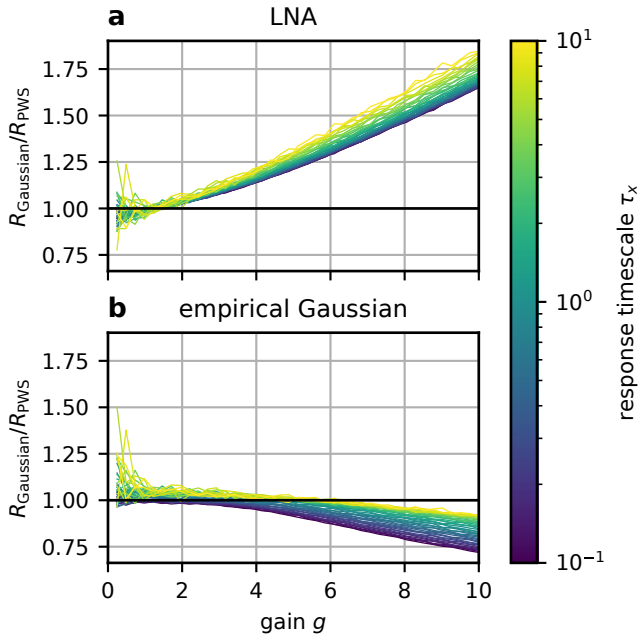



Figure 5.5: Relative deviation from the exact information rate for the approximate Gaussian information rate. The relative deviation was computed for both the LNA-based approximation and the empirical Gaussian approximation using numerically estimated power spectra, across varying system gains and response timescales (see Section 5.2.2 for details). The curves for different response timescales largely overlap, indicating that the relative approximation error is primarily influenced by system gain rather than response timescale. This highlights that system gain is the dominant factor in determining the accuracy of the Gaussian approximation.

would yield better agreement with PWS, as the input-output dynamics are more linear for slow timescales, and thus better approximated by the Gaussian model. The fact that this is not the case for the LNA approximation is intriguing, indicating the need for further investigation into the interplay between timescales, system nonlinearity, and the LNA.

# 6 ML-PWS: Quantifying Information Transmission Using Neural Networks

*Understanding the flow of information in natural and engineered systems is crucial for their design and analysis. The mutual information rate is the fundamental measure to describe the transmission of information for systems with time-varying signals. Yet, computing it accurately is extremely challenging due to the high-dimensional nature of trajectories. Previous methods include the Gaussian approximation which is limited to linear systems with Gaussian noise. A recent technique, Path Weight Sampling (PWS), in principle addresses these limitations, but it requires a stochastic model, which is often not available for the systems of interest. We propose leveraging recent advances in machine learning to obtain such a stochastic model directly from data, and provide a general-purpose tool for estimating the mutual information rate. Specifically, using unsupervised learning, we estimate the probability distribution of trajectories by training a neural stochastic model on time-series data. We demonstrate that by combining machine learning with PWS (ML-PWS) we can accurately compute the information transmission rate of nonlinear systems, even in the absence of a known mechanistic or phenomenological model. This approach represents a significant advance for data-driven quantification of information transmission in general nonlinear and non-Gaussian systems.*

This chapter is co-authored by Manuel Reinhardt, Gašper Tkačik (IST Austria), and Pieter Rein ten Wolde.

Information theory is the most general framework for studying signal processing systems and quantifying their performance. Shannon [164] introduced the mutual information as a measure to quantify how much information is communicated between two random variables, such as the input and output of a system at a given time. However, for most systems the mapping from input to the output cannot be directly described as a sequence of independent transmissions, rather information can generally be contained in temporal correlations within the input or output signals. Therefore, the mutual information between signal values at a given instant is in general ill-suited for measuring the amount of information transmission. To correctly quantify the information transmitted via time-varying signals requires computing the mutual information between the entire input and output trajectories of the system [194]. The rate at which this trajectory mutual information increases with the trajectory duration in the long-time limit defines the mutual information rate [164]. This rate represents the speed at which distinct messages are transmitted through the system, and it depends not only on the accuracy of the input-output mapping but also on the correlations within the input and output signals. In the absence of feedback this rate also equals the multi-step transfer entropy [106, 160].

The mutual information rate is the key measure to quantify information flow in dynamical systems. It is used to quantify biochemical signaling performance [107, 72, 119], to perform model reduction [159], to detect the causality of interactions [56, 75] or to test for nonlinearities in time series [136]. This includes applications such as financial markets, to assess dependencies between stock prices or market indices over time [103, 41, 40], or neuroscience, where it is used to measure the amount of information exchanged between different regions of the brain [148, 172]. Being one of the key performance measures in information theory, the mutual information rate is thus of paramount practical relevance.

Yet, computing the mutual information rate poses a significant challenge because trajectories are high-dimensional objects, making its accurate estimation difficult. Traditional techniques first estimate the joint probability distribution of input and output, e.g. via histograms or kernel density estimates, and then compute the mutual information from the distribution estimate [53, 118]. However, these distribution estimates are typically infeasible in high-dimensional spaces [137, 25], and have only been successfully used for computing the mutual information between trajectories in very simple systems [112]. Non-parametric estimators, such as the k-nearest-neighbor estimator [94] attempt to circumvent some of these issues but require selecting an appropriate metric in trajectory space and still suffer from uncontrolled biases for high-dimensional data [58, 25]. For systems that exhibit approximately Gaussian dynamics, the mutual information can be estimated directly from correlation functions [194, 195], though this method is limited to linear sys-

tems.[1] Finally, analytical and numerical approaches have been developed to accurately compute or approximate the trajectory mutual information from a dynamical model of the system [194, 46, 152, 119, 169]. In particular, we recently introduced Path Weight Sampling (PWS), a model-based Monte Carlo technique to exactly compute the mutual information rate [152]. But, since all these techniques require an accurate stochastic model describing the system, they cannot be directly applied to data.

Neural network-based methods offer a promising alternative. By leveraging gradient descent for learning complex high-dimensional distributions, we can potentially estimate the mutual information more accurately. So far, most of these approaches have primarily focused on training neural networks to optimize variational bounds of the mutual information [7, 129, 2, 13, 132, 143], often with the goal of learning effective latent state representations. However, these variational bounds are frequently not tight due to limited amounts of training data and the difficulty of optimizing over high-dimensional spaces [108, 143, 93, 25, 76], leading to significant underestimation of the mutual information. The Difference-of-Entropies (DoE) estimator by McAllester and Stratos [108] neither provides an upper nor a lower bound on the mutual information, but often results in more accurate mutual information estimates. As discussed in Section 6.3, this estimator shares some similarities with PWS. Given the effectiveness of neural networks for modeling sequential data, we thus asked whether machine learning could be combined with PWS to create a robust, data-driven estimator for the mutual information rate.

In this chapter, we present ML-PWS, an extension of PWS for computing the mutual information rate directly from experimental time-series data. By leveraging current machine learning methods, and combining them with PWS, we obtain a flexible architecture for computing the mutual information rate. We demonstrate that neural autoregressive sequence prediction models, which have been used in speech synthesis [198] or text generation [178], can be trained to learn a nonlinear stochastic model directly from data consisting of many input-output pairs of time-series. The model is trained by minimizing the Kullback-Leibler divergence between the model predictions and the observed trajectories. With this approach, the model learns the stochastic properties of the trajectories, enabling the computation of the mutual information rate using PWS. Here, the neural network is both used to generate stochastic trajectories, as well as to compute the path weights required for the PWS Monte Carlo estimate. Moreover, we show that by leveraging variational techniques we can significantly improve the PWS estimator itself, employing neural importance sampling [124] to efficiently marginalize the joint distribution of

[1] See also Chapter 5, where we extensively discuss the limitations of the Gaussian approximation for nonlinear systems.

input and output—a key step in the algorithm. We posit that these advances lead to a generic, flexible, and efficient framework for computing the mutual information rate directly from experimental data.

We test our approach by computing the mutual information rate for a minimal nonlinear model and comparing our results against the true mutual information as well as against the Gaussian approximation. We find that the autoregressive sequence model effectively learns the stochastic dynamics of the nonlinear system, and that PWS yields accurate mutual information estimates, including in regimes where the widely-used Gaussian approximation fails. Notably, we find that even for approximately linear systems, our model combined with PWS provides more accurate mutual information estimates than the Gaussian approximation because it suffers less from bias caused by using a finite-size dataset.

## 6.1 Methods

### 6.1.1 The Mutual Information Rate for Discrete-time Processes

The mutual information between two random variables $S$ and $X$ is defined as

$$I(S,X) = \iint \mathrm{P}(s,x) \ln \frac{\mathrm{P}(s,x)}{\mathrm{P}(s)\,\mathrm{P}(x)}\, ds\, dx\,, \tag{6.1}$$

or, equivalently, using Shannon entropies

$$\begin{aligned} I(S,X) &= H(S) + H(X) - H(S,X) \\ &= H(S) - H(S|X) \\ &= H(X) - H(X|S)\,. \end{aligned} \tag{6.2}$$

In the context of a noisy communication channel, $S$ and $X$ represent the messages at the sending and receiving end, respectively. Then, $I(S,X)$ is the amount of information about $S$ that is communicated when only $X$ is received. If $S$ can be perfectly reconstructed from $X$, then $I(S,X) = H(S)$. On the contrary, if $S$ and $X$ are independent, $I(S,X) = 0$. The mutual information thus is always non-negative and quantifies the degree of statistical dependence between two random variables.

For systems that repeatedly transmit information, this concept must be extended to sequences of messages $S_{1:n} = (S_1, \dots, S_n)$ and $X_{1:n} = (X_1, \dots, X_n)$. The mutual information between random sequences is defined analogously as

$$I(S_{1:n}, X_{1:n}) = \left\langle \ln \frac{\mathrm{P}(s_{1:n}, x_{1:n})}{\mathrm{P}(s_{1:n})\,\mathrm{P}(x_{1:n})} \right\rangle_{\mathrm{P}(s_{1:n}, x_{1:n})} \tag{6.3}$$

where the expected value is taken with respect to the full joint probability of both sequences. This quantity can be interpreted as the total information that is communicated via $n$ transmissions $S_i \mapsto X_i$.

Note that, unless the individual messages are independent, the total amount of information communicated is not equal to the sum of the mutual information between individual messages. Thus, in general

$$I(S_{1:n}, X_{1:n}) \neq \sum_{i=1}^{n} I(S_i, X_i)\,. \tag{6.4}$$

Intuitively, this makes sense. On one hand, auto-correlations within the input or output sequences reduce the amount of information transmitted per message such that the left term of the inequality may become smaller than the right term. On the other hand, information can be encoded in temporal features of the sequences, such that the left term could become larger than the right term. These observations show that generally the instantaneous mutual information $I(S_t, X_t)$ at any given time $t$ does not provide a meaningful measure of information flow, as has been pointed out before [112, 49]. To correctly quantify the amount of information transmitted per unit time we must take the whole sequence of messages over time into account.

For that reason, the relevant performance measure for an information processing system is the mutual information rate. Let the input and output of an information processing system be given by two discrete-time stochastic processes $\mathcal{S} = \{S_t \mid t = 1, 2, \ldots\}$ and $\mathcal{X} = \{X_t \mid t = 1, 2, \ldots\}$. Then, the mutual information rate between $\mathcal{S}$ and $\mathcal{X}$ is

$$R(\mathcal{S}, \mathcal{X}) = \lim_{t \to \infty} \frac{1}{t} I(S_{1:t}, X_{1:t})\,. \tag{6.5}$$

The mutual information rate quantifies the amount of information that can reliably be transmitted per unit time.

The definitions above however do not provide an obvious way of how to compute the mutual information rate in practice. Path Weight Sampling is a Monte Carlo estimator introduced recently for exactly computing the mutual information between trajectories [152].

### 6.1.2 Path Weight Sampling

In the previous chapters, we developed Path Weight Sampling, which addresses many of the shortcomings of previous techniques for computing the mutual information. PWS is an exact technique that supports nonlinear systems and, in contrast to the Gaussian approximation, correctly takes into account higher-order correlations that may be present. Given a mechanistic model that describes the stochastic

dynamics of a system, PWS makes it possible to directly compute the mutual information rate for this model.

PWS is based on the exact evaluation of conditional path probabilities and requires that we have a model of the system as well as its input statistics. Specifically, it has three requirements. To compute the Monte Carlo estimate of the mutual information one needs to

1. sample from the input distribution $\mathrm{P}(s_{1:n})$,
2. sample from the conditional output distribution $\mathrm{P}(x_{1:n}|s_{1:n})$, and
3. evaluate the conditional probability density $\mathrm{P}(x_{1:n}|s_{1:n})$, i.e., the path weight.

We thus require a model of the system, that describes how the output $x_i$ evolves stochastically for a given input sequence $s_{1:n}$, as well as a model that describes the stochastic input $s_{1:n} \sim \mathrm{P}(s_{1:n})$ to the system. Note, that we only need an estimate of the probability density for the output $\mathrm{P}(x_{1:n}|s_{1:n})$, but not for the input.

Given these models for input and output, the mutual information is computed using a Monte Carlo estimate of Eq. (6.3)

$$I_{MC}(S_{1:n}, X_{1:n}) = \frac{1}{N}\sum_{i=1}^{N} \ln \frac{\mathrm{P}(x^i_{1:n}|s^i_{1:n})}{\mathrm{P}(x^i_{1:n})} \tag{6.6}$$

where $(s^1_{1:n}, x^1_{1:n}), \ldots, (s^N_{1:n}, x^N_{1:n})$ are pairs of input-output trajectories that need to be drawn independently from the full joint distribution of trajectories, given by $\mathrm{P}(s_{1:n}, x_{1:n}) = \mathrm{P}(s_{1:n})\,\mathrm{P}(x_{1:n}|s_{1:n})$. Such draws can be realized by first generating an input sequence $s_{1:n} \sim \mathrm{P}(s_{1:n})$, and subsequently generating an output from the conditional model $x_{1:n} \sim \mathrm{P}(x_{1:n}|s_{1:n})$. As $N \to \infty$ this estimate converges to the true mutual information $I(S_{1:n}, X_{1:n})$, making PWS an exact Monte Carlo scheme. Equation (6.6) requires evaluating the conditional probability density $\mathrm{P}(x_{1:n}|s_{1:n})$, as well as the marginal probability density $\mathrm{P}(x_{1:n})$ for a potentially large set of Monte Carlo samples. How to evaluate these densities efficiently is the crux of PWS.

The first important observation is that we can typically directly evaluate the conditional probability density of output sequences $\mathrm{P}(x_{1:n}|s_{1:n})$ from the stochastic model of our system. For instance, suppose the output model is given by a Langevin equation $\dot{x} = f(x, s, t) + \sigma\xi(t)$ with delta-correlated unit white noise $\xi(t)$. A discretized path $x_{1:n}$ sampled from the model, can be represented by the initial state $x_1$ and the sequence of random numbers $\epsilon_1, \ldots, \epsilon_{n-1} \sim \mathcal{N}(0, 1)$ that were used to generate the path with a stochastic integration scheme. The conditional probability density of the path can then be written as

$$\mathrm{P}(x_{1:n}|s_{1:n}) = \mathrm{P}(x_1|s_1) \prod_{i=1}^{n-1} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\epsilon_i^2/2\right). \tag{6.7}$$

A similar formula exists if the model is given by a master equation [25, 152], which is based on the random numbers drawn in the Gillespie algorithm. There is also a class of deep generative models with tractable probability distributions. More generally, it is known that efficiently evaluating the conditional probability density $\mathrm{P}(x_{1:n}|s_{1:n})$ of sequences is tractable for any autoregressive sequence model without latent (unobserved) variables [57]. In this chapter we will exclusively deal with tractable conditional distributions, allowing us to evaluate the numerator in Eq. (6.6).

Unfortunately, the for denominator of Eq. (6.6), i.e., the marginal probability $\mathrm{P}(x_{1:n})$, no simple formulae like Eq. (6.7) typically exist. Yet, $\mathrm{P}(x_{1:n})$ is required for the Monte Carlo estimate of the mutual information. The only way of computing $\mathrm{P}(x_{1:n})$ exactly from the conditional probability density $\mathrm{P}(x_{1:n}|s_{1:n})$ is via marginalization over the input paths:

$$\mathrm{P}(x_{1:n}) = \int \mathrm{P}(x_{1:n}|s_{1:n})\,\mathrm{P}(s_{1:n})ds_{1:n}\,. \tag{6.8}$$

In practice, directly evaluating this integral is typically infeasible. A simple "brute force" Monte Carlo estimate of Eq. (6.8) can be obtained by sampling $s^1_{1:n}, \ldots, s^M_{1:n}$ from $\mathrm{P}(s_{1:n})$ and computing

$$\mathrm{P}(x_{1:n}) \approx \frac{1}{M}\sum_{i=1}^{M} \mathrm{P}(x_{1:n}|s^i_{1:n})\,. \tag{6.9}$$

For $M \to \infty$ this estimate converges to $\mathrm{P}(x_{1:n})$. This direct Monte Carlo estimate forms the basis of *Direct PWS*, the simplest variant of PWS, and can be sufficiently accurate for short trajectories. However, due to the combinatorial explosion, the required amount of samples $M$ to achieve an accurate result grows exponentially with trajectory length, and thus for long trajectories the brute force estimate becomes intractable. The problem is that for a given $x_{1:n}$ most of the density $\mathrm{P}(x_{1:n}|s_{1:n})$ will typically be concentrated in a very small region of $s_{1:n}$-space. Therefore, for longer trajectories more sophisticated Monte Carlo samplers must be used to achieve good results. Two more powerful variants of PWS were introduced in Chapter 3.

While PWS is a powerful exact method to compute the mutual information between trajectories, it cannot be applied directly to experimental data. The need for a stochastic model that provides an expression for $\mathrm{P}(x_{1:n}|s_{1:n})$ represents the most significant challenge for the use of PWS in practice. In many cases, a detailed mechanistic or phenomenological model of the experimental system is not available. To overcome this problem, we learn a stochastic model from data which can then be used in combination with PWS to compute the mutual information rate directly from experimental time series data.

### 6.1.3 Autoregressive neural networks for stochastic sequence modeling

For computing the mutual information between trajectories using PWS, we require a generative sequence model that specifies the stochastic dynamics of the input-output mapping. We assume that the input distribution $\mathrm{P}(s_{1:n})$ is known and can be sampled from. In experimental practice, we often have control over the input that is delivered to the system. The challenge is thus in accurately modeling the unknown stochastic dynamics of the system.

Hence, we require a model which, given an input sequence $s_{1:n} = (s_1, \dots, s_n)$, models the statistics of the stochastic output sequence $x_{1:n} = (x_1, \dots, x_n)$, i.e., we want a generative model for the distribution $\mathrm{P}(x_1, \dots, x_n \mid s_1, \dots, s_n) = \mathrm{P}(x_{1:n} \mid s_{1:n})$. However, for large $n$ the space of multivariate distributions in $x_{1:n}$ is vast, and we need to make simplifying assumptions to be able to fit a stochastic model to observed data. We develop a trainable machine learning model to obtain a generative sequence model from experimental data that meets the requirements for using PWS.

We can factorize the joint probability of a sequence $x_{1:n}$ as

$$\mathrm{P}(x_{1:n} \mid s_{1:n}) = \prod_{i=1}^{n} \mathrm{P}(x_i \mid x_{1:i-1}, s_{1:n}), \tag{6.10}$$

i.e., the stochastic dynamics are fully specified by the conditional stepping probabilities. Note that in a physical system obeying causality, the output $x_i$ cannot depend on future inputs. Thus, we can simplify Eq. (6.10) to

$$\mathrm{P}(x_{1:n} \mid s_{1:n}) = \prod_{i=1}^{n} \mathrm{P}(x_i \mid x_{1:i-1}, s_{1:i}). \tag{6.11}$$

A common approach for modeling stochastic sequences is to assume Markov statistics, meaning each element depends only on its immediate predecessor, simplifying the conditional to $\mathrm{P}(x_i \mid x_{1:i-1}, s_{1:i}) = \mathrm{P}(x_i \mid x_{i-1}, s_i)$. In the case of a stationary system, the transition probability $\mathrm{P}(x_i \mid x_{i-1}, s_i)$ is the same for all $i$, such that only one scalar distribution needs to be specified to define the Markovian process. While this assumption significantly reduces the complexity of the distribution space, Markov models are severely limited in that they cannot accurately model sequences with long-range dependencies or feedback. Yet, these non-Markovian features are often crucial to describe physical or biological processes.

Hence, we use a more general approach to directly learn Eq. (6.11) and parameterize the probability $\mathrm{P}(x_i \mid x_{1:i-1}, s_{1:i})$ at each time $i$ using neural networks. These

models are called *autoregressive models* and have been used for modeling the probability distribution of sequential data in a large variety of contexts [178, 70, 11, 198]. To efficiently model a sequence, we need to make two main choices: (a) which parametric family of distributions to use for modeling each conditional probability, and (b) which neural network architecture to use for obtaining the parameters for the chosen family of distributions, at each time.

### Gaussian autoregressive model

The choice of the parametric family of distributions depends on the nature of the data. For example, for scalar continuous data, a Gaussian distribution might be appropriate, whereas for discrete data, a categorical distribution is more suitable. More complex data might require richer distributional families, such as autoregressive flows or variational approaches, which allow for more flexible modeling of dependencies between sequence elements. For this chapter, we assume that the experimental data is scalar and continuous and can be sufficiently well modeled by Gaussian conditional distributions.

Hence, we consider an autoregressive model where each conditional distribution $\mathrm{P}(x_i \mid x_{1:i-1}, s_{1:i})$ is Gaussian. For $i = 1, \ldots, n$, the model uses a neural network to predict the mean $\mu_i(x_{1:i-1}, s_{1:i})$ and standard deviation $\sigma_i(x_{1:i-1}, s_{1:i})$ of the current sequence element $x_i$, given the previous elements. Thus, conditional on the input and its predecessors, the variable $x_i$ is normal distributed, with $\mathrm{P}(x_i|x_{1:i-1}, s_{1:i}) = \mathcal{N}(\mu_i(x_{1:i-1}, s_{1:i}), \sigma_i(x_{1:i-1}, s_{1:i}))$. The functions $\mu_i : \mathbb{R}^{i-1} \to \mathbb{R}$ and $\sigma_i : \mathbb{R}^{i-1} \to \mathbb{R}^+$ are implemented as neural networks and trained from experimental data.

Importantly, while each conditional distribution is Gaussian, the whole sequence is not Gaussian due to the nonlinear nature of the neural network. This means that Gaussian autoregressive models are a generalization of regular multivariate Gaussian models. In fact, a Gaussian autoregressive model can learn arbitrarily complex nonlinear relationships between the individual elements of a sequence, where the complexity is only limited by the neural network architecture. In contrast, a Gaussian model can only describe linear correlations between different sequence elements. Various neural network architectures can be used to implement the nonlinear functions $\mu_i$ and $\sigma_i$, and the choice must be made depending on the amount of training data, the complexity of the input-output mapping, as well as computational constraints.

### Network architecture

The network architecture is crucial because it directly determines the number of neural network parameters (or weights) that need to be learned. This, in turn, af-

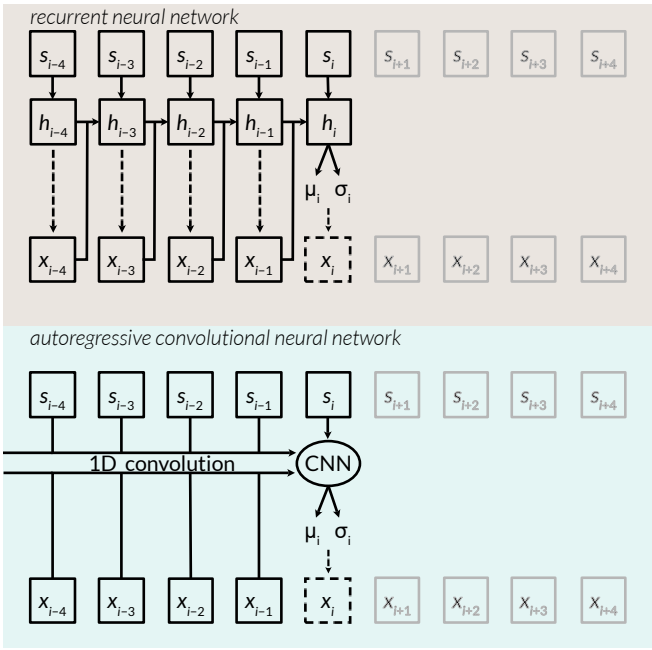



Figure 6.1: Two possible network architectures for autoregressive sequence models. In both cases, the next output $x_i$ is sampled from a Gaussian distribution with parameters $(\mu_i, \sigma_i)$ which the neural network computes from the history $s_{0:i}, x_{0:i-1}$.

fects both the flexibility of the model and the computational efficiency of training and inference. A more flexible architecture, with a larger number of weights, can potentially capture more complex relationships in the data but comes with the trade-off of increased computational cost and the risk of overfitting.

In principle, for modeling a sequence of length $n$, learning an autoregressive sequence model would require training $n$ separate neural networks, one for each conditional distribution $\mathrm{P}(x_i \mid x_{1:i-1}, s_{1:n})$, with $i = 1, \dots, n$. In practice, shared weights can drastically reduce the number of parameters to be learned. If the sequence is stationary or has other types of periodic features, the neural networks corresponding to different time steps can often share a majority, if not all, of their weights. This drastically simplifies training and evaluation of the autoregressive model.

We discuss two neural network architectures (schematically in Fig. 6.1) that are widely used for stochastic sequence prediction and make use of weight-sharing to reduce computational costs: recurrent neural networks (RNNs) and autoregressive convolutional neural networks (CNNs). Other sequence models like transformer

models [202, 39] could similarly be used but are not presented here.

**Recurrent neural networks** Recurrent Neural Networks (RNNs) process sequential data while maintaining a hidden state that evolves over time. At each time step, an RNN takes the current input and the previous hidden state to produce an output and update the hidden state. This mechanism allows the network to store relevant information about past inputs in the hidden state, effectively creating a form of memory. This makes the use of RNNs attractive for generic autoregressive sequence prediction models.

Given the sequences $s_{1:n}$ and $x_{1:n}$ the RNN takes an initial state $h_0 \in \mathbb{R}^d$ and generates a sequence $h_{1:n} = (h_1, \dots, h_n)$ from a recursive relation

$$h_i = f_\theta(s_i, x_{i-1}, h_{i-1}) \tag{6.12}$$

where $h_i \in \mathbb{R}^d$ for $i \in \{1, \dots, n\}$ and an activation function $f_\theta$. The activation function $f_\theta$ could for instance be parameterized by a simple neural network layer

$$f_\theta(s, x, h) = \tanh(Us + Vx + Wh + b) \tag{6.13}$$

with parameters $\theta = (U \in \mathbb{R}^{d\times 1}, V \in \mathbb{R}^{d\times 1}, W \in \mathbb{R}^{d\times d}, b \in \mathbb{R}^d)$ and applying tanh elementwise. Other possible choices for $f_\theta$ include LSTM units [80] or GRU units [28] which often allow the model to better learn long-term dependencies.

From the RNN we can obtain a stochastic representation of the output sequence $x_{1:n}$. We extend the recursive relation above by adding a sampling step to obtain $x_i$ from $h_i$

$$x_i \mid h_i \sim \mathcal{N}(\mu(h_i), \sigma(h_i)) \tag{6.14}$$

such that each $x_i$ is a normal-distributed random variable whose mean $\mu(h_i)$ and standard deviation $\sigma(h_i)$ are computed from the current hidden state $h_i$. In practice we use the following form for $\mu$ and $\sigma$

$$\mu(h) = W_\mu h + b_\mu \tag{6.15}$$
$$\sigma(h) = \exp(W_\sigma h + b_\sigma) \tag{6.16}$$

where the weights $W_\mu$ and $W_\sigma$ are $d$-dimensional row vectors and the exponential function ensures that the standard deviation is always positive. We denote the combination of all neural network parameters by $\boldsymbol{\theta} = (\theta, W_\mu, b_m u, W_\sigma, b_\sigma)$.

The recursive relations (6.12) and (6.14) fully define the conditional probability distribution $P(x_{1:n}|s_{1:n}, \boldsymbol{\theta})$ of the output sequence given the input sequence. Since $h_i$ depends on $s_i$ and $x_{i-1}$, as well as $h_{i-1}$, it encodes information about the entire past $s_{1:i}$ and $x_{1:i-1}$. Therefore, the model can incorporate long-range information for predicting the next output.

**Convolutional neural networks** Autoregressive convolutional networks model sequential data using masked 1D-convolutions. The masking ensures that the prediction at each time step only depends on the current and preceding elements of the sequence, maintaining causality. Unlike RNNs, which process sequences one time step at a time, CNNs can efficiently compute representations of the entire sequence in parallel, leading to substantial improvements in computational speed. This parallelism is particularly advantageous when working with long sequences. The architecture we describe here is inspired by MADE [138], as well as PixelCNN [198].

The autoregressive CNN processes the data by applying a series of masked convolutional layers. At time step $i$, the CNN predicts the Gaussian parameters $\mu_i, \sigma_i$ which describe the conditional $\mathrm{P}(x_i \mid s_{1:i}, x_{1:i-1})$. The mask is applied to each convolutional layer and restricts connections to only inputs in the past, or previously predicted outputs, i.e., non-causal connections are masked out by setting their weights to zero. The output of the first convolutional layer at time step $i$ depends on the local receptive field $(s_{i-k+1:i}, x_{i-k:i-1})$ of the input and output sequences, where $k$ is the kernel size, i.e.,

$$h_i = f_\theta(s_{i-k+1:i}, x_{i-k:i-1}) \tag{6.17}$$

where $f_\theta$ represents the 1D convolutional operation with learnable parameters $\theta$. The operation $f_\theta$ is composed of a set of learned convolutional filters and a non-linear activation function such as ReLU. Unlike for a RNN $h_i$ is not computed via a recursive relation since $h_i$ does not depend on $h_{i-1}$. Thus, we need a different mechanism to capture long-range dependencies.

Typically we apply multiple convolutional layers in series. This enables the model to capture long-range dependencies beyond the kernel size $k$ in the input sequence, as the depth of the network increases the temporal span of the receptive field. Moreover, stacking convolutional layers with non-linear activation functions allows the model to learn more complex representations of the data, potentially improving the accuracy of the model.

To generate the output sequence $x_{1:n}$, we add a sampling step similar to the RNN case. Specifically, the output $x_i$ at each time step is sampled from a distribution parameterized by the corresponding output from the CNN $h_i$:

$$x_i \mid h_i \sim \mathcal{N}(\mu(h_i), \sigma(h_i)) \tag{6.18}$$

The resulting conditional probability distribution $\rho_\theta(x_{1:n}|s_{1:n})$ is fully defined by the convolution weights $\theta$ and the sampling step.

### Evaluating and training the network

The autoregressive neural network described above can be viewed as generative models that approximate the conditional distribution $\mathrm{P}(x_{1:n}|s_{1:n})$, i.e., we can use them to draw independent samples from $\rho_\theta(x_{1:n}|s_{1:n}) \approx \mathrm{P}(x_{1:n}|s_{1:n})$ where $\theta$ represents the network weights. Generating $x_{1:n}$ given $s_{1:n}$ is done sequentially. For $i = 1, \dots, n$, the sampling procedure alternates between computing the parameters of the conditional distribution, $\hat{\mu}_i = \mu_i(x_{1:i-1}, s_{1:i})$, $\hat{\sigma}_i = \sigma_i(x_{1:i-1}, s_{1:i})$, and sampling the next $x_i \sim \mathcal{N}(\hat{\mu}_i, \hat{\sigma}_i)$.

The conditional probability of the resulting output sequence is given by

$$\begin{aligned} \rho_\theta(x_{1:n}|s_{1:n}) &= \prod_{i=1}^{n} \mathrm{P}(x_i|x_{1:t-1}, s_{1:t}, \theta) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\hat{\sigma}_i} \exp\left(-\frac{(x_i - \hat{\mu}_i)^2}{2\hat{\sigma}_i^2}\right). \end{aligned} \tag{6.19}$$

This probability can be evaluated on the fly while generating the sequence. Moreover, for a given pair of input and output sequences, we can also use the model to directly evaluate $\rho_\theta(x_{1:n}|s_{1:n})$ which is required for PWS. Note that in practice, to numerically accurately compute a product with many terms it is typically necessary to perform the computation in log space.

Note that while generating a sequence is inherently a sequential process, the path likelihood $\rho_\theta(x_{1:n}|s_{1:n})$ can be evaluated in parallel in some neural architectures like the CNN. Specifically, for a given pair of sequences $(s_{1:n}, x_{1:n})$, the conditional probability in Eq. (6.19) is parallelizable, since the computations for $\hat{\mu}_i$ and $\hat{\sigma}_i$ for $i = 1, \dots, n$ are independent of each other. This allows for efficient training on parallel computing hardware.

To train the model, we minimize the negative log likelihood of its predictions when evaluated on the training data. Specifically, we assume that the training data consists of $N$ pairs of sequences $(s^k_{1:n}, x^k_{1:n})$ for $k = 1, \dots, N$. The loss function to be minimized is then given by the sum of the individual negative log likelihoods for the trajectory pairs:

$$\mathcal{L}(\theta) = -\sum_{k=1}^{N} \ln \rho_\theta(x_{1:n}|s_{1:n}). \tag{6.20}$$

This training objective is equivalent to minimizing an empirical estimate of the Kullback-Leibler (KL) divergence between the distribution of the training data and the distribution defined by the model, thus training the model to fit the underlying data distribution [17].

There are a few practical considerations for efficiently training the model. Training is performed in iterations and it is often beneficial to introduce stochasticity between iterations to speed up gradient descent and regularize the loss function to prevent overfitting [21, 51]. For this reason, as typically done for training neural networks, the loss function in Eq. (6.20) is only computed for a subset of the training data, in mini-batches of size $M$, instead of the whole training set of size $N$. At the beginning of each iteration, the data subset that is used is randomly selected (without replacement) from the whole data set.

### 6.1.4 Efficient Marginalization Using Variational Inference

In the preceding section, we have shown how machine learning techniques can be leveraged to obtain a Path Weight Sampling (PWS) estimate of the mutual information rate directly from empirical data. By employing machine learning algorithms to learn the underlying stochastic model of the data, we enable accurate computation of the mutual information rate using the PWS framework.

In this section, we employ machine learning differently, to optimize the PWS method itself. Specifically, we address the computationally most demanding task: the evaluation of the marginalization integral. While we have presented alternative techniques for computing this integral in Chapters 2 and 3, here we leverage recent advances in machine learning and introduce an efficient marginalization strategy based on variational inference.

The idea of the variational marginalization procedure is to train a second neural network, the *inference model*, that parameterizes an importance sampling distribution over $s_{1:n}$ to enable the efficient computation of the marginal probability $\mathrm{P}(x_{1:n})$. This inference model, often referred to as the backward model, operates in reverse directionality to the actual system, i.e., it generates an input given an output. This is in contrast to the previous section, which focused on generative models for the output $x_{1:n}$ given the input sequence, governed by $\mathrm{P}(x_{1:n}|s_{1:n})$. Roughly, the inference network takes an output trajectory $x_{1:n}$ and generates input trajectories $s_{1:n}$ that could have likely produced the corresponding output. When this network is used as an importance sampler, we can significantly accelerate the computation of the marginal probability and thus the mutual information. We denote the inference model's generative distribution as $q(s_{1:n}|x_{1:n})$.

To compute the marginal probability with help of the inference model, we write

$\mathrm{P}(x_{1:n})$ as the expectation with respect to a probability density $q(s_{1:n}|x_{1:n})$, i.e.,

$$\begin{aligned}\mathrm{P}(x_{1:n}) &= \int \mathrm{P}(x_{1:n}|s_{1:n})\,\mathrm{P}(s_{1:n})\,ds_{1:n} \\ &= \mathbb{E}_{\mathrm{P}(s_{1:n})}\left[\mathrm{P}(x_{1:n}|s_{1:n})\right] \\ &= \mathbb{E}_{\mathrm{P}(s_{1:n})}\left[\frac{\mathrm{P}(x_{1:n}|s_{1:n})}{q(s_{1:n}|x_{1:n})}q(s_{1:n}|x_{1:n})\right] \\ &= \mathbb{E}_{q(s_{1:n}|x_{1:n})}\left[\frac{\mathrm{P}(s_{1:n})\,\mathrm{P}(x_{1:n}|s_{1:n})}{q(s_{1:n}|x_{1:n})}\right]\end{aligned} \tag{6.21}$$

where $q(s_{1:n}|x_{1:n})$ can be chosen arbitrarily in principle. Equation (6.21) is estimated using Monte Carlo sampling, by using the inference network to generate a set of trajectories $\{s^1_{1:n}, \dots, s^M_{1:n}\}$ from $q(s_{1:n}|x_{1:n})$ and computing the respective importance weights $w_1, \dots, w_M$ according to $w_k = w(s^k_{1:n}, x_{1:n})$ where

$$w(s_{1:n}, x_{1:n}) = \frac{\mathrm{P}(s_{1:n})\,\mathrm{P}(x_{1:n}|s_{1:n})}{q(s_{1:n}|x_{1:n})}\,. \tag{6.22}$$

The marginal probability is then estimated by

$$\mathrm{P}(x_{1:n}) \approx \frac{1}{M}\sum_{k=1}^{M} w_k\,. \tag{6.23}$$

In this process, $q(s_{1:n}|x_{1:n})$ serves as the importance sampling distribution. Regardless of the choice of $q(s_{1:n}|x_{1:n})$, this estimate always converges to $\mathrm{P}(x_{1:n})$ in the limit $M \to \infty$. However, crucially, for finite $M$ the choice of $q(s_{1:n}|x_{1:n})$ determines the variance and thus efficiency of the estimate.

If, as done in Direct PWS (Chapter 2), we choose the "prior" probability $\mathrm{P}(s_{1:n})$ as the importance sampling distribution, the resulting estimate is typically highly inefficient. This is because with that choice the importance weights are usually very unevenly distributed, with heavy tails which significantly increases the variance of the estimator, see also Chapter 3. It is well-known that the prior is generally a poor importance sampling distribution since it often allocates significant probability mass to regions of the configuration space that contribute little to the likelihood [14, 63]. The hypothetical optimal choice for the importance sampling distribution $q(s_{1:n}|x_{1:n})$ is the true "posterior" distribution $\mathrm{P}(s_{1:n}|x_{1:n})$, as this makes the importance weights constant, resulting in a theoretically zero variance estimator. However, since the true posterior is typically intractable, we instead aim to approximate the posterior by a tractable distribution, to reduce the variance as much as possible.

The idea of variational inference is to train the inference model using a loss function that minimizes the Kullback-Leibler (KL-)divergence between the variational distribution $q(s_{1:n}|x_{1:n})$ and $\mathrm{P}(s_{1:n}|x_{1:n})$. Since the KL-divergence is always non-negative, and is only exactly zero if $q(s_{1:n}|x_{1:n}) = \mathrm{P}(s_{1:n}|x_{1:n})$, this criterion optimizes the importance sampling distribution. The idea of using an inference network to approximate the posterior was popularized with the introduction of the variational autoencoder (VAE) by Kingma and Welling [88], a powerful technique for approximating complex distributions. The training objective used in variational inference is the Evidence Lower Bound Objective (ELBO) which provides a lower bound on the "evidence" $\mathrm{P}(x_{1:n})$. Maximizing this bound brings the variational approximation closer to the true posterior. The ELBO can be derived by applying Jensen's inequality to the last line of Eq. (6.21):

$$\ln \mathrm{P}(x_{1:n}) \geq \mathbb{E}_{q(s_{1:n}|x_{1:n})}\left[\ln \frac{\mathrm{P}(s_{1:n})\,\mathrm{P}(x_{1:n}|s_{1:n})}{q(s_{1:n}|x_{1:n})}\right] = \mathcal{L}_{\mathrm{ELBO}} \tag{6.24}$$

It is easy to show that maximizing the ELBO is equivalent to minimizing the KL-divergence between the variational distribution and the true posterior. Although the estimate in Eq. (6.23) is always unbiased, i.e., it converges to the marginal probability $\mathrm{P}(x_{1:n})$ as $M \to \infty$ regardless of the choice of $q(s_{1:n}|x_{1:n})$, in practice, convergence will be slow unless the inference network accurately approximates the posterior. Thus, optimizing the inference network by maximizing the ELBO is crucial for efficient marginalization.

To closely approximate the posterior, the inference network needs enough flexibility to capture the key features of the true posterior. Efficient marginalization in trajectory space, in particular, requires an inference network capable of modeling high-dimensional distributions with complex dependencies between variables. Specifically, for efficiently marginalizing in trajectory space, we require an inference network that can model high-dimensional distributions with complex dependencies between variables. However, selecting an appropriate inference network can be challenging: if the network lacks sufficient flexibility, it may oversimplify the posterior, which may be hard to diagnose. Various approaches for designing flexible inference networks have been studied previously and could be applied here [153, 42, 89, 197, 87, 92, 139].

In our example, we use an inference model for marginalization that closely resembles the generative model discussed in the previous section and uses an autoregressive sequence model for $q(s_{1:n}|x_{1:n})$. The inference network consists of a RNN-based encoder-decoder architecture [179] which helps in approximating the posterior $\mathrm{P}(s_{1:n}|x_{1:n})$ by effectively using the information in $x_{1:n}$ to guide the generation of $s_{1:n}$.

The encoder, a RNN, first processes the sequence $x_{1:n}$ in reverse to create a latent representation $h_{1:n}$, which provides contextual information to the decoder. This latent sequence is used by the decoder to generate a new sequence $\tilde{s}_{1:n} \sim q(\tilde{s}_{1:n}|x_{1:n})$ that is compatible with $x_{1:n}$. Because the encoder processes $x_{1:n}$ in reverse, the latent sequence $h_{1:n}$ enables the decoder to incorporate information about all future values $x_j$ for $j \geq i$ into each $\tilde{s}_i$. Importantly, by including future information from $x_{i:n}$ to sample $\tilde{s}_i$, we can effectively capture the dependence structure of the true posterior $\mathrm{P}(\tilde{s}_{1:n}|x_{1:n})$ into our variational approximation. Thus, using an encoder is a key ingredient for accurately approximating the posterior.

The decoder then autoregressively generates the sequence $\tilde{s}_{1:n} \sim q(\tilde{s}_{1:n}|x_{1:n})$, using the context sequence $h_{1:n}$ provided by the encoder. Thus, each $\tilde{s}_i$ is generated conditioned on both $h_i$ and the previously generated $\tilde{s}_{i-1}$. As in the forward model, $\tilde{s}_i$ is generated probabilistically; however, rather than a Gaussian distribution, the decoder uses a mixture of logistic distributions, inspired by the Flow++ architecture [78]. This encoder-decoder setup provides a flexible, expressive model capable of accurately approximating the posterior distribution.

Given that all parameters in our model are continuous, we optimize the weights of the inference network using stochastic gradient variational Bayes (SGVB) [88], a widely used method for training variational models. SGVB leverages the reparameterization trick to estimate the gradient of the ELBO, which makes the training procedure efficient.

## 6.2 Application to a Minimal Nonlinear Model

We evaluate our approach using synthetic training data to train a generative model and then using PWS to compute the mutual information rate.

### 6.2.1 Nonlinear Model

To generate training data for the neural network, we combine a linear auto-regressive input, with a stochastic nonlinear output model. Specifically, we considered an input that evolves according to AR(1) statistics:

$$S_t = \phi S_{t-1} + \xi_t \tag{6.25}$$

where $\xi_t$ are iid random variables from a unit Gaussian distribution, and $\phi \in [0, 1)$ is a parameter. In steady state, the autocovariance of this process is given by

$$\langle S_\tau S_{\tau+t} \rangle = \frac{\phi^{|t|}}{1-\phi^2}\,. \tag{6.26}$$

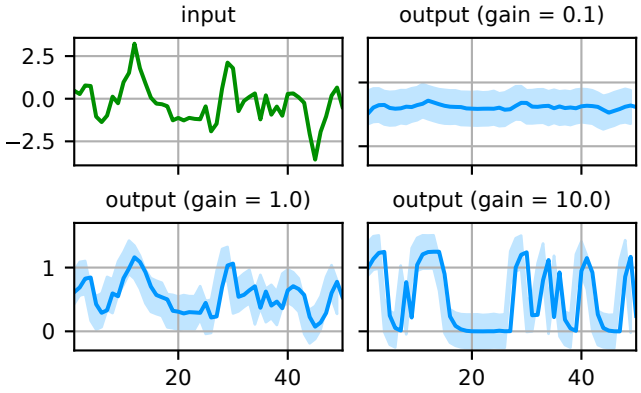



Figure 6.2: Example time series from the training set. The training set was generated using parameters $\phi = 0.5, \rho = 0.2$, and $\vartheta = 0.2$. In the upper left panel one stochastic realization of the input process is shown. The other panels show the mean output as well as 10/90-th percentiles for the output at different values of the input gain $\gamma$. The effect of the gain on the output can be clearly seen. For the highest gain $\gamma = 10.0$, we observe saturation of the output.

The output $X_t$ is governed by the equation

$$X_t = \sigma(\gamma S_t) + \rho X_{t-1} + \vartheta \eta_t \tag{6.27}$$

where $\eta_t$ are iid Gaussian random numbers, $\gamma$, $\rho$ and $\vartheta$ are positive real parameters, and

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{6.28}$$

is the logistic function. The gain $\gamma$ effectively controls the strength of the nonlinearity; see Fig. 6.2. This process models a response that saturates as the input grows. In fact, $\sigma(x)$ is equivalent to the Hill function commonly used in biochemistry to describe saturating enzyme kinetics [47].

### 6.2.2 Training the Neural Networks

We trained our machine learning model with synthetic data generated according to Eq. (6.27) for various values of the gain, denoted by $\gamma$. For each value of $\gamma$, we created a distinct training set of 1000 pairs of time series $(s_{1:50}, x_{1:50})$ and trained one autoregressive model per training set. Once the models were trained, we estimated the mutual information for each of them using PWS, employing variational inference to perform the marginalization.

For the generative model, we use a recurrent neural network (RNN). Using Gated Recurrent Unit (GRU) cells [28] with a hidden size of 64 for temporal processing, along with a dense layer that outputs two values, representing the mean $\mu$ and log-variance $\ln \sigma^2$ of a Gaussian distribution. For each time step $i$, the model receives input signals $s_i$ and $x_{i-1}$ and predicts the next output value $x_i$ by sampling from this Gaussian. The model is trained by iteratively optimizing over 100 epochs over the entire training set. The training set is split into mini-batches of size 64 which are shuffled after each epoch. We optimize the model using the Adam optimizer [86] ($b_1 = 0.9, b_2 = 0.999$) with weight decay regularization [99] of $1 \times 10^{-4}$ and using a cosine decay learning rate schedule [98] that smoothly decreases the learning rate from $1 \times 10^{-2}$ to $1 \times 10^{-3}$ throughout the training process.

The inference model used for marginalization is also modeled as a RNN. This model consists of an encoder-decoder architecture for approximating the posterior $\mathrm{P}(s_{1:n}|x_{1:n})$. The encoder first processes the input sequence $x_{1:n}$ in reverse using a GRU cell with 64 hidden units to create a latent sequence $h_{1:n}$. The decoder then uses another RNN with a hidden size of 64 (using a GRU cell), to autoregressively generate a sequence $\tilde{s}_{1:n}$ using the context sequence $h_{1:n}$. The sequence $\tilde{s}_{1:n}$ is generated probabilistically by sampling from a mixture of logistic distributions with five components. The inference network is trained for 100 epochs with mini-batches of size 64. For each $x_{1:n}$ in the training set, 16 Monte Carlo draws from the inference network $\tilde{s}_{1:n} \sim q(\tilde{s}_{1:n}|x_{1:n})$ are used to estimate the ELBO loss in Eq. (6.24). The loss function gradient is estimated using SGVB [88]. The model is optimized using the ADAM optimizer with weight decay regularization (same parameters as above). We use an initial learning rate of $5 \times 10^{-3}$ that decays exponentially by a factor of 0.5 as training progresses.

To compute the marginal probability $\mathrm{P}(x_{1:n})$ from our models, we use the inference network to generate $2^{14} = 16\,384$ samples $\tilde{s}_{1:n}$ for each sequence $x_{1:n}$. From these samples $\tilde{s}_{1:n}$, the marginal probability is estimated using Eq. (6.23). We monitor the convergence of the variational marginalization procedure by computing the effective sample size from the importance weights [105]. In our case, the effective sample size always remained above 85% of the actual sample size, indicating that the inference network approximates the posterior well.

### 6.2.3 Comparison Against the True Mutual Information

The green dots in Fig. 6.3 display the ML-PWS estimate of the mutual information $I(S_{1:50}, X_{1:50})$ as a function of the gain $\gamma$, see Eq. (6.27). As expected, for small $\gamma$, the mutual information grows with $\gamma$ as the gain enhances the signal-to-noise ratio. For larger values of $\gamma$, we observe a saturation and even a decline in the information rate due to the saturation effect of the logistic function. This behavior is indicative

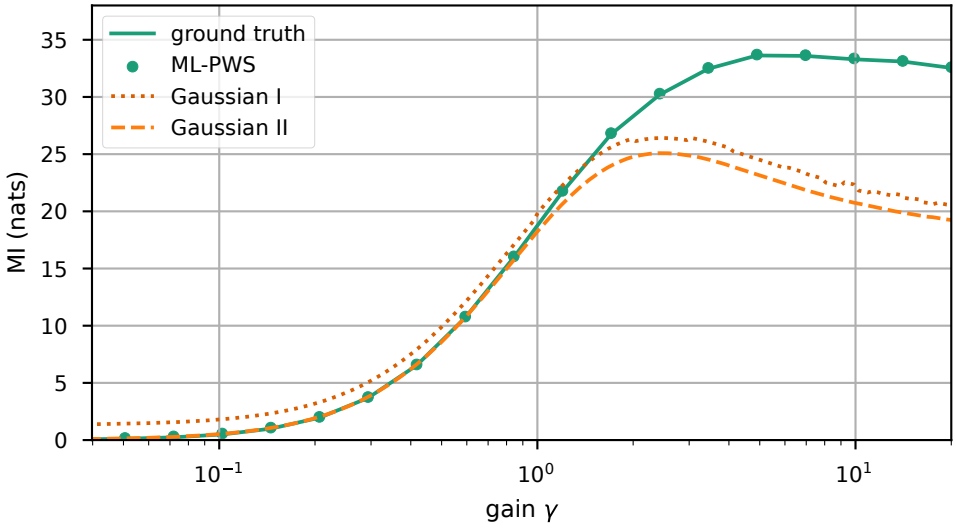



Figure 6.3: Mutual information estimates for the nonlinear 1D time-series model across a range of input gain values (see Eq. (6.27)). The green dots represent the ML-PWS estimates (using an autoregressive RNN model for learning the stochastic dynamics), while the solid green line indicates the ground truth mutual information calculated by applying PWS directly to the nonlinear model. For comparison, the Gaussian approximation has been obtained in two diferent ways: the dotted orange line (labeled *Gaussian I*) represents Gaussian information estimates obtained from the same dataset ($N = 1000$) that was used to train the machine learning model, showing finite sample size effects. The dashed orange line (labeled *Gaussian II*) represents a "reduced-bias" Gaussian approximation using an extended dataset ($N = 100\,000$). For low gain ($\gamma \lesssim 10$), both Gaussian approximations align closely with the ground truth and ML-PWS. At high gain, however, the Gaussian approximation fails to capture nonlinear effects, and only provides a lower bound on the mutual information. Yet, ML-PWS does not suffer from this problem and correctly estimates the mutual information for the whole range of $\gamma$.

of the nonlinearity of the system.

Next, we compared the mutual information estimates against various benchmarks. First, we compute the true mutual information of the nonlinear system using PWS directly with the model given in Eq. (6.27). Since this model represents the true underlying dynamics of the training data, we consider this result as the "ground truth" mutual information.

Figure 6.3 (solid green line) shows the ground truth mutual information. We can see that the machine learning approach matches the ground truth very well across all values of $\gamma$. This demonstrates that the autoregressive neural network can accurately learn the stochastic dynamics of the nonlinear model and reliably estimate the path likelihood, which is required for the Monte Carlo estimate of mutual information. These results confirm that combining PWS with machine learning is a feasible and promising approach for computing the mutual information rate in complex nonlinear systems.

### 6.2.4 Comparison Against the Gaussian Approximation

Second, we use the Gaussian approximation as a benchmark, which is widely used for directly estimating mutual information rates from time-series data. We obtain the Gaussian estimate by computing the $2n \times 2n$ covariance matrix

$$\Sigma = \begin{pmatrix} \Sigma_{ss} & \Sigma_{sx} \\ \Sigma_{xs} & \Sigma_{xx} \end{pmatrix} \tag{6.29}$$

from our dataset of trajectories $s_{1:n}, x_{1:n}$. The individual blocks of $\Sigma$ are $n \times n$ matrices defined by $\Sigma^{ij}_{\alpha\beta} = \langle \alpha_i \beta_j \rangle$. The Gaussian approximation for the mutual information (in nats) is given by

$$I(S_{1:n}, X_{1:n}) = \frac{1}{2} \ln \frac{|\Sigma_{ss}||\Sigma_{xx}|}{|\Sigma|} . \tag{6.30}$$

See Appendix A for more details.

To make a fair comparison of our machine learning method with the Gaussian approximation, we use the same dataset for obtaining the Gaussian approximation that was used for training the ML model. We refer to this benchmark as "Gaussian I" and it corresponds to the dotted orange line in Fig. 6.3. The Gaussian approximation suffers from two sources of systematic bias: a finite sample size bias due to imperfect correlation function estimates from 1000 trajectory pairs, and a bias arising from the assumption of linearity which does not hold at large $\gamma$. Our aim was to distinguish these two sources of bias.

We created another benchmark, called "Gaussian II" (dashed orange line in Fig. 6.3) to be able to quantify the bias introduced by the small sample size of the "Gaussian I"

benchmark. Gaussian II is similar to Gaussian I but is obtained from a significantly larger dataset of 100 000 trajectory pairs $(s_{1:50}, x_{1:50})$, generated from the nonlinear model. This larger dataset allows for very precise estimates of the (cross-)covariance matrices required for the Gaussian approximation, effectively eliminating the sample size bias. However, it should be noted that this benchmark is "unfair" since it uses a much larger dataset than the one that was used to train the autoregressive ML model.

In Fig. 6.3 we compare the results obtained using PWS against the two variants of the Gaussian approximation, and we observe that for $\gamma \lesssim 10$ the Gaussian approximations (the regular one, Gaussian I, and the one with reduced bias, Gaussian II) closely match the ground truth while significantly deviations appear at larger gain. Indeed, in the low-gain regime, the nonlinearity of the system does not significantly impact the output, the Gaussian approximation provides a reliable estimate of the information. In the high-gain regime, the Gaussian approximation fails to correctly capture the nonlinear dynamics of the system, and only yields a lower bound for the mutual information, as expected from information theory [115].

Figure 6.3 also clearly displays the effect of finite sample size on the accuracy of the Gaussian approximation. Specifically, the Gaussian approximation obtained from the smaller training dataset (Gaussian I, displayed as dotted orange line in Fig. 6.3) consistently overestimates the mutual information as computed with the Gaussian approximation from the larger dataset (Gaussian II, dashed orange line in Fig. 6.3). This means that at low gain, even though the system is approximately linear, the Gaussian approximation obtained from the training set slightly overestimates the true mutual information. This over-estimation is purely an artifact of finite sample size bias, and is not present in the "reduced-bias" Gaussian approximation. Strikingly thus, our new method based on machine learning yields a better estimate for the mutual information using the training set, even at low gain, where the Gaussian approximation is expected to hold.

## 6.3 Discussion

By combining neural networks with PWS, we introduced ML-PWS, a new scheme to estimate the mutual information between input and output trajectories of a system. PWS is a Monte Carlo method for calculating mutual information between trajectories, relying on a stochastic model that defines the probability distribution of trajectories to determine the path action. We demonstrated how autoregressive sequence prediction models can be trained on time-series data to learn this stochastic model, making it possible to use PWS with such models to compute mutual information. By applying ML-PWS to a nonlinear model of information processing, we showed

that it provides more accurate mutual information estimates than the Gaussian approximation. While this example serves as a proof of concept, it shows the potential of advanced machine learning techniques to automatically derive stochastic models from experimental data, and to make it possible to compute information-theoretic measures for complex, high-dimensional data.

Using the mutual information rate as a measure for time series correlation possesses a distinct advantage compared to other, simpler, measures: it remains invariant under deterministic and lossless transformations of the sequences [30]. Not only is this property desirable on philosophical grounds, but it can also simplify the training of machine learning models. Specifically, in some cases, it could be beneficial to preprocess the time series data by transforming it into a different representation (e.g. a symbolic encoding) that is more conducive to machine learning analysis [97]. Such a transformation could be applied before employing ML-PWS to compute the mutual information rate. Additionally, this concept could be used for model reduction, making it possible to answer the question of whether a time series with a simplified representation still maintains the same mutual information rate.

As our results in Fig. 6.3 demonstrate, ML-PWS with Gaussian autoregressive models is significantly more general than the Gaussian framework [194]. The range of stochastic processes representable by neural autoregressive models is much larger than the range of processes which can be described by a Gaussian model. Even though in the autoregressive model the distribution of each $x_i$ conditional on $x_{1:i-1}$ and $s_{1:i}$ is Gaussian, the distribution of the whole sequence $\mathrm{P}(x_{1:n}|s_{1:n})$ is generally not Gaussian due to the nonlinearity of the neural network. In fact, a Gaussian autoregressive model can be seen as a (time-discretized) representation of a nonlinear stochastic differential equation of the form

$$dx_t = f(s_{1:t}, x_{1:t-1})\,dt + g(s_{1:t}, x_{1:t-1})\,dW_t \tag{6.31}$$

where $f(s_{1:t}, x_{1:t-1})$ and $g(s_{1:t}, x_{1:t-1})$ are the potentially nonlinear drift and diffusion terms that are learned by the neural network. This representation is a generalization of the Gaussian framework, which assumes $f$ and $g$ to be linear operators.

Nonetheless, assuming that each $x_i$ is normally distributed given its and the signal's history, does restrict the expressive power of the model. For instance, such a model can never represent multimodal predictive distributions or discrete state spaces. In these cases, other predictive distributions must be chosen.

ML-PWS shares some characteristics with the Difference-of-Entropies (DoE) estimator for the mutual information proposed by McAllester and Stratos [108]. Let $\boldsymbol{S}$ and $\boldsymbol{X}$ be (high-dimensional) random variables and $I(\boldsymbol{S},\boldsymbol{X}) = H(\boldsymbol{X}) - H(\boldsymbol{X}|\boldsymbol{S})$ the desired mutual information computed as a difference of entropies. In both ML-PWS

and the DoE method, the generative model for the conditional probability $\mathrm{P}(\boldsymbol{x}|\boldsymbol{s})$ is trained by maximizing the likelihood, an objective that, as shown in Ref. [108], results in an upper bound of the conditional entropy $H(\boldsymbol{X}|\boldsymbol{S})$. However, unlike ML-PWS, McAllester and Stratos [108] use a second generative model that represents the marginal probability $\mathrm{P}(\boldsymbol{x})$, yielding an upper bound of the marginal entropy $H(\boldsymbol{X})$. PWS instead computes the marginal probability $\mathrm{P}(\boldsymbol{x})$ by marginalizing the joint probability $\mathrm{P}(\boldsymbol{s},\boldsymbol{x}) = \mathrm{P}(\boldsymbol{s})\,\mathrm{P}(\boldsymbol{x}|\boldsymbol{s})$. In this way, $\mathrm{P}(\boldsymbol{x})$ can in principle be computed for any desired accuracy. This direct marginalization in ML-PWS also enables efficient computation of the mutual information $I(\boldsymbol{S}',\boldsymbol{X}')$ between a input signal $\boldsymbol{S}'$ with different statistics and the corresponding output $\boldsymbol{X}'$, without needing to retrain the marginal model. This is particularly useful if one is interested in the channel capacity, as determined by the input distribution that maximizes the mutual information or information rate for the system of interest. Indeed, in systems without feedback, $\mathrm{P}(\boldsymbol{x}|\boldsymbol{s})$ is a property of the system and does not change upon changing the input. The generative model then remains the same, such that ML-PWS can directly recompute the mutual information for different input statistics. Determining which of these approaches is preferable for estimating mutual information remains an open question for future research.

Alongside ML-PWS, we introduced a new marginalization scheme for PWS, leveraging techniques from variational inference. In PWS, marginalization is used to compute the marginal probability of a trajectory, $\mathrm{P}(x_{1:n})$. Our approach trains an inference network to generate input trajectories $s_{1:n}$ approximately distributed according to the posterior distribution $\mathrm{P}(s_{1:n}|x_{1:n})$, enabling efficient calculation of $\mathrm{P}(x_{1:n})$ through importance sampling. Importantly, this marginalization technique is general and can be applied to any generative model, regardless of whether it is based on neural networks. This flexibility makes it a powerful marginalization scheme for any application of PWS to a system with continuous state space. Furthermore, since marginalization is mathematically equivalent to a free-energy computation (see Chapter 3), our approach demonstrates that variational techniques can yield efficient methods for free-energy estimation.

In conclusion, ML-PWS—our integration of neural networks and variational inference into the PWS framework—represents a promising advance for estimating the mutual information of high-dimensional data, a notoriously difficult problem. While our initial tests on a toy example demonstrate the technique's potential, further evaluation on more complex systems is needed. Additionally, comparing ML-PWS with other neural methods for mutual information estimation will help clarify its advantages and limitations. By enabling the accurate computation of the mutual information rate, we anticipate that ML-PWS could contribute to a deeper understanding of complex dynamical systems, with potential applications spanning neuroscience, biology, physics, and machine learning.

# A Notes on the Computation of the Gaussian Approximation

Mathematically, computing the mutual information between trajectories in the Gaussian framework is relatively simple. From Shannon's formula for the entropy of a multivariate Gaussian distribution it follows that the MI is given by

$$I(S,X) = \frac{1}{2} \ln\left(\frac{|\Sigma_{ss}|\,|\Sigma_{xx}|}{|Z|}\right) \quad [\text{nats}]\,. \tag{A.1}$$

See Tostevin and ten Wolde [194, 195] for details. This is a straightforward formula involving the logarithm of determinants of symmetric matrices.

However, computationally evaluating this formula correctly and efficiently requires some thought. The matrices $\Sigma_{ss}$ and $\Sigma_{xx}$ have dimensions $N \times N$, and the matrix $Z$ has dimensions $2N \times 2N$ where $N$ is the trajectory length. Thus, for long trajectories, the involved matrices can become very large. Computing the determinant of a large matrix is computationally non-trivial and possibly numerically unstable. Note that the generic algorithm to compute the determinant of a matrix of dimensions $N \times N$ scales with order $N^3$. Thus, doubling the size of the matrix makes the computation of the determinant take approximately 8 times as long. Moreover, the determinant of a large matrix may be very close to zero or very close to infinity. Representing such numbers in the computer using floating point representations can lead to significant numerical accuracy issues. In practice this further limits the maximal size of covariance matrices that one can use.

Fortunately, we can leverage the special structure of the Gaussian covariance matrices to simplify the computation of determinants significantly. Generally, all involved matrices are symmetric and positive definite. Using a clever trick we can speed up the computation of the determinant of a symmetric matrix. But even then, we still have a scaling of $n^3$ with matrix size. Using one additional assumption we can do much better. If the system under consideration is in steady state, the matrices have Toeplitz structure. We will see that this structure allows us to construct a $n \log n$ scaling algorithm that approximates the determinant very well for large $n$. Lastly, to deal with numerical accuracy issues, we will discuss how to compute the log-determinant of a matrix directly. This will be dramatically more accurate than first computing the determinant and then taking the logarithm.

### Structure of Covariance Matrices in the Gaussian Framework

We consider a Gaussian system with stochastic input $\boldsymbol{s}$ and output $\boldsymbol{x}$. Both, input and output, are vectors representing trajectories $s(t)$ and $x(t)$ sampled at discrete times $t_0, \ldots, t_{d-1}$. Hence, $d \in \mathbb{N}$ is the dimensionality of the vector space of $\boldsymbol{s}$ and $\boldsymbol{x}$. The joint probability density of $\boldsymbol{s}$ and $\boldsymbol{x}$ is given by

$$\mathrm{P}(\boldsymbol{s}, \boldsymbol{x}) = \frac{1}{(2\pi|Z|)^d} \exp\left[-\frac{1}{2}\begin{pmatrix} \boldsymbol{s} & \boldsymbol{x} \end{pmatrix} Z^{-1} \begin{pmatrix} \boldsymbol{s} \\ \boldsymbol{x} \end{pmatrix}\right] \tag{A.2}$$

which is the PDF of a $2d$-dimensional multivariate Gaussian distribution. $Z$ is a $2d \times 2d$ matrix with block form

$$Z = \begin{pmatrix} \Sigma_{ss} & \Sigma_{sx} \\ \Sigma_{xs} & \Sigma_{xx} \end{pmatrix}. \tag{A.3}$$

The individual block entries of $Z$ correspond to the (cross-)covariance matrices of input and output. Specifically, $\Sigma_{\alpha\beta}$ is a $d \times d$ matrix with entries

$$\Sigma_{\alpha\beta}^{ij} = \langle \alpha(t_i)\beta(t_j) \rangle . \tag{A.4}$$

We can make a few observations concerning the form of these matrices. It follows immediately from the definition that both $\Sigma_{ss}$ and $\Sigma_{xx}$ are positive definite, symmetric matrices. $\Sigma_{sx}$ and $\Sigma_{xs}$ are generally not symmetric, but they are transposes of each other, i.e., $\Sigma_{sx}^T = \Sigma_{xs}$. It follows from these two observations that $Z$ itself is symmetric. Furthermore, we usually deal with systems in steady state, in which case the matrices have even more structure. When a system is in steady state, the correlation functions are (time-)translation invariant. That means that the correlation of observations at $t_i$ and $t_j$ only depends on the difference $t_j - t_i$, i.e., we can find functions $C_{\alpha\beta}(t)$ such that $C_{\alpha\beta}(t_j - t_i) = \langle \alpha(t_i)\beta(t_j) \rangle$. It follows that the matrix element $\Sigma_{\alpha\beta}^{ij}$ only depends on the difference $j - i$. We can thus write the matrix elements as $\Sigma_{\alpha\beta}^{ij} = c_{\alpha\beta}^{j-i}$ where $c_{\alpha\beta}^{i} = C_{\alpha\beta}(t_i - t_0)$ are the components of a vector $\boldsymbol{c}$ that fully specifies the matrix. A matrix with this structure is called *Toeplitz* matrix. We can use this structure to simplify the computation of determinants significantly, as described below. But before we explain how to compute determinants efficiently, we discuss one further issue regarding the matrix $Z$ itself.

Note that unlike the matrices $\Sigma_{\alpha\beta}$, $Z$ itself does not have Toeplitz structure in general. This means the method described below is not directly applicable for the computation of $|Z|$. Yet, it seems we need to find $|Z|$ for computing the MI using Eq. (A.1). It turns out that this is not necessary. One can make use of the fact that $Z$ is composed of Toeplitz blocks to avoid computing its determinant. At the end of

the following section we show how to adapt the formula described in Ref. [123] for computing the mutual information rate to derive an efficient approximation of it in terms of the discrete Fourier transform.

### Efficient Evaluation of Determinants

For a symmetric positive definite matrix $\Sigma \in \mathbb{R}^{d\times d}$ the determinant can be computed faster than for a general matrix. We exploit the fact that such a matrix can always be decomposed as

$$\Sigma = LL^T \tag{A.5}$$

where $L \in \mathbb{R}^{d\times d}$ is a lower triangular matrix with positive diagonal entries. This factorization is called *Cholesky decomposition* and there are efficient algorithms for computing the Cholesky decomposition in many numerical linear algebra packages.

Now we make use of two basic properties of the determinant. First, the determinant of a product of matrices is equal to the product of the individual matrices' determinants. Second, the determinant of the transpose is equal to the determinant of the original matrix. Combining these two facts, we see that $|\Sigma| = |L||L^T| = |L|^2$. Now, we remember that $L$ is a lower triangular matrix. It is easily checked that the determinant of a lower triangular matrix is given by the product of its diagonal entries $\ell_0, \ldots, \ell_{d-1}$ where $\ell_k = L^{kk}$. Thus, we find two important identities for symmetric positive definite matrices:

$$|\Sigma| = \prod_{i=0}^{d-1} \ell_i^2 \,, \tag{A.6}$$

$$\ln|\Sigma| = 2\sum_{i=0}^{d-1} \ln \ell_i \,. \tag{A.7}$$

The second form is especially useful for large matrices. There, computing the log-determinant is numerically preferable to computing the determinant itself.

We can also exploit the structure of symmetric Toeplitz matrices to further simplify the computation of the determinant. However, here we won't obtain an exact result but only an asymptotic approximation. This will nonetheless be very useful for large matrices where even the Cholesky factorization can be difficult to obtain.

We will investigate the asymptotic behavior of eigenvalues of Toeplitz matrices. First, we recall the defining property of Toeplitz matrices. Let $T \in \mathbb{R}^{d\times d}$ be a Toeplitz matrix. Then we can write $T^{ij} = t^{|j-i|}$ where $\boldsymbol{t} \in \mathbb{R}^d$ is a vector that fully specifies the matrix. We will now try to understand what happens in the asymptotic limit that the vector $\boldsymbol{t}$ is very high dimensional. Thus, suppose we have an infinite sequence of real numbers $(t_k \mid k = \ldots, -2, -1, 0, 1, 2, \ldots)$. From this sequence we

can construct a sequence of Toeplitz matrices $T_n \in \mathbb{R}^{n\times n}$ for $n = 1, 2, \ldots$ as follows. For any $n \in \mathbb{N}$, we define the matrix elements of $T_n$ as

$$T_n^{ij} = t_{i-j} \quad \text{for } i, j = 0, 1, \ldots, n-1\,. \tag{A.8}$$

The definition of $T_n$ will thus only need $2n-1$ elements $t_{-(n-1)}, \ldots, t_0, \ldots, t_{n-1}$ from the sequence. Note that for a symmetric Toeplitz sequence $t_k = t_{-k}$.

Under some light conditions there exists a continuous $2\pi$-periodic function $f(\lambda)$ that is defined through the discrete-time Fourier transform

$$f(\omega) = \sum_{k=-\infty}^{\infty} t_k e^{-i\omega k}\,. \tag{A.9}$$

Conversely, the $t_k$ can be recovered from $f$ using inverse transform

$$t_k = \frac{1}{2\pi}\int_0^{2\pi} d\omega\, f(\omega) e^{i\omega k}\,. \tag{A.10}$$

Thus the sequence $(t_k)$ determines the function $f$ and vice versa. Moreover, this means that the entire sequence of Toeplitz matrices is completely specified by the continuous function $f$.

Now we are ready to state one of the most important theorems about Toeplitz matrices: *Szegő's theorem*. Let $\tau_{n,1}, \ldots, \tau_{n,n}$ be the eigenvalues of the matrix $T_n$. Szegő's theorem states that

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n-1} F(\tau_{n,i}) = \frac{1}{2\pi}\int_0^{2\pi} d\omega\, F(f(\omega)) \tag{A.11}$$

for any continuous function $F$. This result is very useful to find the asymptotic determinant. In particular, note that the determinant is the product of eigenvalues, and hence $\ln|T_n| = \sum_{i=1}^{n-1} \ln \tau_{n,i}$. Thus, it is not hard to show that

$$\lim_{n\to\infty} \ln \frac{|T_n|}{|T_{n-1}|} = \frac{1}{2\pi}\int_0^{2\pi} d\omega\, \ln f(\omega) \tag{A.12}$$

which we can leverage to compute information rates. In particular, this formula can be used for analytical computation of entropy rates of Gaussian processes [194].

We will instead use this theorem to find a an approximate way to compute the determinant of a large Toeplitz matrix. We can approximate the integral by a Riemann sum

$$\lim_{n\to\infty} \frac{\ln|T_n|}{n} \approx \frac{1}{N}\sum_{m=0}^{N-1} \ln f\left(\omega_m\right) \tag{A.13}$$

where $\omega_m = 2\pi m/N$ for some large $N$. Now, inserting Eq. (A.9) we find

$$\lim_{n\to\infty} \frac{\ln|T_n|}{n} \approx \frac{1}{N}\sum_{m=0}^{N-1} \ln \sum_{k=-\infty}^{\infty} t_k \exp\left(-\frac{i2\pi km}{N}\right). \tag{A.14}$$

In practice we cannot perform the infinite sum over $k$. Thus, we must truncate the series. We truncate the series such that $t_k = 0$ for $|k| > N/2$. Then, the sum over $k$ becomes the discrete Fourier transform (DFT). We recall that the DFT of the sequence $t_k$ for $k = -\lfloor N/2\rfloor, \dots, 0, \dots, \lfloor N/2\rfloor$ is another sequence $(\lambda_m)$, defined by

$$\lambda_m = \sum_{k=-\lfloor N/2\rfloor}^{\lfloor N/2\rfloor} t_k \exp\left(-\frac{i2\pi km}{N}\right) \tag{A.15}$$

for $m = 0, \dots, N-1$. Hence, we have derived the approximation

$$\lim_{n\to\infty} \frac{\ln|T_n|}{n} \approx \frac{1}{N}\sum_{m=0}^{N-1} \ln\lambda_m\,. \tag{A.16}$$

The DFT coefficients $\lambda_m$ represent the inner sum in Eq. (A.14), when truncated to $k = -\lfloor N/2\rfloor, \dots, 0, \dots, \lfloor N/2\rfloor$. In principle we want to choose $N$ as large as possible. However, given a matrix $T_n$ we only know the values $t_{-n}, \dots, t_0, \dots, t_n$. Thus, the maximum value of $N$ that we can use is $N = 2n$.

This means, our approximation for the log-determinant of $T_n$ for large $n$ can be written as

$$\ln|T_n| \approx \frac{1}{2}\sum_{m=0}^{2n-1} \ln\lambda_m\,. \tag{A.17}$$

It is easy to verify that this formula converges to Eq. (A.14) as $n \to \infty$. This formula is also very efficient to evaluate on modern computers. The sequence $\lambda_0, \dots, \lambda_{2n-1}$ can be easily computed via the FFT algorithm from the sequence of $t_k$. Efficient implementations of the FFT algorithm are widely available.

Finally, we discuss how to evaluate the mutual information *rate*, defined as $R(S,X) = \lim_{n\to\infty} I(S[t_0 : t_n], X[t_0 : t_n])/n$, using Szegő's theorem. Munakata and Kamiyabu [123] used Szegő's theorem to derive the formula

$$R(S,X) = -\frac{1}{2}\int_0^{2\pi} d\omega\ \ln\left(1 - \frac{|f_{sx}(\omega)|^2}{f_{ss}(\omega)f_{xx}(\omega)}\right) \tag{A.18}$$

where the functions $f_{\alpha\beta}(\omega) = \sum_{k=-\infty}^{\infty} t_{\alpha\beta,k}e^{-i\omega k}$ represent the discrete-time Fourier transforms of the covariances of the matrices $\Sigma_{\alpha\beta}$ in the limit $n \to \infty$. As above, we

are going to approximate this formula using the discrete Fourier transform. We define DFT sequences for the matrices $\Sigma_{\alpha\beta}$ as $\lambda_{\alpha\beta,m} = \sum_{k=-n}^{n} t_{\alpha\beta,m} \exp(-i\omega_m k)$ for $\omega_m = 2\pi m/N$. This results in the estimate for the mutual information rate

$$R(S,X) \approx -\frac{1}{2} \sum_{m=0}^{2n-1} \ln\left(1 - \frac{|\lambda_{sx,m}|^2}{\lambda_{ss,m}\lambda_{xx,m}}\right). \tag{A.19}$$

More information about asymptotics and other results for Toeplitz matrices can be found in the review of Gray [71].

# References


[1] C. Adami. The use of information theory in evolutionary biology. *Annals of the New York Academy of Sciences*, 1256(1):49–65, 2012. ISSN 0077-8923. doi:10.1111/j.1749-6632.2011.06422.x.

[2] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy. Deep Variational Information Bottleneck. *arXiv*, 2016. doi:10.48550/arxiv.1612.00410.

[3] R. J. Allen, D. Frenkel, and P. R. ten Wolde. Simulating rare events in equilibrium or nonequilibrium stochastic systems. *The Journal of Chemical Physics*, 124(2):024102, 2006. ISSN 0021-9606. doi:10.1063/1.2140273.

[4] C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010. ISSN 1467-9868. doi:10.1111/j.1467-9868.2009.00736.x.

[5] M. Aspelmeyer, T. J. Kippenberg, and F. Marquardt. Cavity optomechanics. *Reviews of Modern Physics*, 86(4):1391–1452, 2014. ISSN 0034-6861. doi:10.1103/revmodphys.86.1391.

[6] A. C. Barato, D. Hartich, and U. Seifert. Rate of Mutual Information Between Coarse-Grained Non-Markovian Variables. *Journal of Statistical Physics*, 153 (3):460–478, 2013. ISSN 0022-4715. doi:10.1007/s10955-013-0834-5.

[7] D. Barber and F. Agakov. The IM Algorithm : A variational approach to Information Maximization. In *Advances in Neural Information Processing Systems 16*, pages 201–208, Cambridge, Massachusetts, 2004. MIT Press. ISBN 0262201526.

[8] N. Barkai and S. Leibler. Robustness in simple biochemical networks. *Nature*, 387(6636):913–917, 1997. ISSN 0028-0836. doi:10.1038/43199.

[9] A. Baronchelli, E. Caglioti, and V. Loreto. Measuring complexity with zippers. *European Journal of Physics*, 26(5):S69–S77, 2005. ISSN 0143-0807. doi:10.1088/0143-0807/26/5/s08.

[10] M. Bauer, M. D. Petkova, T. Gregor, E. F. Wieschaus, and W. Bialek. Trading bits in the readout from a genetic network. *Proceedings of the National Academy of Sciences*, 118(46):e2109011118, 2021.

[11] J. Bayer and C. Osendorfer. Learning Stochastic Recurrent Networks. *arXiv*, 2014. doi:10.48550/arxiv.1411.7610.

[12] N. B. Becker, R. J. Allen, and P. R. ten Wolde. Non-stationary forward flux sampling. *The Journal of Chemical Physics*, 136(17):174118, 2012. ISSN 0021-9606. doi:10.1063/1.4704810.

[13] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm. Mutual information neural estimation. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 531–540. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/belghazi18a.html.

[14] C. H. Bennett. Efficient estimation of free energy differences from Monte Carlo data. *Journal of Computational Physics*, 22(2):245–268, 1976. ISSN 0021-9991. doi:10.1016/0021-9991(76)90078-4.

[15] H. Berg and E. Purcell. Physics of chemoreception. *Biophysical Journal*, 20 (2):193–219, 1977. ISSN 0006-3495. doi:10.1016/s0006-3495(77)85544-6.

[16] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59(1):65–98, 2017. ISSN 0036-1445. doi:10.1137/141000671.

[17] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 1 edition, 2006. ISBN 978-0-387-31073-2.

[18] S. M. Block, J. E. Segall, and H. C. Berg. Adaptation kinetics in bacterial chemotaxis. *Journal of Bacteriology*, 154(1):312–323, 1983. ISSN 0021-9193. doi:10.1128/jb.154.1.312-323.1983.

[19] P. G. Bolhuis, D. Chandler, C. Dellago, and P. L. Geissler. TRANSITION PATH SAMPLING: Throwing Ropes Over Rough Mountain Passes, in the Dark. *Annual Review of Physical Chemistry*, 53(1):291–318, 2002. ISSN 0066-426X. doi:10.1146/annurev.physchem.53.082301.113146.

[20] A. Borst and F. E. Theunissen. Information theory and neural coding. *Nature Neuroscience*, 2(11):947–957, 1999. ISSN 1097-6256. doi:10.1038/14731.

[21] L. Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In *Proceedings of COMPSTATS'2010*, pages 177–186, Heidelberg, Germany, 2010. Physica-Verlag HD. ISBN 9783790826036. doi:10.1007/978-3-7908-2604-3_16.

[22] M. J. Boulware and J. S. Marchant. Timing in Cellular Ca2+ Signaling. *Current Biology*, 18(17):R769–R776, 2008. ISSN 0960-9822. doi:10.1016/j.cub.2008.07.018.

[23] C. G. Bowsher and P. S. Swain. Identifying sources of variation and the flow of information in biochemical networks. *Proceedings of the National Academy of Sciences*, 109(20):E1320–E1328, 2012. ISSN 0027-8424. doi:10.1073/pnas.1119407109.

[24] C. Castellano, S. Fortunato, and V. Loreto. Statistical physics of social dynamics. *Reviews of Modern Physics*, 81(2):591–646, 2009. ISSN 0034-6861. doi:10.1103/revmodphys.81.591.

[25] S. A. Cepeda-Humerez, J. Ruess, and G. Tkačik. Estimating information in time-varying signals. *PLoS computational biology*, 15(9):e1007290, 2019. ISSN 1553-734X. doi:10.1371/journal.pcbi.1007290.

[26] M. Chalk, O. Marre, and G. Tkačik. Toward a unified theory of efficient, predictive, and sparse coding. *Proceedings of the National Academy of Sciences*, 115(1):186–191, 2018.

[27] R. Cheong, A. Rhee, C. J. Wang, I. Nemenman, and A. Levchenko. Information Transduction Capacity of Noisy Biochemical Signaling Networks. *Science*, 334(6054):354–358, 2011. ISSN 0036-8075. doi:10.1126/science.1204553.

[28] K. Cho, B. v. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv*, 2014. doi:10.48550/arxiv.1406.1078.

[29] D. Clausznitzer, G. Micali, S. Neumann, V. Sourjik, and R. G. Endres. Predicting Chemical Environments of Bacteria from Receptor Signaling. *PLoS Computational Biology*, 10(10):e1003870, 2014. ISSN 1553-734X. doi:10.1371/journal.pcbi.1003870.

[30] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2 edition, 2006. ISBN 978-0-471-24195-9.

[31] M. W. Covert, T. H. Leung, J. E. Gaston, and D. Baltimore. Achieving stability of lipopolysaccharide-induced nf-κb activation. *Science*, 309(5742):1854–1857, 2005.

[32] J. Cremer, A. Melbinger, and E. Frey. Growth dynamics and the evolution of cooperation in microbial populations. *Scientific Reports*, 2(1):281, 2012. doi:10.1038/srep00281.

[33] G. E. Crooks. Path-ensemble averages in systems driven far from equilibrium. *Physical Review E*, 61(3):2361–2366, 2000. ISSN 1539-3755. doi:10.1103/physreve.61.2361.

[34] A. Das and P. R. t. Wolde. Exact computation of Transfer Entropy with Path Weight Sampling. *arXiv*, 2024. doi:10.48550/arxiv.2409.01650.

[35] W. H. de Ronde, F. Tostevin, and P. R. ten Wolde. Effect of feedback on the fidelity of information transmission of time-varying signals. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 82(3):031914, 2010.

[36] P. Del Moral. Nonlinear filtering: Interacting particle resolution. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 325(6):653–658, 1997. ISSN 0764-4442. doi:10.1016/s0764-4442(97)84778-7.

[37] M. Delbrück. Statistical Fluctuations in Autocatalytic Reactions. *The Journal of Chemical Physics*, 8(1):120–124, 1940. ISSN 0021-9606. doi:10.1063/1.1750549.

[38] C. Dellago, P. G. Bolhuis, and D. Chandler. Efficient transition path sampling: Application to Lennard-Jones cluster rearrangements. *The Journal of Chemical Physics*, 108(22):9236–9245, 1998. ISSN 0021-9606. doi:10.1063/1.476378.

[39] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv*, 2018. doi:10.48550/arxiv.1810.04805.

[40] T. Dimpfl and S. Jank. Can Internet Search Queries Help to Predict Stock Market Volatility? *European Financial Management*, 22(2):171–192, 2016. ISSN 1354-7798. doi:10.1111/eufm.12058.

[41] T. Dimpfl and F. J. Peter. Using transfer entropy to measure information flows between financial markets. *Studies in Nonlinear Dynamics and Econometrics*, 17(1):85–102, 2013. ISSN 1081-1826. doi:10.1515/snde-2012-0044.

[42] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. *arXiv*, 2016. doi:10.48550/arxiv.1605.08803.

[43] M. D. Donsker and S. R. S. Varadhan. Asymptotic evaluation of certain markov process expectations for large time. IV. *Communications on Pure and Applied Mathematics*, 36(2):183–212, 1983. ISSN 0010-3640. doi:10.1002/cpa.3160360204.

[44] A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later. In D. Crisan and B. Rozovskii, editors, *Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011. ISBN 9780199532902.

[45] J. O. Dubuis, G. Tkačik, E. F. Wieschaus, T. Gregor, and W. Bialek. Positional information, in bits. *Proceedings of the National Academy of Sciences*, 110(41): 16301–16308, 2013.

[46] L. Duso and C. Zechner. Path mutual information for a class of biochemical reaction networks. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 6610–6615, 2019. doi:10.1109/CDC40024.2019.9029316.

[47] L. Edelstein-Keshet. *Mathematical Models in Biology*. Society for Industrial and Applied Mathematics, USA, 2005. ISBN 0898715547.

[48] M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, 2000. ISSN 0028-0836. doi:10.1038/35002125.

[49] R. Fan and A. Hilfinger. Characterizing the nonmonotonic behavior of mutual information along biochemical reaction cascades. *Physical Review E*, 110 (3):034309, 2024. ISSN 2470-0045. doi:10.1103/physreve.110.034309.

[50] W. Feller. Die Grundlagen der Volterraschen Theorie des Kampfes ums Dasein in wahrscheinlichkeitstheoretischer Behandlung. *Acta Biotheoretica*, 5 (1):11–40, 1939. ISSN 0001-5342. doi:10.1007/bf01602932.

[51] Y. Feng and Y. Tu. The inverse variance–flatness relation in stochastic gradient descent is critical for finding flat minima. *Proceedings of the National Academy of Sciences*, 118(9):e2015617118, 2021. ISSN 0027-8424. doi:10.1073/pnas.2015617118.

[52] P. Fiedor. Networks in financial markets based on the mutual information rate. *Physical Review E*, 89(5):052801, 2014. ISSN 1539-3755. doi:10.1103/physreve.89.052801.

[53] A. M. Fraser and H. L. Swinney. Independent coordinates for strange attractors from mutual information. *Physical Review A*, 33(2):1134–1140, 1986. ISSN 1050-2947. doi:10.1103/physreva.33.1134.

[54] D. Frenkel and A. J. C. Ladd. New Monte Carlo method to compute the free energy of arbitrary solids. Application to the fcc and hcp phases of hard spheres. *The Journal of Chemical Physics*, 81(7):3188–3193, 1984. ISSN 0021-9606. doi:10.1063/1.448024.

[55] D. Frenkel and B. Smit. *Understanding Molecular Simulation*. Academic Press, San Diego, 2 edition, 2002. ISBN 9780122673511. doi:10.1016/b978-0-12-267351-1.x5000-7.

[56] S. Frenzel and B. Pompe. Partial Mutual Information for Coupling Analysis of Multivariate Time Series. *Physical Review Letters*, 99(20):204101, 2007. ISSN 0031-9007. doi:10.1103/physrevlett.99.204101.

[57] B. J. Frey. *Graphical Models for Machine Learning and Digital Communication*. Adaptive Computation and Machine Learning. The MIT Press, 1998. ISBN 9780262062022.

[58] S. Gao, G. V. Steeg, and A. Galstyan. Efficient Estimation of Mutual Information for Strongly Dependent Variables. *arXiv*, 2014. doi:10.48550/arxiv.1411.2003.

[59] Y. Gao, I. Kontoyiannis, and E. Bienenstock. Estimating the Entropy of Binary Time Series: Methodology, Some Theory and a Simulation Study. *Entropy*, 10 (2):71–99, 2008. doi:10.3390/entropy-e10020071.

[60] C. Gardiner. *Handbook of Stochastic Methods*. Number 13 in Springer Series in Synergetics. Springer-Verlag, Berlin Heidelberg, 4 edition, 2009. ISBN 978-3-540-70712-7.

[61] M. Gehri, N. Engelmann, and H. Koeppl. Mutual Information of a class of Poisson-type Channels using Markov Renewal Theory. *arXiv*, 2024. doi:10.48550/arxiv.2403.15221.

[62] A. Gelman and X.-L. Meng. Simulating normalizing constants: from importance sampling to bridge sampling to path sampling. *Statistical Science*, 13 (2):163–185, 1998. ISSN 0883-4237. doi:10.1214/ss/1028905934.

[63] J. Geweke. Bayesian Inference in Econometric Models Using Monte Carlo Integration. *Econometrica*, 1989.

[64] C. S. Gillespie and A. Golightly. Guided proposals for efficient weighted stochastic simulation. *The Journal of Chemical Physics*, 150(22):224103, 2019. ISSN 0021-9606. doi:10.1063/1.5090979.

[65] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976. ISSN 0021-9991. doi:10.1016/0021-9991(76)90041-3.

[66] A. Golightly and D. J. Wilkinson. Bayesian inference for Markov jump processes with informative observations. *Statistical Applications in Genetics and Molecular Biology*, 14(2):169–188, 2015. ISSN 2194-6302. doi:10.1515/sagmb-2014-0070.

[67] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F Radar and Signal Processing*, 140(2):107, 1993. ISSN 0956-375X. doi:10.1049/ip-f-2.1993.0015.

[68] A. A. Granados, J. M. J. Pietsch, S. A. Cepeda-Humerez, I. L. Farquhar, G. Tkačik, and P. S. Swain. Distributed and dynamic intracellular organization of extracellular information. *Proceedings of the National Academy of Sciences*, 115(23):201716659, 2018. ISSN 0027-8424. doi:10.1073/pnas.1716659115.

[69] P. Grassberger. Pruned-enriched Rosenbluth method: Simulations of $\theta$ polymers of chain length up to 1 000 000. *Physical Review E*, 56(3):3682–3693, 1997. ISSN 1539-3755. doi:10.1103/physreve.56.3682.

[70] A. Graves. Generating Sequences With Recurrent Neural Networks. *arXiv*, 2013. doi:10.48550/arxiv.1308.0850.

[71] R. M. Gray. Toeplitz and Circulant Matrices: A Review. *Foundations and Trends® in Communications and Information Theory*, 2(3):155–239, 2005. ISSN 1567-2190. doi:10.1561/0100000006.

[72] L. Hahn, A. M. Walczak, and T. Mora. Dynamical Information Synergy in Biochemical Signaling Networks. *Physical Review Letters*, 131(12):128401, 2023. ISSN 0031-9007. doi:10.1103/physrevlett.131.128401.

[73] D. Hartich, A. C. Barato, and U. Seifert. Stochastic thermodynamics of bipartite systems: transfer entropy inequalities and a Maxwell's demon interpretation. *Journal of Statistical Mechanics: Theory and Experiment*, 2014(2): P02016, 2014. doi:10.1088/1742-5468/2014/02/p02016.

[74] D. Helbing. Traffic and related self-driven many-particle systems. *Reviews of Modern Physics*, 73(4):1067–1141, 2001. ISSN 0034-6861. doi:10.1103/revmodphys.73.1067.

[75] K. Hlaváčková-Schindler, M. Paluš, M. Vejmelka, and J. Bhattacharya. Causality detection based on information-theoretic approaches in time series analysis. *Physics Reports*, 441(1):1–46, 2007. ISSN 0370-1573. doi:10.1016/j.physrep.2006.12.004.

[76] M. Hledík, T. R. Sokolowski, and G. Tkačik. A Tight Upper Bound on Mutual Information. In *2019 IEEE Information Theory Workshop (ITW)*, pages 1–5, 2019. ISBN 978-1-5386-6900-6. doi:10.1109/itw44776.2019.8989292.

[77] M. Hledík, N. Barton, and G. Tkačik. Accumulation and maintenance of information in evolution. *Proceedings of the National Academy of Sciences*, 119(36):e2123152119, 2022. ISSN 0027-8424. doi:10.1073/pnas.2123152119.

[78] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel. Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design. *arXiv*, 2019. doi:10.48550/arxiv.1902.00275.

[79] A. Hobolth and E. A. Stone. Simulation from endpoint-conditioned, continuous-time Markov chains on a finite state space, with applications to molecular evolution. *The Annals of Applied Statistics*, 3(3):1204–1231, 2009. ISSN 1932-6157. doi:10.1214/09-aoas247.

[80] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997. ISSN 0899-7667. doi:10.1162/neco.1997.9.8.1735.

[81] A. Kaiser and T. Schreiber. Information transfer in continuous processes. *Physica D: Nonlinear Phenomena*, 166(1-2):43–62, 2002. ISSN 0167-2789. doi:10.1016/s0167-2789(02)00432-3.

[82] K. Kamino, J. M. Keegstra, J. Long, T. Emonet, and T. S. Shimizu. Adaptive tuning of cell sensory diversity without changes in gene expression. *Science Advances*, 6(46):eabc1087, 2020. doi:10.1126/sciadv.abc1087.

[83] J. M. Keegstra, F. Avgidis, Y. Mulla, J. S. Parkinson, and T. S. Shimizu. Near-critical tuning of cooperativity revealed by spontaneous switching in a protein signalling array. *bioRxiv*, page 2022.12.04.518992, 2023. doi:10.1101/2022.12.04.518992.

[84] J. L. Kelly. A New Interpretation of Information Rate. *Bell System Technical Journal*, 35(4):917–926, 1956. ISSN 0005-8580. doi:10.1002/j.1538-7305.1956.tb03809.x.

[85] K.-Y. Kim and J. Wang. Potential Energy Landscape and Robustness of a Gene Regulatory Network: Toggle Switch. *PLoS Computational Biology*, 3(3):e60, 2007. ISSN 1553-734X. doi:10.1371/journal.pcbi.0030060.

[86] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv*, 2014. doi:10.48550/arxiv.1412.6980.

[87] D. P. Kingma and P. Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. *arXiv*, 2018. doi:10.48550/arxiv.1807.03039.

[88] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *arXiv*, 2013. doi:10.48550/arxiv.1312.6114.

[89] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improving Variational Inference with Inverse Autoregressive Flow. *arXiv*, 2016. doi:10.48550/arxiv.1606.04934.

[90] J. B. Kinney and G. S. Atwal. Equitability, mutual information, and the maximal information coefficient. *Proceedings of the National Academy of Sciences*, 111(9):3354–3359, 2014. ISSN 0027-8424. doi:10.1073/pnas.1309933111.

[91] P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Springer, Berlin, Heidelberg, 1992. ISBN 9783642081071. doi:10.1007/978-3-662-12616-5.

[92] I. Kobyzev, S. J. Prince, and M. A. Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, 2019. ISSN 0162-8828. doi:10.1109/tpami.2020.2992934.

[93] A. Kolchinsky, B. D. Tracey, and D. H. Wolpert. Nonlinear Information Bottleneck. *Entropy*, 21(12):1181, 2019. doi:10.3390/e21121181.

[94] A. Kraskov, H. Stögbauer, and P. Grassberger. Estimating mutual information. *Physical Review E*, 69(6):066138, 2004. ISSN 1539-3755. doi:10.1103/physreve.69.066138.

[95] M. D. Lazova, T. Ahmed, D. Bellomo, R. Stocker, and T. S. Shimizu. Response rescaling in bacterial chemotaxis. *Proceedings of the National Academy of Sciences*, 108(33):13870–13875, 2011. ISSN 0027-8424. doi:10.1073/pnas.1108608108.

[96] M. Li and G. L. Hazelbauer. Cellular Stoichiometry of the Components of the Chemotaxis Signaling Complex. *Journal of Bacteriology*, 186(12):3687–3694, 2004. ISSN 0021-9193. doi:10.1128/jb.186.12.3687-3694.2004.

[97] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, page 2–11, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 9781450374224. doi:10.1145/882082.882086. URL https://doi.org/10.1145/882082.882086.

[98] I. Loshchilov and F. Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. *arXiv*, 2016. doi:10.48550/arxiv.1608.03983.

[99] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. *arXiv*, 2017. doi:10.48550/arxiv.1711.05101.

[100] T. Lux. Herd Behaviour, Bubbles and Crashes. *The Economic Journal*, 105 (431):881–896, 1995. ISSN 0013-0133. doi:10.2307/2235156.

[101] D. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, Cambridge, UK, 2003. ISBN 978-0521642989.

[102] G. Malaguti and P. R. ten Wolde. Theory for the optimal detection of time-varying signals in cellular sensing systems. *eLife*, 10:e62574, 2021. doi:10.7554/elife.62574.

[103] R. Marschinski and H. Kantz. Analysing the information flow between financial time series. *The European Physical Journal B - Condensed Matter and Complex Systems*, 30(2):275–281, 2002. ISSN 1434-6028. doi:10.1140/epjb/e2002-00379-2.

[104] C. Marshall. Specificity of receptor tyrosine kinase signaling: Transient versus sustained extracellular signal-regulated kinase activation. *Cell*, 80(2):179–185, 1995. ISSN 0092-8674. doi:10.1016/0092-8674(95)90401-8.

[105] L. Martino, V. Elvira, and F. Louzada. Effective sample size for importance sampling based on discrepancy measures. *Signal Processing*, 131:386–401, 2017. ISSN 0165-1684. doi:10.1016/j.sigpro.2016.08.025. URL https://www.sciencedirect.com/science/article/pii/S0165168416302110.

[106] J. L. Massey. Causality, Feedback and Directed Information. In *Proc. 1990 Int. Symp. on Info. Th. & its Applications*, Proc. 1990 Int. Symp. on Info. Th. & its Applications, page 303–305, Hawaii, USA, 11 1990.

[107] H. H. Mattingly, K. Kamino, B. B. Machta, and T. Emonet. Escherichia coli chemotaxis is information limited. *Nature Physics*, 17(12):1426–1431, 2021. ISSN 1745-2473. doi:10.1038/s41567-021-01380-3.

[108] D. McAllester and K. Stratos. Formal Limitations on the Measurement of Mutual Information. *arXiv*, 2018. doi:10.48550/arxiv.1811.04251.

[109] D. A. McQuarrie. Kinetics of Small Systems. I. *The Journal of Chemical Physics*, 38(2):433–436, 1963. ISSN 0021-9606. doi:10.1063/1.1733676.

[110] D. A. McQuarrie, C. J. Jachimowski, and M. E. Russell. Kinetics of Small Systems. II. *The Journal of Chemical Physics*, 40(10):2914–2921, 1964. ISSN 0021-9606. doi:10.1063/1.1724926.

[111] P. Mehta, S. Goyal, T. Long, B. L. Bassler, and N. S. Wingreen. Information processing and signal integration in bacterial quorum sensing. *Molecular Systems Biology*, 5(1):325–325, 2009. ISSN 1744-4292. doi:10.1038/msb.2009.79.

[112] M. Meijers, S. Ito, and P. R. t. Wolde. Behavior of information flow near criticality. *Physical Review E*, 103(1):L010102, 2021. ISSN 2470-0045. doi:10.1103/physreve.103.l010102.

[113] B. A. Mello and Y. Tu. Effects of Adaptation in Maintaining High Sensitivity over a Wide Range of Backgrounds for Escherichia coli Chemotaxis. *Biophysical Journal*, 92(7):2329–2337, 2007. ISSN 0006-3495. doi:10.1529/biophysj.106.097808.

[114] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. ISSN 0021-9606. doi:10.1063/1.1699114.

[115] P. P. Mitra and J. B. Stark. Nonlinear limits to the information capacity of optical fibre communications. *Nature*, 411(6841):1027–1030, 2001.

[116] 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. *Scientific Reports*, 8(1):13230, 2018. doi:10.1038/s41598-018-31626-9.

[117] J. Monod, J. Wyman, and J.-P. Changeux. On the nature of allosteric transitions: A plausible model. *Journal of Molecular Biology*, 12(1):88–118, 1965. ISSN 0022-2836. doi:10.1016/s0022-2836(65)80285-6.

[118] Y.-I. Moon, B. Rajagopalan, and U. Lall. Estimation of mutual information using kernel density estimators. *Physical Review E*, 52(3):2318–2321, 1995. ISSN 1539-3755. doi:10.1103/physreve.52.2318.

[119] A.-L. Moor and C. Zechner. Dynamic information transfer in stochastic biochemical networks. *Physical Review Research*, 5(1):013032, 2023. doi:10.1103/physrevresearch.5.013032.

[120] A.-L. Moor, A. Tjalma, M. Reinhardt, P. R. ten Wolde, and C. Zechner. Reaction-based information processing in biochemical networks. Unpublished manuscript, 2024.

[121] C. J. Morton-Firth, T. S. Shimizu, and D. Bray. A free-energy-based stochastic simulation of the tar receptor complex. *Journal of Molecular Biology*, 286(4): 1059–1074, 1999. ISSN 0022-2836. doi:10.1006/jmbi.1999.2535.

[122] M. Müller and W. Paul. Measuring the chemical potential of polymer solutions and melts in computer simulations. *The Journal of Chemical Physics*, 100(1):719–724, 1994. ISSN 0021-9606. doi:10.1063/1.466937.

[123] T. Munakata and M. Kamiyabu. Stochastic resonance in the FitzHugh-Nagumo model from a dynamic mutual information point of view. *The European Physical Journal B - Condensed Matter and Complex Systems*, 53(2): 239–243, 2006. ISSN 1434-6028. doi:10.1140/epjb/e2006-00363-x.

[124] T. Müller, B. Mcwilliams, F. Rousselle, M. Gross, and J. Novák. Neural Importance Sampling. *ACM Transactions on Graphics (TOG)*, 38(5):1–19, 2019. ISSN 0730-0301. doi:10.1145/3341156.

[125] E. A. Nadaraya. On Estimating Regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964. ISSN 0040-585X. doi:10.1137/1109020.

[126] P. Nałęcz-Jawecki, P. A. Gagliardi, M. Kochańczyk, C. Dessauges, O. Pertz, and T. Lipniacki. The MAPK/ERK channel capacity exceeds 6 bit/hour. *PLOS Computational Biology*, 19(5):e1011155, 2023. ISSN 1553-734X. doi:10.1371/journal.pcbi.1011155.

[127] R. M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2): 125–139, 2001. ISSN 0960-3174. doi:10.1023/a:1008923215028.

[128] J. V. Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, Princeton, N.J., 60th anniversary ed. edition, 2004. ISBN 9780691119939.

[129] X. Nguyen, M. J. Wainwright, and M. I. Jordan. Estimating Divergence Functionals and the Likelihood Ratio by Convex Risk Minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010. ISSN 0018-9448. doi:10.1109/tit.2010.2068870.

[130] G. Nicoletti and D. M. Busiello. Tuning Transduction from Hidden Observables to Optimize Information Harvesting. *Physical Review Letters*, 133(15): 158401, 2024. ISSN 0031-9007. doi:10.1103/physrevlett.133.158401.

[131] L. Onsager and S. Machlup. Fluctuations and Irreversible Processes. *Physical Review*, 91(6):1505–1512, 1953. ISSN 0031-899X. doi:10.1103/physrev.91.1505.

[132] A. v. d. Oord, Y. Li, and O. Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv*, 2018. doi:10.48550/arxiv.1807.03748.

[133] A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing*. Prentice-Hall, 1 1975. ISBN 0-13-214635-5.

[134] D. R. Ortega, C. Yang, P. Ames, J. Baudry, J. S. Parkinson, and I. B. Zhulin. A phenylalanine rotameric switch for signal-state control in bacterial chemoreceptors. *Nature Communications*, 4(1):2881, 2013. doi:10.1038/ncomms3881.

[135] S. E. Palmer, O. Marre, M. J. Berry, and W. Bialek. Predictive information in a sensory population. *Proceedings of the National Academy of Sciences*, 112 (22):6908–6913, 2015. ISSN 0027-8424. doi:10.1073/pnas.1506855112.

[136] M. Paluš. Testing for nonlinearity using redundancies: quantitative and qualitative aspects. *Physica D: Nonlinear Phenomena*, 80(1-2):186–205, 1995. ISSN 0167-2789. doi:10.1016/0167-2789(95)90079-9.

[137] L. Paninski. Estimation of Entropy and Mutual Information. *Neural Computation*, 15(6):1191–1253, 2003. ISSN 0899-7667. doi:10.1162/089976603321780272.

[138] G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/6c1da886822c67822bcf3679d04369fa-Paper.pdf.

[139] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. URL http://jmlr.org/papers/v22/19-1028.html.

[140] J.-M. Park, E. Muñoz, and M. W. Deem. Quasispecies theory for finite populations. *Physical Review E*, 81(1):011902, 2010. ISSN 1539-3755. doi:10.1103/physreve.81.011902.

[141] K. J. H. Peters and S. R. K. Rodriguez. Exceptional Precision of a Nonlinear Optical Sensor at a Square-Root Singularity. *Physical Review Letters*, 129(1): 013901, 2022. ISSN 0031-9007. doi:10.1103/physrevlett.129.013901.

[142] K. R. Pilkiewicz and M. L. Mayo. Fluctuation sensitivity of a transcriptional signaling cascade. *Physical Review E*, 94(3):032412, 2016. ISSN 2470-0045. doi:10.1103/physreve.94.032412.

[143] B. Poole, S. Ozair, A. v. d. Oord, A. A. Alemi, and G. Tucker. On Variational Bounds of Mutual Information. *arXiv*, 2019. doi:10.48550/arxiv.1905.06922.

[144] A. Prados, J. J. Brey, and B. Sánchez-Rey. A dynamical monte carlo algorithm for master equations with time-dependent transition rates. *Journal of Statistical Physics*, 89(3-4):709–734, 1997. ISSN 0022-4715. doi:10.1007/bf02765541.

[145] T. Prellberg and J. Krawczyk. Flat Histogram Version of the Pruned and Enriched Rosenbluth Method. *Physical Review Letters*, 92(12):120602, 2004. ISSN 0031-9007. doi:10.1103/physrevlett.92.120602.

[146] J. Purvis and G. Lahav. Encoding and Decoding Cellular Information through Signaling Dynamics. *Cell*, 152(5):945–956, 2013. ISSN 0092-8674. doi:10.1016/j.cell.2013.02.005.

[147] C. Rackauckas and Q. Nie. DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. *Journal of Open Research Software*, 5(1):15, 5 2017. ISSN 2049-9647. doi:10.5334/jors.151.

[148] K. Rad and L. Paninski. Information Rates and Optimal Decoding in Large Neural Populations. In *Advances in Neural Information Processing Systems*, volume 24, pages 846–854. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper_files/paper/2011/file/8eefcfdf5990e441f0fb6f3fad709e21-Paper.pdf.

[149] V. Rao and Y. W. Teh. Fast MCMC sampling for Markov jump processes and extensions. *Journal of Machine Learning Research*, 14(67):3295–3320, 2013. ISSN 1533-7928. URL http://jmlr.org/papers/v14/rao13a.html.

[150] M. Reinhardt. PathWeightSampling.jl (v0.1.0), Nov. 2021. URL https://doi.org/10.5281/zenodo.6334035. Zenodo.

[151] M. Reinhardt. manuel-rhdt/PathWeightSampling.jl, 2024. URL https://github.com/manuel-rhdt/PathWeightSampling.jl. GitHub.

[152] M. Reinhardt, G. Tkačik, and P. R. ten Wolde. Path Weight Sampling: Exact Monte Carlo Computation of the Mutual Information between Stochastic Trajectories. *Physical Review X*, 13(4):041017, 2023. doi:10.1103/physrevx.13.041017.

[153] D. Rezende and S. Mohamed. Variational Inference with Normalizing Flows. *Proceedings of Machine Learning Research*, 37:1530–1538, 2015. URL https://proceedings.mlr.press/v37/rezende15.html.

[154] F. Rieke, D. Warland, R. de Ruyter van Steveninck, and W. Bialek. *Spikes: exploring the neural code*. MIT Press, Cambridge, Massachusetts, 1999. ISBN 978-0262681087.

[155] M. N. Rosenbluth and A. W. Rosenbluth. Monte Carlo Calculation of the Average Extension of Molecular Chains. *The Journal of Chemical Physics*, 23 (2):356–359, 1955. ISSN 0021-9606. doi:10.1063/1.1741967.

[156] J. Runge, J. Heitzig, V. Petoukhov, and J. Kurths. Escaping the Curse of Dimensionality in Estimating Multivariate Transfer Entropy. *Physical Review Letters*, 108(25):258701, 2012. ISSN 0031-9007. doi:10.1103/physrevlett.108.258701.

[157] V. Sachdeva, T. Mora, A. M. Walczak, and S. E. Palmer. Optimal prediction with resource constraints using the information bottleneck. *PLoS computational biology*, 17(3):e1008743, 2021.

[158] L. J. Savage. *The Foundations of Statistics*. Dover Publications, New York, 2 edition, 1972. ISBN 9780486623498.

[159] M. S. Schmitt, M. Koch-Janusz, M. Fruchart, D. S. Seara, and V. Vitelli. Information theory for model reduction in stochastic dynamical systems. *arXiv*, 2023. doi:10.48550/arxiv.2312.06608.

[160] T. Schreiber. Measuring information transfer. *Physical review letters*, 85(2): 461–4, 2000. ISSN 0031-9007. doi:10.1103/physrevlett.85.461.

[161] W. Schwarting, J. Alonso-Mora, and D. Rus. Planning and Decision-Making for Autonomous Vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):1–24, 2018. ISSN 2573-5144. doi:10.1146/annurev-control-060117-105157.

[162] J. E. Segall, S. M. Block, and H. C. Berg. Temporal comparisons in bacterial chemotaxis. *Proceedings of the National Academy of Sciences*, 83(23):8987–8991, 1986. ISSN 0027-8424. doi:10.1073/pnas.83.23.8987.

[163] J. Selimkhanov, B. Taylor, J. Yao, A. Pilko, J. Albeck, A. Hoffmann, L. Tsimring, and R. Wollman. Accurate information transmission through dynamic biochemical signaling networks. *Science*, 346(6215):1370–1373, 2014. ISSN 0036-8075. doi:10.1126/science.1254933.

[164] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423, 1948. ISSN 0005-8580. doi:10.1002/j.1538-7305.1948.tb01338.x.

[165] T. S. Shimizu, Y. Tu, and H. C. Berg. A modular gradient-sensing network for chemotaxis in Escherichia coli revealed by responses to time-varying stimuli. *Molecular Systems Biology*, 6(1):382, 2010. ISSN 1744-4292. doi:10.1038/msb.2010.37.

[166] R. Shwartz-Ziv and N. Tishby. Opening the Black Box of Deep Neural Networks via Information. *arXiv*, 2017. doi:10.48550/arxiv.1703.00810.

[167] J. I. Siepmann. A method for the direct calculation of chemical potentials for dense chain systems. *Molecular Physics*, 70(6):1145–1158, 1990. ISSN 0026-8976. doi:10.1080/00268979000101591.

[168] J. I. Siepmann and D. Frenkel. Configurational bias Monte Carlo: a new sampling scheme for flexible chains. *Molecular Physics*, 75(1):59–70, 1992. ISSN 0026-8976. doi:10.1080/00268979200100061.

[169] M. Sinzger-D'Angelo and H. Koeppl. Counting Processes with Piecewise-Deterministic Markov Conditional Intensity: Asymptotic Analysis, Implementation and Information-Theoretic Use. *IEEE Transactions on Information Theory*, pages 1–1, 2023. ISSN 0018-9448. doi:10.1109/tit.2023.3293996.

[170] M. Skoge, Y. Meir, and N. S. Wingreen. Dynamics of Cooperativity in Chemical Sensing among Cell-Surface Receptors. *Physical Review Letters*, 107(17): 178101, 2011. ISSN 0031-9007. doi:10.1103/physrevlett.107.178101.

[171] A. F. M. Smith and A. E. Gelfand. Bayesian Statistics without Tears: A Sampling-Resampling Perspective. *The American Statistician*, 46(2):84, 1992. ISSN 0003-1305. doi:10.2307/2684170.

[172] K. So, A. C. Koralek, K. Ganguly, M. C. Gastpar, and J. M. Carmena. Assessing functional connectivity of neural ensembles using directed information. *Journal of Neural Engineering*, 9(2):026004, 2012. ISSN 1741-2552. doi:10.1088/1741-2560/9/2/026004.

[173] V. Sourjik and H. C. Berg. Receptor sensitivity in bacterial chemotaxis. *Proceedings of the National Academy of Sciences*, 99(1):123–127, 2002. ISSN 0027-8424. doi:10.1073/pnas.011589998.

[174] V. Sourjik and H. C. Berg. Binding of the Escherichia coli response regulator CheY to its target measured in vivo by fluorescence resonance energy transfer. *Proceedings of the National Academy of Sciences*, 99(20):12669–12674, 2002. ISSN 0027-8424. doi:10.1073/pnas.192463199.

[175] V. Sourjik and H. C. Berg. Functional interactions between receptors in bacterial chemotaxis. *Nature*, 428(6981):437–441, 2004. ISSN 0028-0836. doi:10.1038/nature02406.

[176] M. Staniek and K. Lehnertz. Symbolic Transfer Entropy. *Physical Review Letters*, 100(15):158101, 2008. ISSN 0031-9007. doi:10.1103/physrevlett.100.158101.

[177] S. P. Strong, R. Koberle, R. R. de Ruyter van Steveninck, and W. Bialek. Entropy and Information in Neural Spike Trains. *Physical Review Letters*, 80(1): 197–200, 1998. ISSN 0031-9007. doi:10.1103/physrevlett.80.197.

[178] I. Sutskever, J. Martens, and G. Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICMLÍ1, pages 1017–1024, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

[179] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. *arXiv*, 2014. doi:10.48550/arxiv.1409.3215.

[180] S. Tănase-Nicola, P. B. Warren, and P. R. ten Wolde. Signal detection, modularity, and the correlation between extrinsic and intrinsic noise in biochemical networks. *Physical review letters*, 97(6):068102, 2006.

[181] Y. Tang, A. Adelaja, F. X.-F. Ye, E. Deeds, R. Wollman, and A. Hoffmann. Quantifying information accumulation encoded in the dynamics of biochemical signaling. *Nature Communications*, 12(1):1272, 2021. doi:10.1038/s41467-021-21562-0.

[182] V. H. Thanh and C. Priami. Simulation of biochemical reactions with time-dependent rates by the rejection-based algorithm. *The Journal of Chemical Physics*, 143(5):054104, 2015. ISSN 0021-9606. doi:10.1063/1.4927916.

[183] P. J. Thomas and A. W. Eckford. Capacity of a Simple Intercellular Signal Transduction Channel. *IEEE Transactions on Information Theory*, 62(12): 7358–7382, 2016. ISSN 0018-9448. doi:10.1109/tit.2016.2599178.

[184] N. Tishby and N. Zaslavsky. Deep Learning and the Information Bottleneck Principle. *arXiv*, 2015. doi:10.48550/arxiv.1503.02406.

[185] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. *arXiv*, 2000. doi:10.48550/arxiv.physics/0004057.

[186] A. J. Tjalma and P. R. ten Wolde. Predicting concentration changes via discrete receptor sampling. *Physical Review Research*, 6(3):033049, 2024.

[187] A. J. Tjalma, V. Galstyan, J. Goedhart, L. Slim, N. B. Becker, and P. R. ten Wolde. Trade-offs between cost and information in cellular prediction. *Proceedings of the National Academy of Sciences*, 120(41):e2303078120, 2023.

[188] G. Tkačik and W. Bialek. Information Processing in Living Systems. *Annual Review of Condensed Matter Physics*, 7(1):89–117, 3 2016. ISSN 1947-5454. doi:10.1146/annurev-conmatphys-031214-014803.

[189] G. Tkačik and A. M. Walczak. Information transmission in genetic regulatory networks: a review. *Journal of Physics: Condensed Matter*, 23(15):153102, 2011. ISSN 0953-8984. doi:10.1088/0953-8984/23/15/153102.

[190] G. Tkačik, C. G. Callan, and W. Bialek. Information flow and optimization in transcriptional regulation. *Proceedings of the National Academy of Sciences*, 105(34):12265–12270, 2008. ISSN 0027-8424. doi:10.1073/pnas.0806077105.

[191] G. Tkačik, A. M. Walczak, and W. Bialek. Optimizing information flow in small genetic networks. *Physical Review E*, 80(3):031920, 2009. ISSN 1539-3755. doi:10.1103/physreve.80.031920.

[192] G. Tkačik, J. O. Dubuis, M. D. Petkova, and T. Gregor. Positional Information, Positional Error, and Read-Out Precision in Morphogenesis: A Mathematical Framework. *Genetics*, 199(1):genetics.114.171850, 2014. ISSN 0016-6731. doi:10.1534/genetics.114.171850.

[193] G. Tkačik, T. Mora, O. Marre, D. Amodei, S. E. Palmer, M. J. Berry, and W. Bialek. Thermodynamics and signatures of criticality in a network of neurons. *Proceedings of the National Academy of Sciences*, 112(37):11508–11513, 2015. ISSN 0027-8424. doi:10.1073/pnas.1514188112.

[194] F. Tostevin and P. R. ten Wolde. Mutual Information between Input and Output Trajectories of Biochemical Networks. *Physical Review Letters*, 102(21): 218101, 2009. ISSN 0031-9007. doi:10.1103/physrevlett.102.218101.

[195] F. Tostevin and P. R. ten Wolde. Mutual information in time-varying biochemical systems. *Phys. Rev. E*, 81:061917, Jun 2010. doi:10.1103/PhysRevE.81.061917. URL https://link.aps.org/doi/10.1103/PhysRevE.81.061917.

[196] N. Umeki, Y. Kabashima, and Y. Sako. Evaluation of information flows in the RAS-MAPK system using transfer entropy measurements. *bioRxiv*, page 2023.08.06.552214, 2024. doi:10.1101/2023.08.06.552214.

[197] R. Van Den Berg, L. Hasenclever, J. M. Tomczak, and M. Welling. Sylvester normalizing flows for variational inference. In A. Globerson, A. Globerson, and R. Silva, editors, *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 393–402. Association For Uncertainty in Artificial Intelligence (AUAI), Jan. 2018.

[198] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1747–1756, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/oord16.html.

[199] F. van der Meulen and M. Schauer. Bayesian estimation of discretely observed multi-dimensional diffusion processes using guided proposals. *Electronic Journal of Statistics*, 11(1):2358–2396, 2017. ISSN 1935-7524. doi:10.1214/17-ejs1290.

[200] T. S. van Erp, D. Moroni, and P. G. Bolhuis. A novel path sampling method for the calculation of rate constants. *The Journal of Chemical Physics*, 118(17): 7762–7774, 2003. ISSN 0021-9606. doi:10.1063/1.1562614.

[201] N. G. van Kampen. *Stochastic Processes in Physics and Chemistry*. Elsevier, Amsterdam, 3 edition, 2007. ISBN 9780444529657. doi:10.1016/b978-0-444-52965-7.x5000-4.

[202] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. *arXiv*, 2017. doi:10.48550/arxiv.1706.03762.

[203] T. J. H. Vlugt and B. Smit. On the efficient sampling of pathways in the transition path ensemble. *PhysChemComm*, 4(2):11–17, 2001. ISSN 1460-2733. doi:10.1039/b009865p.

[204] A. M. Walczak, A. Mugler, and C. H. Wiggins. A stochastic spectral analysis of transcriptional regulatory cascades. *Proceedings of the National Academy of Sciences*, 106(16):6529–6534, 2009. ISSN 0027-8424. doi:10.1073/pnas.0811999106.

[205] A. M. Walczak, G. Tkačik, and W. Bialek. Optimizing information flow in small genetic networks. II. Feed-forward interactions. *Physical Review E*, 81 (4):041905, 2010. ISSN 1539-3755. doi:10.1103/physreve.81.041905.

[206] P. B. Warren, S. Tănase-Nicola, and P. R. ten Wolde. Exact results for noise power spectra in linear biochemical reaction networks. *The Journal of Chemical Physics*, 125(14):144904, 10 2006. ISSN 0021-9606. doi:10.1063/1.2356472. URL `https://doi.org/10.1063/1.2356472`.

[207] G. S. Watson. Smooth Regression Analysis. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, 26(4):359, 1964.

[208] M. F. Weber and E. Frey. Master equations and the theory of stochastic path integrals. *Reports on Progress in Physics*, 80(4):046601, 4 2017. ISSN 0034-4885. doi:10.1088/1361-6633/aa5ae2.

[209] W. Weidlich and M. Braun. The master equation approach to nonlinear economics. *Journal of Evolutionary Economics*, 2(3):233–265, 1992. ISSN 0936-9937. doi:10.1007/bf01202420.

[210] E. Ziv, I. Nemenman, and C. H. Wiggins. Optimal Signal Processing in Small Stochastic Biochemical Networks. *PLoS ONE*, 2(10):e1077, 2007. doi:10.1371/journal.pone.0001077.

# Summary

Understanding and quantifying information transmission is crucial for improving and analyzing biological and engineered systems. Most, if not all, information-processing systems process signals that vary in time and generally information is not just contained in the instantaneous values of these signals, but also in their temporal characteristics. Quantifying information transmitted via time-varying signals thus requires the mutual information between input and output signals as a function of time, i.e., between the input and output trajectories. Moreover, the trajectory mutual information is needed for calculating the key performance measure of information processing—the mutual information rate—quantifying the amount of information transmitted per unit time. Yet, computing the mutual information between trajectories is notoriously difficult due to the high dimensionality of trajectory space. The curse of dimensionality makes traditional non-parametric information estimates infeasible, so existing methods for computing the mutual information between trajectories rely on approximations or simplifying assumptions.

This thesis introduces Path Weight Sampling (PWS), a novel Monte Carlo framework for exactly calculating the trajectory mutual information for any system described by a dynamical stochastic model. The principal idea is to use the stochastic model to evaluate the exact conditional probability of an individual output trajectory, for a given input trajectory, and to compute the marginal probability for the same output trajectory via a Monte Carlo marginalization in trajectory space. By averaging the log-ratio of these probabilities over many stochastic realizations of the system, we obtain an unbiased estimate of the trajectory mutual information. Moreover, PWS also makes it possible to compute the mutual information between input and output trajectories for systems with hidden internal states as well as systems with feedback from output to input. In this thesis, we present three variants of PWS, which all compute the conditional probability in the same way but differ in the way the marginal output probability is obtained.

In **Chapter 2**, we present Direct PWS, the simplest variant of PWS which computes the marginal output probability as a brute-force average over the conditional trajectory probabilities. While this scheme is feasible for simple systems, the direct Monte Carlo averaging procedure becomes more difficult for larger systems and exponentially harder for longer trajectories.

Our second and third variants of PWS are based on the realization that computing the marginal trajectory probability for a stochastic model is equivalent to computing

the partition function in statistical physics. These schemes leverage techniques for computing free energies from statistical physics. In **Chapter 3**, we first introduce Rosenbluth-Rosenbluth PWS (RR-PWS) which exploits the analogy between signal trajectory sampling and polymer sampling, and is based on efficient techniques for computing the (excess) chemical potential of a polymer. Secondly, we introduce thermodynamic integration PWS (TI-PWS) which is based on using MCMC sampling trajectory space to compute the marginal probability via thermodynamic integration. We apply PWS to a simple toy model of gene expression, showing that both of these variants, but especially RR-PWS, significantly improve the performance of PWS.

To demonstrate the power of PWS and to gain new insights about the accuracy of biochemical signaling we apply PWS to the bacterial chemotaxis system, a complex biological information-processing system. In **Chapter 4**, we build a mechanistic model of chemotaxis based on previous literature and use PWS with this model to compute the mutual information rate of a bacterium in a shallow gradient. By comparing our model against experiments performed by Mattingly et al. [107], we find discrepancies between the model predictions and the experimental results. We resolve these discrepancies by adapting our literature-based model, suggesting that in *E. coli* the number of receptor clusters is much smaller than hitherto believed, while their size is much larger. Moreover, using the adjusted model we find that in shallow gradients the mutual information rate computed using PWS closely matches the rate obtained by Mattingly et al. via a Gaussian approximation, justifying their use of the approximation *a posteriori*.

The Gaussian approximation for the mutual information rate is widely used in practice, yet it relies on assumptions of linear dynamics and additive Gaussian noise which are often violated in inherently nonlinear physical or biological systems. To assess the accuracy of the Gaussian approximation when applied to nonlinear systems we require an exact method, such as PWS, to provide an accurate benchmark for the trajectory mutual information. In **Chapter 5** we use PWS to systematically assess the accuracy of the Gaussian approximation through two case studies: first, a discrete linear system which has near-Gaussian statistics but leads to a surprisingly large error in the Gaussian information rate; and second, a continuous diffusive system with a nonlinear transfer function, allowing us to quantify the error of the Gaussian approximation as nonlinearity increases. These findings highlight key instances where the Gaussian approximation fails and exact methods like PWS become essential.

While PWS does not suffer from the limitations of the Gaussian approximation, it requires a stochastic model of the system of interest, which is often unavailable. In **Chapter 6** we leverage recent advances in machine learning to learn a data-driven stochastic model directly from experimental time-series data and use PWS with

the resulting model to compute the trajectory mutual information. This approach, which we call ML-PWS, makes it possible to compute the mutual information rate of nonlinear systems, even in the absence of a known mechanistic or phenomenological model. We demonstrate that ML-PWS can yield highly accurate mutual information estimates purely from data, outperforming the Gaussian approximation for nonlinear systems.

# List of Publications

**M. Reinhardt**, G. Tkačik, and P. R. ten Wolde, *Path Weight Sampling: Exact Monte Carlo Computation of the Mutual Information between Stochastic Trajectories*, Physical Review X **13**, 041017 (2023).

A. Tjalma*, **M. Reinhardt***, A.-L. Moor, and P. R. ten Wolde, *Mutual Information Rate—Linear Noise Approximation and Exact Computation*, in preparation.

**M. Reinhardt**, G. Tkačik, and P. R. ten Wolde, *ML-PWS: Computing the Mutual Information between Stochastic Trajectories using Neural Networks*, in preparation

†A.-L. Moor, A. Tjalma, **M. Reinhardt**, P. R. ten Wolde, and C. Zechner, *State- versus Reaction-Based Information Processing in Biochemical Networks*, arXiv preprint arXiv:2505.13373 (2025).

*These authors contributed equally
†Publication not appearing in this thesis.

# Acknowledgments

This thesis could not have been written without the support of many people. More than just scientific support I received lots of emotional support which is essential yet underrated. To express my gratitude for this support I want to acknowledge the wonderful people who made this possible.

First, **Pieter Rein**, thank you for sticking with me, having my back, fighting for me, and helping me through thick and thin. You always were able to reignite my enthusiasm, even when my own motivation would waver. You provide a great atmosphere for research, you are always available for discussions, you care about the people at least as much as about the project, and you are willing to consider different perspectives and evolve your views. Your commitment to fair, open, and rigorous science is one of your strongest qualities. During our many meetings I learned a lot from you, not just about research but more generally about life.

Thank you also to my co-supervisor **Gašper**. Your advice can only be described as magical. Many times, when Pieter Rein and I found ourselves stuck on a certain problem, an email to you would result in an eager reply filled with extremely useful suggestions and creative ideas that set us back on track. I also deeply appreciated your hospitality when you hosted me in Vienna for a few weeks; I felt warmly welcomed by your group and left truly inspired.

**Mareike**, you were the first person I met when I stepped into AMOLF on my first day as a master's student. I immediately sensed a connection with you. My senses did not betray me; from that moment on we started developing a friendship that has become one of my most treasured. You have been of tremendous importance to me, and I am not exaggerating when I say that without you, this thesis would not exist. It all started even before I moved to Amsterdam. As a master's student contemplating an internship position at the biochemical networks group at AMOLF, I looked for the most recently hired PhD student in this group—you—and emailed to ask about the work environment. The response I received was not only overwhelmingly positive but also incredibly kind, convincing me to come to Amsterdam. About a year later, when I was offered a PhD position at AMOLF, you were the first person I wanted to share the news with. I vividly remember our joy about me being able to stay for 4 more years. Throughout the ensuing PhD time, your example and enthusiasm provided strong guidance and support. But you have been far more than a wonderful mentor at AMOLF. Many of our cherished stories unfolded outside of work, including countless trips, crazy parties and beautiful evenings sharing a bot-

tle of wine. Many of the people that I thank below I know through you. I am happy to look back and see that we built a little community here in Amsterdam that has shaped our lives. I feel so grateful for having met you.

I want to thank my paranymphs, **Christian** and **Katharina**, who have been a never-ending source of inspiration, support, and fun over the course of my PhD. **Chris**, I am deeply grateful for the privilege of closely sharing the journey of life with you over the past few years. Our friendship means so much to me. You genuinely care about the people close to you, and your presence is a gift. Maybe you don't know this but I admire your eloquence. You have a remarkable ability to find words for expressing your sensations and feelings genuinely and precisely. With this comes a deep sense of security that you embody and radiate. With you by my side I feel empowered to push boundaries and step out of my comfort zone, yet always feeling safe. I look back on so many joyful, exciting, and daring adventures we shared and undoubtedly will keep sharing. It is rare to find someone like you whom I can fully trust one hundred percent. Moreover, your enthusiasm is contagious and when you deeply care about something it is truly inspiring and moving. We have transformed together, learned so much through each other, and I am certain that our mutual growth will continue. **Kathi**, you are a beacon of joy in my life. Over the years, we must have accumulated hundreds of hours of talking to each other about anything and everything. I remember countless times of coming to your office or to the tweezer lab for a "quick question" or a bit of gossip, only to find ourselves, two hours later, still completely lost in conversation. With you, any ordinary activity transforms into an exciting mission, whether it's shopping for lamps, finding the best Airbnb, shooting a movie scene, exploring activities at Christmas lunch, or discovering the best party locations at pride. We complement each other so well as a team and I am looking forward to our future missions. You care deeply about the people around you, always willing to help with a lot of empathy. Because you're naturally drawn to reflect deeply on situations and people, your advice is incredibly valuable and thoughtful. I know that whenever I struggle, I can count on you. Moreover, I admire your devotion and energy for pursuing the things you care about, whether it is a scientific publication, sleeping for a night in the mad king's castle, fairly managing interpersonal conflict, or obtaining K-pop tickets for a sold-out show. Thank you for all the time we spend together—whether dancing, supporting each other, deeply reflecting about philosophy, working on a project, or simply being in each other's presence.

Next, I want to thank my partynymphs, **Lucie** and **Imme**, for bringing so much light and enlightenment into my life. **Imme**, we met during a global pandemic, yet we managed to build a close friendship within weeks. And then you had the wonderfully bold idea of doing a house swap: me living in Utrecht and you living in Amsterdam for a week. It was a success and your house since has always felt like a

second home to me, a special and beautiful connection. I am happy to say that bold ideas and outside-of-the-box thinking are rather the rule than the exception with you. You are so inspiring! You introduced me to the wonderful sport of ice skating, which I started enjoying so much that we decided to sign up for courses together religiously every winter. You also made me listen to your favorite Dutch artists which led to me enjoying the music so much that we ended up going to music festivals together for it. I am curious what will come next! We have navigated ups and downs together, forging a bond so strong that I am convinced we will remain connected, no matter what the future holds. **Lucie**, with our overlapping social circles, it was only a matter of time before we became friends ourselves. Yet, when our friendship finally formed, it surpassed all my expectations. We have incredibly deep chats about a plethora of topics, and I am very grateful for all the amazing advice you offer. Not only that, but we can also laugh endlessly over the same »joke« for months. You are so much fun to be around! And your heart is in the right place, too. Your passion for fighting patriarchy and tackling the climate catastrophe is powerful and contagious. The energy and thought you bring into your relationships—with your partner, family, friends, and yourself—deeply inspire me. Thank you for sharing your enthusiasm and opinions, your joy and wisdom.

**Laura**, meeting you led to the single most impactful shift in my Amsterdam life. Without knowing each other all too well, we decided not only to move in together, but also to plan a holiday to Belice. Although we ended up going to Mexico instead, I am glad that we took the courage for both decisions as they marked the start of our adventures together. You are a wonderfully kind and loving flatmate, as well as a brave and enthusiastic travel companion; what a unique combination! Beyond Mexico, we have explored many beautiful locations together, including Noordwijk, Vienna, Nice, Munich, and Colombia. I hope this is only the beginning of a list that will keep growing. Not only that, but we also manage to make life in Amsterdam an adventure. We have seen countless movies together, shopped in the most exclusive boutiques, we have been partners in crime (literally). Our shared excitement for niche French concerts brings me joy. I also want to especially thank you for supporting me in the final and sometimes challenging stretches of the PhD. You made that time so much easier for me. With few people can I feel such uninhibited joy as with you and I want to thank you for that.

I want to thank my present and past colleagues from the biochemical networks group for making daily work so enjoyable. To **Harmen**, who left the group not long after I started, thank you for welcoming me into the group and helping me get started. **Alex**, I remember many fun lunches with you and Mareike during the pandemic which would represent the highlight of my day. You have a great sense of humor as well as very deep knowledge about many topics but what stands out most about you is your sense of true curiosity. Not only are you curious in a scientific

sense about the mysteries of the world, but even more so curious and enthusiastic about the people that surround you. Thank you for our deep conversations, for your advice and wisdom. **Age**, we started our PhD on the same day, and our defenses are only six days apart. It has been such a joy to embark on this journey alongside you. I admire your capacity for structured thought, your analytical math skills and your pragmatic, down-to-earth attitude. It makes working with you a real inspiration and joy. I feel we are such a good team together, which is proven by how quickly we were able to finish our project together. I enjoyed all the conferences that we went to together. I also appreciate your sailing skills that saved us from the sand bank. **Jenny**, thank you for radiating joy and excitement as well as always being willing to engage in thoughtful debate about topics that you care about. **Mike**, I greatly appreciated your passion for helping everyone in the group with our projects. Your vast knowledge and willingness to share it truly made a difference. **Muriël**, it was a pleasure to supervise you and witness you gradually taking ownership and responsibility for your project. **Maxim**, your good spirits and intelligent questions were consistently inspiring and your project became a big success. **Ramon**, I've had some of the most interesting conversations with you; thank you for sharing your knowledge. I applaud your principled attitude towards science and societal topics in general; you represent integrity and honesty in a way few people do. I thank you for your involvement in the works council for improving working conditions for everyone. **Avishek**, I am happy that after getting acquainted with each others' humor, we can finally laugh so much together. I deeply appreciate all the work and thought you put into extending PWS and all the incredibly illuminating discussions we had as a result. **Julien**, though I often only saw you on screen during group meetings, your visits to Amsterdam were always a lot of fun. I especially fondly remember cooking a delicious meal with you on the sailing boat during our group outing. **Vahe**, I appreciate your unique perspective on science and your genuine fascination for nature and physics. **Daan**, I am delighted by your unconventional way of thinking, which adds so much value to the group! Thank you for bringing Dutch poetry to the group lunch. I wish you a lot of success in finishing your PhD. **Claudio**, you radiate positive energy. In the relatively short time you have been at AMOLF, we have connected on so many layers beyond science. **Menno**, you just started your PhD and even though we overlapped very briefly, I enjoyed exchanging ideas with you. **Geert** and **Robin**, it was a joy to see you taking on your master's projects with enthusiasm and finishing them successfully.

**Bela**, thank you for showing up for the daily coffee break, every morning at 10:30, without fail. Fortunately, even during the pandemic you insisted on keeping the coffee break alive via virtual sessions, which I am very grateful for. I don't know how I would have been able to endure the social isolation without these regular meetings. Both the virtual and the real coffee breaks were essential for creating a feeling

of connection and belonging within the group. Your retirement from AMOLF had a big impact on me and the group, and coffee breaks would not be the same anymore but I am happy that you are still regularly coming to AMOLF even after your retirement.

A special thanks goes to our experimental collaborators at AMOLF. **Fotis**, your expertise of *E. coli* chemotaxis is spectacular and talking to you about the biological intricacies was very fascinating and useful. **Evan**, I admire your determination and attention to detail in doing complex single-cell *E. coli* experiments. Moreover, your way of presenting science is magnetic. Thank you for putting your heart into this project! **Tom**, thank you for seemingly never stopping to think deeply about any research question as well as for all the advice you have offered me. Your scientific questions and comments show a very deep understanding of and passion for science. Your interest in my projects was always a huge motivator.

I also enjoyed fruitful collaborations and interactions beyond AMOLF. A special thanks goes to my collaborators at MPI-CBG in Dresden. **Anne**, you took the initiative in forming a collaboration between our labs and I am so grateful for that. It was amazing to see how you managed to build the mathematical tools to answer a question that seemed so difficult to address. I enjoyed working with you and talking with you. With you, it's a double-edged sword. One one hand, I am excited to have found someone that works in the same niche as me. On the other hand, you also humble me as you clearly understand information transmission on deeper level than me. **Christoph**, thank you for many insightful conversations and your honest feedback on ideas and drafts.

A special thanks also goes to the members of the Tkačik lab at ISTA who welcomed me so enthusiastically during my time in Vienna. **Camila**, **Tereza**, **Natalia**, **Bahti**, **Julian**, **Athina**, **Simon**, **Réka**, **Michal**, **David**, it was great meeting all of you. I particularly enjoyed our stimulating conversations about science and many other topics in the cafeteria as well as various fun trips into the city of Vienna.

Beyond those I directly worked with, I want to thank many other people at AMOLF whom I am very grateful to have gotten to know during my PhD. **Moritz** thank you for your kindness, loyalty, and for being a fantastic companion in our bouldering sessions and as we learned to climb outdoors together. **Anne-Sophie**, it is always so much fun (and perhaps slightly unproductive) when we are in the office together. Thank you for all the hilarious, deep, senseless, and funny conversation. Moreover, I want to thank you for all the absurd adventures we have had together outside of work. Every memory we make is a great one and we are unquestionably gifted at picking out amazing locations to eat or to party. I look forward to our future adventures! **Kim**, thanks for sharing your incredibly positive energy. The year you were at AMOLF was so much fun. **Francesca** it is always fun to go bouldering together and I enjoyed the parties together. Thank you for bringing me into your

friend group. **Timo**, thank you for sharing your passion for cooking and preparing some chef-quality meals and desserts. You are incredibly generous. **Lori**, it was fun to chat and play board games with you. I enjoyed being your neighbor.

I want to thank the PV, the staff association which brings so much energy and fun to AMOLF through its events. Thanks to **Jente**, **Yvonne**, **Evelijn**, **Igor**, **Sofija**, **Mels**, **Mees**, **Teressa** and all the other PV members, past and present. I deeply appreciate your efforts to bring joy to the AMOLF community, and I cherish the years I spent in the PV myself. Especially the Christmas party was always such a blast!

A very big thank you to the support departments of AMOLF. In particular, I want to thank for the great support from the ICT department in setting up and maintaining the computational cluster which was indispensable for me. Thanks, **Gerben**, **Michiel**, **Sean**, **Kallinikos**, **Cees**, and **Pieter** for solving all the problems regarding the cluster. I appreciate all the work that goes into maintaining it working. Thanks also **Erny** and **Petra** for helping with communicating my research to a wider audience and for organizing the Science Day. Thanks to **Clyde** for caring about workplace ergonomics, to **Hinco** for being able to fix anything, anytime and being cheerful meanwhile. I appreciate the HR department, especially **Wouter** and **Linda**, for helping me find housing and answer all contract-related questions.

I also want to thank the wonderful people I met outside of AMOLF and that shaped my life. **Nina**, thanks for the joy of dancing together, for chatting endlessly, and for sharing our passion for music. **Maxime**, thank you for bringing so much fun, for patiently teaching countless board games, and for the wonderful time you spent in Amsterdam; it was truly a joy having you here. **Victor**, every time I see you you cheer me up with your humor and positive attitude. Thank you for taking me to the best Mexican concert in Amsterdam. **Lucas**, your friendly calmness is addictive. You've got a great sense of humor and I admire your dedication to finishing what you started, whether it is a bike race or a very hot chili pepper. **Mirko**, you are wonderfully down-to-earth and have a perfect subtle wit that always makes me laugh. **Tamara**, you are a person full of life and love. Thank you so much for sharing your amazing joy with the people around you.

And, thank you to my friends from back home in Germany. **Victoria**, thank you for always being on my side. **Dennis** and **Lukas**, you both know me since the very early days. It makes me so happy that we still maintain a connection that persists through all the changes that life brings. I am so curious where all our journeys will bring us. **Ben**, **Stefan**, **Flo**, **Julian**, **Veronica**, **Rudi**, **Christian**, thanks for helping me with the physics homework during our studies. The effort finally paid off. We have all come such a long way since then!

Finalmente, gracias a mi familia. Vuestro amor incondicional ha sido tan importante para mí. Gracias por celebrar cada logro conmigo y por ayudarme en los

momentos difíciles. **Mamá**, gracias por estar siempre a mi lado y ayudarme en todas las situaciones y etapas de mi vida. Gracias por dejarme la libertad de hacer lo que a mí me gusta y encontrar mi propio camino. Gracias por leer conmigo y enseñarme la belleza de la literatura. Sin ti no hubiera llegado hasta aquí.

**Juana**, thank you for creating the magnificent cover art for this thesis.

# About the Author

Manuel Reinhardt was born on May 16, 1996, in Ebersberg, Germany. He grew up with both German and Spanish roots. His early fascination with science and problem-solving led him to participate in the International Physics Olympiad, where he reached the third national round in Germany during the 2013/14 competition.

Driven by a desire to understand the fundamental rules that govern the world, Manuel pursued a Bachelor of Science in physics at the Ludwig-Maximilians-University in Munich from 2014 to 2018. For his bachelor's thesis, he focused on computational quantum physics, reflecting his growing interest in numerical simulations. Seeking to broaden his horizons, he spent a semester studying abroad at Universidad Autónoma de Madrid as part of the Erasmus+ program.

After his experience abroad, Manuel continued his studies at LMU Munich, graduating with a Master of Science in 2020. Inspired by his previous experience abroad, he decided to complete his master's thesis in Amsterdam, working under the supervision of Pieter Rein ten Wolde. Part of his work during this period are presented in this thesis.

In 2020, Manuel embarked on his PhD at AMOLF Amsterdam, continuing in the Biochemical Networks group under Pieter Rein ten Wolde's supervision. His research explores the fundamental principles of biological information flow, using theoretical and computational approaches to understand how cells process information. The results of this research are presented in this thesis.