Path Weight Sampling: Exact Monte Carlo Computation of the Mutual Information between Stochastic Trajectories

Manuel Reinhardt[*],[1,*] Gašper Tkačik[*],[2] and Pieter Rein ten Wolde[*],[1,†]

[1]*AMOLF, Science Park 104, 1098 XG, Amsterdam, The Netherlands*
[2]*Institute of Science and Technology Austria, 3400 Klosterneuburg, Austria*

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. We present three variants of PWS, which all generate the trajectories using the standard stochastic simulation algorithm. While direct PWS is a brute-force method, Rosenbluth-Rosenbluth PWS exploits the analogy between signal trajectory sampling and polymer sampling, and thermodynamic integration PWS is based on a reversible work calculation in trajectory space. 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. Applying PWS to the bacterial chemotaxis system, consisting of 182 coupled chemical reactions, demonstrates not only that the scheme is highly efficient but also that the number of receptor clusters is much smaller than hitherto believed, while their size is much larger.

Subject Areas: Biological Physics,
Computational Physics,
Statistical Physics

I. INTRODUCTION

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 [1–3]. Shannon introduced the mutual information and the information rate as the central measures of information theory more than 70 years ago [4]. These measures quantify the fidelity by which a noisy system transmits information from its inputs to its outputs. However, computing these quantities exactly remains notoriously difficult, if not impossible, because the inputs and outputs are often not scalar values but rather temporal trajectories.

Most, if not all, information-processing systems transmit signals that vary in time. The canonical measure for quantifying information transmission via time-varying signals is the mutual information rate [4–7]. 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. In the absence of feedback, this rate also equals the multistep transfer entropy [8,9].

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

---

[*]Corresponding author: reinhardt@amolf.nl
[†]Corresponding author: tenwolde@amolf.nl

has changed in the recent past [10,11]. Moreover, the information from the input may be encoded in the temporal dynamics of the output [12–15]. 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 [6].

Unfortunately, computing the mutual information between trajectories is exceptionally difficult. The conventional approach requires nonparametric distribution estimates of the input and output distributions, e.g., via histograms of data obtained through simulations or experiments [16–21]. These nonparametric distribution estimates are necessary because the mutual information generally cannot 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 [22]. Except for the simplest systems with a binary state space [21], 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-neighbor entropy estimator [23,24], 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 [26]; however, they merely provide a lower bound of the mutual information, and it remains unclear how tight these lower bounds are [25,27,28]. Analytical results are available for simple systems [29]; for linear systems that obey Gaussian statistics, the mutual information between trajectories can be obtained from the covariance matrix [6]. However, many information processing systems are complex and nonlinear 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 [30] and generalized in Ref. [31]. 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 [32–35], biological population growth [36–38], economic processes [39,40], and a large variety of other systems [41,42], expanding 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 $\mathcal{P}[x]$ of the output trajectories x and the conditional output entropy associated with the output distribution $\mathcal{P}[x|s]$ conditioned on the input trajectory s. Our scheme is inspired by the observation of Cepeda-Humerez *et al.* [25] that the path likelihood, i.e., the probability $\mathcal{P}[x|s]$, 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 $\mathcal{P}[x|s]$ can also be computed for a dynamical input trajectory 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 $\mathcal{P}[x]$ cannot be computed with the approach of Cepeda-Humerez *et al.* [25]; thus, it requires a different scheme.

We show how, for time-varying input signals, the marginal probability $\mathcal{P}[x]$ can be obtained as a Monte Carlo average of $\mathcal{P}[x|s]$ over a large number of input trajectories. How to do this effectively is the crux of PWS. We then use the Monte Carlo estimate for $\mathcal{P}[x]$ to compute the marginal output entropy. We 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 $\mathcal{P}[x]$ is carried out.

To compute $\mathcal{P}[x]$, direct PWS (DPWS) performs a brute-force average of the path likelihoods $\mathcal{P}[x|s]$ over the input trajectories 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.

Our second and third variants of PWS are based on the realization that the marginal probability $\mathcal{P}[x]$ is akin to a partition function. These schemes leverage techniques for computing free energies from statistical physics. Specifically, the second scheme, Rosenbluth-Rosenbluth PWS (RR-PWS), exploits the observation that the computation of $\mathcal{P}[x]$ is analogous to the calculation of the (excess) chemical potential of a polymer, for which efficient methods have been developed [43–45]. The third scheme, thermodynamic integration PWS (TI-PWS), is based on the classic free-energy estimation technique of thermodynamic integration [46–48] in conjunction with a

trajectory space MCMC sampler using ideas from transition path sampling [49].

All three PWS variants make it possible to compute the mutual information between trajectories exactly, without the need for time discretizing the input and output signals.

The DPWS method is presented in Sec. II, followed by a review of the required concepts from the theory of Markov jump processes and master equations. The other two PWS schemes are then presented as improvements of DPWS in Sec. III.

In Sec. IV, 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 Appendix C.

In Sec. V, we apply PWS to two well-known model systems. The first is a simple pair of coupled birth-death processes, which allow 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 [6] and the technique by Duso and Zechner [30]. Our second application concerns the bacterial chemotaxis system, which is arguably the best characterized signaling system in biology. Mattingly *et al.* [50] recently argued that bacterial chemotaxis in shallow gradients is information limited. However, to compute the information rate from their experimental data, they had to employ a Gaussian framework. PWS makes it possible to assess the accuracy of this approximation. Our results show that the Gaussian framework is accurate in the regime of shallow concentration gradients as studied by Mattingly *et al.* [50]. However, comparing the PWS predictions of a model based on previous literature against their experimental results reveals that the composition of the receptor array differs from that hitherto believed.

II. MONTE CARLO ESTIMATE OF THE MUTUAL INFORMATION

In this section, we present the fundamental ideas of PWS. These ideas lie at the heart of DPWS and also form the foundation of the other two, more-advanced, PWS variants, which will be explained in subsequent sections.

A. Statement of the problem

All information processing systems repeatedly take an input value s and produce a corresponding output x. Because of noise, the output produced for the same input
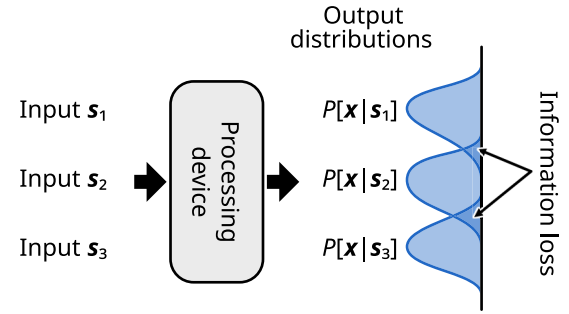

FIG. 1. Schematic of information processing under the influence of noise. Overlapping output distributions for different inputs lead to information loss because the input cannot always be uniquely inferred from the output. The mutual information $I(\mathcal{S}, \mathcal{X})$ quantifies how much information the observation of the output typically retains about the input signal.

can be different every time, such that the system samples outputs from the distribution $P(x|s)$. 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 $P(s)$ such that the whole setup of the signal and device is completely described by the joint probability density $P(s, x) = P(s)P(x|s)$.

When the conditional output distributions $P(x|s)$ overlap with each other, information is lost because the input can not always be inferred uniquely from the output (see Fig. 1). The remaining information that the output carries about the signal, on average, is quantified by the mutual information between the 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 as

$$I(\mathcal{S}, \mathcal{X}) = \iint ds \, dx \, P(s, x) \, \ln \frac{P(s, x)}{P(s)P(x)}, \qquad (1)$$

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 nonnegative real number, representing the mutual information between $\mathcal{S}$ and $\mathcal{X}$ in nats. The integrals in Eq. (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, so the integrals become path integrals.

In general, the mutual information can be decomposed into two terms—conditional and marginal entropy. Because of the symmetry of Eq. (1) with respect to the exchange of $\mathcal{S}$ and $\mathcal{X}$, this decomposition can be written as

$$I(\mathcal{S}, \mathcal{X}) = H(\mathcal{S}) - H(\mathcal{S}|\mathcal{X}) = H(\mathcal{X}) - H(\mathcal{X}|\mathcal{S}). \quad (2)$$

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

$$H(\mathcal{X}) = - \int dx\, P(x)\,\ln P(x), \qquad (3)$$

$$H(\mathcal{X}|\mathcal{S}) = - \int ds\, P(s) \int dx\, P(x|s)\,\ln P(x|s). \quad (4)$$

The conventional way of computing the mutual information involves generating many samples to obtain empirical distribution estimates for $P(x|s)$ and $P(x)$ 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 [17]. 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.
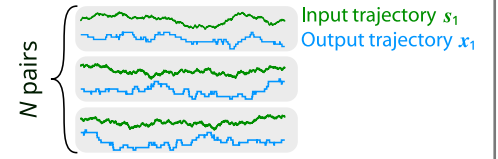
B. Direct PWS

The central idea of PWS is to compute probability densities for trajectories exactly, sidestepping the problem of having to estimate them via histograms. We exploit the fact 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 show the following:

(i) 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.

(ii) 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.

(iii) By combining the two ideas above, we can derive a direct Monte Carlo estimate for the mutual information $I(\mathcal{S}, \mathcal{X})$, as illustrated in Fig. 2.

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

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

Input trajectory s_1
Output trajectory x_1

N pairs

2. **Compute likelihoods** $P[x_i|s_i]$

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

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

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

4. **Compute the mutual information**

$$I(S, X) = \frac{1}{N} \sum_{i=1}^{N} (\ln P[\boldsymbol{x}_i|\boldsymbol{s}_i] - \ln P[\boldsymbol{x}_i])$$

FIG. 2. PWS scheme to compute the mutual information between trajectories in four 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. (14). 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 Sec. II B, whereas RR-PWS and TI-PWS are presented in Secs. III B and III C. 4. Using the likelihoods and the marginal probabilities from the previous steps, we can estimate the mutual information using Eq. (10).

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

$$\langle f[z] \rangle_{\mathcal{P}[z]} \equiv \int \mathcal{D}[z] \mathcal{P}[z] f(z). \qquad (5)$$

We write $\int \mathcal{D}[z]$ to denote a path integral over all possible trajectories of a given duration. We estimate $\langle f[z] \rangle_{\mathcal{P}[z]}$ by generating a large number of trajectories $z_1, \dots, z_N$ from $\mathcal{P}[z]$ and evaluating the corresponding Monte Carlo average

$$\hat{f}_N = \frac{1}{N} \sum_{i=1}^{N} f(z_i), \qquad (6)$$

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

$$
\begin{aligned}
\mathrm{H}(\mathcal{X}|\mathcal{S}) &= -\int \mathcal{D}[\boldsymbol{s}]\mathcal{P}[\boldsymbol{s}] \int \mathcal{D}[\boldsymbol{x}]\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] \ln \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] \\
&= -\langle\langle \ln \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]\rangle_{\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]}\rangle_{\mathcal{P}[\boldsymbol{s}]} \\
&\approx -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{K}\sum_{j=1}^{K} \ln \mathcal{P}[\boldsymbol{x}_{i,j}|\boldsymbol{s}_i].
\end{aligned}
\tag{7}
$$

Second, 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 $\mathcal{P}[\boldsymbol{x}]$:

$$
\begin{aligned}
\mathcal{P}[\boldsymbol{x}] &= \int \mathcal{D}[\boldsymbol{s}]\mathcal{P}[\boldsymbol{s}]\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] \\
&= \langle \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]\rangle_{\mathcal{P}[\boldsymbol{s}]} \\
&\approx \frac{1}{M}\sum_{j=1}^{M}\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}'_j].
\end{aligned}
\tag{8}
$$

The estimate for the marginal entropy is then given by

$$
\begin{aligned}
\mathrm{H}(\mathcal{X}) &= -\int \mathcal{D}[\boldsymbol{x}]\mathcal{P}[\boldsymbol{x}] \ln \mathcal{P}[\boldsymbol{x}] \\
&= -\langle \ln \mathcal{P}[\boldsymbol{x}]\rangle_{\mathcal{P}[\boldsymbol{x}]} \\
&\approx -\frac{1}{N}\sum_{i=1}^{N} \ln \mathcal{P}[\boldsymbol{x}_i] \\
&\approx -\frac{1}{N}\sum_{i=1}^{N} \ln \left[\frac{1}{M}\sum_{j=1}^{M}\mathcal{P}[\boldsymbol{x}_i|\boldsymbol{s}'_{i,j}]\right].
\end{aligned}
\tag{9}
$$

In the last step, we inserted the result from Eq. (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:

$$
\mathrm{I}(\mathcal{S}, \mathcal{X}) = \int \mathcal{D}[\boldsymbol{s}] \int \mathcal{D}[\boldsymbol{x}]\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}] \ln \frac{\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]}{\mathcal{P}[\boldsymbol{x}]}
$$
$$
= \langle \ln \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] - \ln \mathcal{P}[\boldsymbol{x}]\rangle_{\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]}.
\tag{10}
$$

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 DPWS, we use Eq. (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 Sec. III below, we will 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 and exact scheme that is straightforward to implement to compute the mutual information.

Having explained the core ideas of our technique above, we continue this section with a review of the necessary concepts of master equations to implement PWS. First, in Sec. II C, we derive the formula for the conditional probability $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$, which lies at the heart of our technique. In Secs. II C and II D, 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 Sec. III, we present the two other variants of PWS that improve on DPWS.

C. Driven Markov jump process

Throughout this article, 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 Ω is multidimensional 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

$$
\frac{\partial \mathrm{P}(x, t)}{\partial t} = \sum_{\substack{x' \in \Omega \\ x' \neq x}}[w_t(x, x')\mathrm{P}(x', t) - w_t(x', x)\mathrm{P}(x, t)],
\tag{11}
$$

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 $Q_t(x, x) = -\sum_{x' \in \Omega\backslash\{x\}} w_t(x', x)$ the master equation simplifies to

$$
\frac{\partial \mathrm{P}(x, t)}{\partial t} = \sum_{x' \in \Omega} Q_t(x, x')\mathrm{P}(x', t).
\tag{12}
$$

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

$$\mathcal{P}[\boldsymbol{x}] = P(x_0) \times \left(\prod_{i=1}^{n-1} Q_{t_i}(x_i, x_{i-1}) \right)$$
$$\times \left(\prod_{i=1}^{n} \exp \int_{t_{i-1}}^{t_i} dt Q_t(x_{i-1}, x_{i-1}) \right), \quad (13)$$

a product of the probability of the initial state $P(x_0)$, 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 $i = 1, \ldots, n$.

1. Computing the likelihood $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$

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. (13):

$$\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] = P(x_0|s_0) \times \left(\prod_{i=1}^{n-1} Q_{t_i}(x_i, x_{i-1}; \boldsymbol{s}) \right)$$
$$\times \left(\prod_{i=1}^{n} \exp \int_{t_{i-1}}^{t_i} dt Q_t(x_{i-1}, x_{i-1}; \boldsymbol{s}) \right), \quad (14)$$

where $P(x_0|s_0)$ 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. (14) 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

$$\ln \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] = \ln P(x_0|s_0) + \int_{t_0}^{T} dt \mathcal{L}_t[\boldsymbol{s}, \boldsymbol{x}], \quad (15)$$

where

$$\mathcal{L}_t[\boldsymbol{s}, \boldsymbol{x}] = Q_t(x(t), x(t); \boldsymbol{s})$$
$$+ \sum_{i=1}^{n-1} \delta(t - t_i) \ln Q_t(x_i, x_{i-1}; \boldsymbol{s}). \quad (16)$$

The computation of the log-likelihood $\ln \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ for given trajectories $\boldsymbol{s}$ and $\boldsymbol{x}$ according to Eqs. (15) and (16) proceeds as follows:

(i) At the start of the trajectory, we compute the log-probability of the initial condition $\ln P(x_0|s_0)$,
(ii) For every jump $x_{i-1} \to x_i$ in $\boldsymbol{x}$, we compute the log jump propensity $\ln Q_{t_i}(x_i, x_{i-1}; \boldsymbol{s})$.
(iii) 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 as in our examples in Sec. V.

The sum of the three contributions above yields the exact log-likelihood $\ln \mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ as given in Eq. (15).

Thus, notably, 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 quantity in Eq. (15) that cannot be directly obtained from the master equation is the log-probability of the initial state, $\ln P(x_0|s_0)$.

Our scheme can be applied to any system with a well-defined (nonequilibrium) initial distribution $P(s_0, x_0)$ as specified by, e.g., the experimental setup. Most commonly though, one is interested in studying information transmission for systems in a steady state. Then, the initial condition $P(s_0, x_0)$ 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 [51,52] (possibly using simplifying approximations [53,54]) or computationally from stochastic simulations [55].

2. Sampling from $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$

Standard kinetic Monte Carlo simulations naturally produce exact samples of the probability distribution $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$ as defined in Eq. (14). In other words, for any signal trajectory $\boldsymbol{s}$ and initial state x_0 drawn from $P(x_0|s_0)$, 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 [55], this is done by alternatingly sampling the waiting time until 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 s, which itself varies in time. 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 Δt_i is characterized by the survival function

$$S_i(\tau) = \mathrm{P}(\Delta t_i > \tau) = \exp \int_{t_i}^{t_i+\tau} dt Q_t(x_i, x_i; s). \quad (17)$$

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

$$\ln u = \int_{t_i}^{t_i+\Delta t_i} dt Q_t(x_i, x_i; s) \quad (18)$$

for the waiting time Δt_i. Depending on the complexity of $Q_t(x_i, x_i|s)$, 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. (18) analytically, as in the example in Sec. V A. Additionally, in some cases, more efficient variants of the SSA with time-dependent rates could be used [56,57].

D. Input statistics

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

For example, the input signal may be described by a continuous-time jump process, as in Sec. V A. One benefit is that it is possible to generate exact realizations of such a process (using the SSA) and to exactly compute the likelihood $\mathcal{P}[x|s]$ using Eq. (15). Specifically, the likelihood can be exactly evaluated because the transition rates $Q_t(\cdot, \cdot; s)$ for any input trajectory s, while time dependent, are piecewise constant. This implies that the integral in Eq. (15) can be evaluated analytically without approximations. Similarly, for piecewise constant transition rates, the inverse function of Eq. (18) can be evaluated directly such that we can sample exact trajectories from the driven jump process. As a result, when both the input and output are described by a master equation, PWS is a completely exact Monte Carlo scheme used to compute the mutual information.

However, the techniques described here do *not* require the input signal 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 [58]. The application in Sec. V B provides an example.

III. VARIANTS OF PWS

The DPWS scheme presented in the previous section makes it possible to compute the mutual information between trajectories of a stochastic input process and the output of a Markov jump process. However, the number of possible trajectories increases exponentially with trajectory length, leading to a corresponding increase in the variance of the DPWS estimate. Hence, for long trajectories, the DPWS estimate may prove to be computationally too expensive. To address this issue, we describe two improved variants of PWS in this section, both based on free-energy estimators from statistical physics.

A. Marginalization integrals in trajectory space

The computationally most expensive part of our scheme in Sec. II B is the evaluation of the marginalization integral $\mathcal{P}[x_i] = \int \mathcal{D}[s]\mathcal{P}[s, x_i]$ which needs to be performed for every sample $x_1, \ldots, x_N$. 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 $\mathcal{P}[x]$ from $\mathcal{P}[s, x]$ by computing the integral

$$\mathcal{P}[x] = \int \mathcal{D}[s]\mathcal{P}[s, x] = \int \mathcal{D}[s]\mathcal{P}[s]\mathcal{P}[x|s]. \quad (19)$$

In DPWS, we use Eq. (8) to compute $\mathcal{P}[x]$, which involves generating independent input trajectories from $\mathcal{P}[s]$. However, this is not the optimal Monte Carlo technique to perform the marginalization. The generated input trajectories are independent from the output trajectory x. Thus, we ignore the causal connection between s and x, and we typically end up sampling trajectories $s^\star$ whose likelihoods $\mathcal{P}[x|s^\star]$ 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 $\mathcal{P}[s]$ 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, perhaps paradoxically, that this sampling scheme works best when the mutual information is not too large [59].

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.

To understand how we can make use of these ideas to compute the marginal probability $\mathcal{P}[x]$, it is convenient to rephrase the marginalization integral in Eq. (19) in the language of statistical physics. In this language, $\mathcal{P}[x]$ corresponds to the normalization constant, or partition function, of a Boltzmann distribution for the potential

$$\mathcal{U}[s,x] = -\ln \mathcal{P}[s,x]. \qquad (20)$$

In Eq. (20), we interpret s as a variable in the configuration space, whereas x is an auxiliary variable, i.e., a parameter. Note that both s and x still represent trajectories. For this potential, the partition function is given by

$$\mathcal{Z}[x] = \int \mathcal{D}[s]e^{-\mathcal{U}[s,x]}. \qquad (21)$$

The integral only runs over the configuration space; i.e., we integrate only with respect to s but not x, which remains a parameter of the partition function. The partition function is precisely equal to the marginal probability of the output, i.e., $\mathcal{Z}[x] = \mathcal{P}[x]$, as can be verified by inserting the expression for the $\mathcal{U}[s,x]$. Further, the free energy is given by

$$\mathcal{F}[x] = -\ln \mathcal{Z}[x] = -\ln \mathcal{P}[x], \qquad (22)$$

which shows that the computation of the free energy of the trajectory ensemble corresponding to $\mathcal{U}[s,x]$ is equivalent to the computation of (the logarithm of) the marginal probability $\mathcal{P}[x]$.

Note that, above, we omitted any factors of $k_B T$ since temperature is irrelevant here. Also note that while the distribution $\exp(-\mathcal{U}[s,x])$ looks like the equilibrium distribution of a canonical ensemble from statistical mechanics, this does not imply that we can only study systems in thermal equilibrium. Indeed, PWS is used to study information transmission in systems driven out of equilibrium by the input signal. Thus, the notation introduced in this section is nothing but a mathematical reformulation of the marginalization integral used to make the analogy to statistical physics apparent, and we assign no additional meaning of the potentials and free energies introduced 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

$$\Delta \mathcal{F}[x] = \mathcal{F}[x] - \mathcal{F}_0[x] = -\ln \frac{\mathcal{Z}[x]}{\mathcal{Z}_0[x]} \qquad (23)$$

between the system and a reference system with known free energy $\mathcal{F}_0[x]$. The reference system is described by the potential $\mathcal{U}_0[s,x]$, with the corresponding partition function $\mathcal{Z}_0[x]$. In our case, a natural choice of reference potential is

$$\mathcal{U}_0[s,x] = -\ln \mathcal{P}[s], \qquad (24)$$

with the corresponding partition function

$$\mathcal{Z}_0[x] = \int \mathcal{D}[s]\mathcal{P}[s] = 1. \qquad (25)$$

Thus, since $\mathcal{P}[s]$ is a normalized probability density function, the reference free energy is zero ($\mathcal{F}_0[x] = -\ln \mathcal{Z}_0[x] = 0$). Hence, for the above choice of reference system, the free-energy difference is

$$\Delta \mathcal{F}[x] = \mathcal{F}[x] = -\ln \mathcal{P}[x]. \qquad (26)$$

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

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

$$\mathcal{U}[s,x] = \mathcal{U}_0[s] + \Delta \mathcal{U}[s,x]. \qquad (27)$$

The interaction potential makes it apparent that the distribution of s corresponding to the potential $\mathcal{U}[s,x]$ is biased by x with respect to the distribution corresponding to the reference potential $\mathcal{U}_0[s]$. By inserting the expressions for $\mathcal{U}_0[s]$ and $\mathcal{U}[s,x]$ into Eq. (27), we see that

$$\Delta \mathcal{U}[s,x] = -\ln \mathcal{P}[x|s]$$
$$= -\ln \mathrm{P}(x_0|s_0) - \int_0^T dt \mathcal{L}_t[s,x], \qquad (28)$$

where $\mathcal{L}_t[s,x]$ is given by Eq. (15). This expression illustrates that the interaction of the output trajectory x with the ensemble of input trajectories is characterized by the trajectory likelihood $\mathcal{P}[x|s]$. Because we can compute the trajectory likelihood from the master equation, we can compute the interaction potential.

In this section, we have introduced notation (summarized in Table I) to show that computing a marginalization integral is equivalent to the computation of a free-energy difference. This picture allows us to distinguish two input trajectory ensembles, the noninteracting ensemble distributed according to $\exp(-\mathcal{U}_0[s]) = \mathcal{P}[s]$ and the interacting ensemble with input distribution proportional to $\exp(-\mathcal{U}[s,x]) \propto \mathcal{P}[s|x]$. For example, the brute-force estimate of $\mathcal{P}[x]$ used in DPWS can be written as

TABLE I. Translation to the notation of statistical physics. The definitions of $\mathcal{U}$ and $\mathcal{U}_0$ that are used here are given in Eqs. (20) and (24).

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

$$\mathcal{P}[\boldsymbol{x}] = \frac{\mathcal{Z}[\boldsymbol{x}]}{\mathcal{Z}_0[\boldsymbol{x}]} = \langle e^{-\Delta\mathcal{U}[\boldsymbol{s}, \boldsymbol{x}]}\rangle_0, \quad (29)$$

where the notation $\langle\cdots\rangle_0$ refers to an average with respect to the noninteracting 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. (8). As explained in Sec. II B, to compute Eq. (29) using Monte Carlo, it is only necessary to sample from the noninteracting system $\mathcal{U}_0[\boldsymbol{s}]$ and to compute the Boltzmann weight $\Delta\mathcal{U}[\boldsymbol{s}, \boldsymbol{x}]$ (i.e., to sample from $\mathcal{P}[\boldsymbol{s}]$ and to compute the log-likelihood $\ln\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}]$). This is indeed the DPWS scheme. However, by noting the correspondence between signal trajectories and polymers and that Eq. (23) has the same form as the expression for the (excess) chemical potential of a polymer, which is the free-energy difference between the polymer of interest and the ideal chain [43,60], more efficient schemes can be developed, as we show next.

B. 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 given by Rosenbluth and Rosenbluth [61] 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 [44]. 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 spirit to PERM but requires less tuning, the bootstrap particle filter [62]. 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 [63]. In addition, schemes inspired by variants of forward flux sampling [64,65] could be developed.

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 introduce 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 τ_i and τ_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 $P(s_0)$.
(2) Iterate for $i = 1, \ldots, n$:
   (a) Starting with an ensemble of M partial trajectories of duration τ_{i-1} (if $i = 1$, an ensemble of starting points), which we label $\boldsymbol{s}_{[0,i-1]}^k$ for $k = 1, \ldots, M$:

$$\left(\boldsymbol{s}_{[0,i-1]}^1, \ldots, \boldsymbol{s}_{[0,i-1]}^M\right), \quad (30)$$

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

$$p_i^{\text{gen}}(k) = \mathcal{P}\left[\boldsymbol{s}_{[i-1,i]}^k\big|\boldsymbol{s}_{[0,i-1]}^k\right] = e^{-\mathcal{U}_0[\boldsymbol{s}_{[i-1,i]}^k]} \quad (31)$$

   for $k = 1, \ldots, M$.
   (b) Compute the Boltzmann weight

$$U_i^k = \Delta\mathcal{U}\left[\boldsymbol{s}_{[i-1,i]}^k, \boldsymbol{x}_{[i-1,i]}\right] \quad (32)$$

   of each new segment. This Boltzmann weight of a segment from τ_{i-1} to τ_i can be expressed as

$$U_i^k = -\delta_{1i}\ln P(x_0|s_0) - \int_{\tau_{i-1}}^{\tau_i} dt \mathcal{L}_t\left[\boldsymbol{s}_{[i-1,i]}^k, \boldsymbol{x}_{[i-1,i]}\right] \quad (33)$$

   [see Eq. (28)] and is therefore straightforward to compute from the master equation.
   (c) Sample M times from the distribution

$$p_i^{\text{select}}(k) = \frac{e^{-U_i^k}}{w_i}, \quad (34)$$

where the Rosenbluth weight w_i is defined as

$$w_i = \sum_{k=1}^{M} e^{-U_i^k}. \tag{35}$$

This sampling procedure yields M randomly drawn indices $\ell_i^1, \ldots, \ell_i^M$. Each ℓ_i^k is an index that lies in the range $1, \ldots, M$ and that points to one of the trajectories that has been generated up to τ_i. To continue the sampling procedure, we relabel the indices such that the resampled set of trajectories is defined by $\tilde{s}_{[0,i]}^k \leftarrow s_{[0,i]}^{\ell_i^k}$ for $k = 1, \ldots, M$. The list $(\tilde{s}_{[0,i]}^1, \ldots, \tilde{s}_{[0,i]}^M)$ 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

$$\mathcal{W} = \prod_{i=1}^{n} \frac{w_i}{M}. \tag{36}$$

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

$$\hat{\mathcal{P}}[x] = \mathrm{P}(x_0)\mathcal{W}, \tag{37}$$

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 $\mathcal{P}[x]$, substituting the brute-force estimate given in Eq. (8). We call the resulting algorithm used to compute the mutual information RR-PWS.

We now provide an intuitive explanation for the scheme presented above. First, note that steps 1 and 2(a) of the procedure above only involve propagating M trajectories in parallel, according to $\mathcal{P}[s] = \exp(-\mathcal{U}_0[s])$. The interesting

steps are 2(b) and 2(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 $(\ell_i^1, \ldots, \ell_i^M)$ 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 $\ell_i^1, \ldots, \ell_i^M$ may not include every original index $1, \ldots, M$, therefore eliminating some trajectories. Since indices of trajectories with high Boltzmann weight are more likely to be sampled from Eq. (34), this ensures that we are only spending computational effort on propagating those trajectories whose Boltzmann weight is not too small. Hence, at its heart, the particle filter is an algorithm for producing samples that tend to be distributed according to $\exp(-\mathcal{U}_0[s])\exp(-\Delta\mathcal{U}[s,x]) = \exp(-\mathcal{U}[s,x])$, i.e., according to the Boltzmann distribution of the *interacting* ensemble; see also Appendix A. For illustration of the algorithm, one iteration of the particle filter is presented schematically in Fig. 3.

For the efficiency of the particle filter, it is important to carefully choose the number of segments n. When the segments are very short (n large), the accumulated weights [Eq. (33)] tend to differ very little between the newly generated segments $s_{[i-1,i]}^k$. 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_i^k becomes very wide. Then, only a few segments contribute substantially to the corresponding Rosenbluth weight w_i. Hence, to carefully choose n, we need a measure that quantifies the variance of the trajectory weights. To this end, we follow Martino *et al.* [66] and introduce an effective sample size (ESS)

$$M_i^{(\mathrm{eff})} = \frac{w_i^2}{\sum_{k=1}^{M} (e^{-U_i^k})^2}, \tag{38}$$

which lies in the range $1 \leq M_i^{(\mathrm{eff})} \leq M$; $M_i^{(\mathrm{eff})} = 1$ if one trajectory has a much higher weight than all the others, and
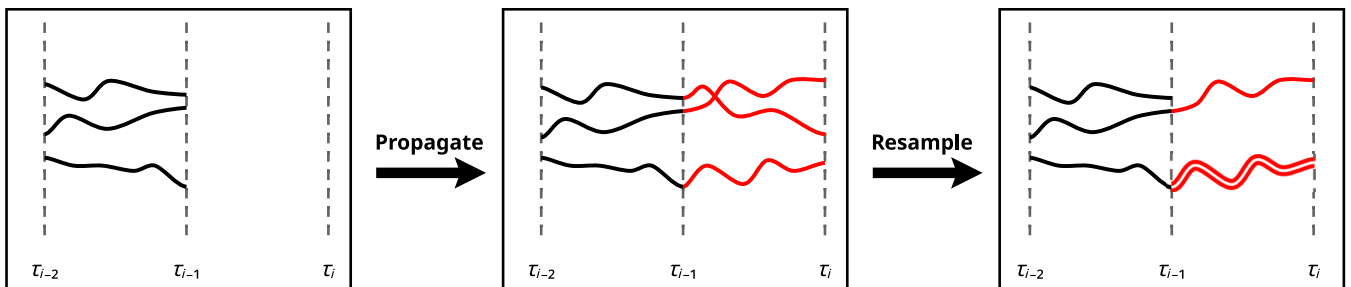

FIG. 3. Illustration of one step of the bootstrap particle filter in RR-PWS. We start with a set of trajectories $s_{[0,i-1]}^k$ with time span $[\tau_0, \tau_{i-1}]$ (left panel). In the next step, we propagate these trajectories forward in time to τ_i, according to $\mathcal{P}[s]$ (central panel). 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 $\mathcal{P}[s]$ towards $\mathcal{P}[s|x]$.

$M_i^{(\text{eff})} = M$ if all trajectories have the same weight. As a rule of thumb, we resample only when the $M_i^{(\text{eff})}$ drops below $M/2$. Additionally, as recommended in Ref. [67], we use the systematic sampling algorithm to randomly draw the indices in step 2(c), which helps to reduce the variance; we find, however, that 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.

C. TI-PWS

Our third scheme, 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

$$\mathcal{Z}_\theta[\boldsymbol{x}] = \int \mathcal{D}[\boldsymbol{s}] e^{-\mathcal{U}_\theta[\boldsymbol{s}, \boldsymbol{x}]}. \tag{39}$$

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

$$\mathcal{U}_\theta[\boldsymbol{s}, \boldsymbol{x}] = \mathcal{U}_0[\boldsymbol{s}, \boldsymbol{x}] + \theta \Delta \mathcal{U}[\boldsymbol{s}, \boldsymbol{x}], \tag{40}$$

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 [47]. Here, we do not consider other paths however, and we derive the thermodynamic integration estimate for the potential given in Eq. (40).

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 θ:

$$\begin{aligned}
\frac{\partial}{\partial \theta} \ln \mathcal{Z}_\theta[\boldsymbol{x}] &= \frac{1}{\mathcal{Z}_\theta[\boldsymbol{x}]} \frac{\partial}{\partial \theta} \int \mathcal{D}[\boldsymbol{s}] e^{-\mathcal{U}_\theta[\boldsymbol{s}, \boldsymbol{x}]} \\
&= -\left\langle \frac{\partial \mathcal{U}_\theta[\boldsymbol{s}, \boldsymbol{x}]}{\partial \theta} \right\rangle_\theta \\
&= -\langle \Delta \mathcal{U}[\boldsymbol{s}, \boldsymbol{x}] \rangle_\theta.
\end{aligned} \tag{41}$$

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

$$\mathcal{P}_\theta[\boldsymbol{s}|\boldsymbol{x}] = \frac{1}{\mathcal{Z}_\theta[\boldsymbol{x}]} e^{-\mathcal{U}_\theta[\boldsymbol{s}, \boldsymbol{x}]}. \tag{42}$$

Integrating Eq. (41) with respect to θ leads to the formula for the free-energy difference

$$\Delta \mathcal{F}[\boldsymbol{x}] = -\int_0^1 d\theta \langle \Delta \mathcal{U}[\boldsymbol{s}, \boldsymbol{x}] \rangle_\theta, \tag{43}$$

which is the fundamental identity underlying thermodynamic integration.

To compute the free-energy difference using Eq. (43), we evaluate the θ integral 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) [49]. As discussed in Appendix B, the efficiency of MCMC samplers strongly depends on the trail moves that are employed. While better trail moves could be conceived, we only use the forward shooting and backward shooting moves of TPS [49] to obtain the results in Sec. V. These moves regrow either the end or the beginning of a trajectory, respectively. A move is accepted according to the Metropolis criterion [68].

IV. 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 Ω. 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, $X^1, X^2, \ldots, X^n$. Each component X^i itself has a corresponding trajectory $\boldsymbol{x}^i$, such that the full trajectory can be represented as a vector $\boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n)$. Indeed, it is also the conditional probability $\mathcal{P}[\boldsymbol{x}|\boldsymbol{s}] = \mathcal{P}[\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n|\boldsymbol{s}]$ and the marginal probability $\mathcal{P}[\boldsymbol{x}] = \mathcal{P}[\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n]$ 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 $X^1, \ldots, X^n$. 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 $X^1, \ldots, X^n$. Without loss of generality, we assume that the nth component is the output component, $X^r = X^n$. The other species $X^1, \ldots, X^{n-1}$ are thus not part of the output but only relay information from the input signal s to the output signal x^n. To determine the mutual information $I(\mathcal{S}, \mathcal{X})$, we need $\mathcal{P}[x^n|s]$, where 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 $\mathcal{P}[x^1, \ldots, x^n|s]$ of all components. To compute the value of $\mathcal{P}[x^n|s]$, we must perform the marginalization integral

$$\mathcal{P}[x^n|s] = \int \mathcal{D}[x^1] \cdots \int \mathcal{D}[x^{n-1}] \mathcal{P}[x^1, \ldots, x^n|s]. \quad (44)$$

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

The marginalization of Eq. (44) entails integrating out degrees of freedom from a known joint probability distribution. In Eq. (8), we solved the analogous problem of obtaining the marginal probability $\mathcal{P}[x]$ by integrating out the input trajectories through the integral $\mathcal{P}[x] = \int ds \mathcal{P}[s, x] = \int ds \mathcal{P}[s] \mathcal{P}[x|s]$. As described in Sec. II B, the integral from Eq. (8) can be computed via a Monte Carlo estimate by sampling many input trajectories from $\mathcal{P}[s]$ and taking the average of the corresponding conditional probabilities $\mathcal{P}[x|s_i]$. We 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. (44). We describe this below. Additionally, in the absence of feedback, the techniques presented in Sec. III above can be employed to develop computationally more efficient schemes.

More specifically, we can evaluate Eq. (44) via a direct Monte Carlo estimate under the condition that the stochastic dynamics of the other components $X^1, \ldots, X^{n-1}$ are not influenced by X^n (i.e., no feedback from the readout). Using the identity

$$\mathcal{P}[x^1, \ldots, x^n|s] = \mathcal{P}[x^1, \ldots, x^{n-1}|s] \mathcal{P}[x^n|x_i^1, \ldots, x_i^{n-1}, s] \quad (45)$$

to rewrite the integrand in Eq. (44), we are able to represent the conditional probability $\mathcal{P}[x^n|s]$ as an average over the readout component's trajectory probability

$$\mathcal{P}[x^n|s] = \langle \mathcal{P}[x^n|x_i^1, \ldots, x_i^{n-1}, s] \rangle_{\mathcal{P}[x^1, \ldots, x^{n-1}|s]}. \quad (46)$$

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

$$\mathcal{P}[x^n|s] \approx \frac{1}{M} \sum_{i=1}^{M} \mathcal{P}[x^n|x_i^1, \ldots, x_i^{n-1}, s], \quad (47)$$

where the samples $x_i^1, \ldots, x_i^{n-1}$ for $i = 1, \ldots, M$ are drawn from $\mathcal{P}[x^1, \ldots, x^{n-1}|s]$. Notice that the derivation of this Monte Carlo estimate is fully analogous to the estimate in Eq. (8), but instead of integrating out the input trajectory s, we integrate out the component trajectories $x^1, \ldots, x^{n-1}$.

To obtain $\mathcal{P}[x^n|x_i^1, \ldots, x_i^{n-1}, s]$ in Eqs. (46) and (47), we note that, in the 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 $x^1, \ldots, x^{n-1}$ and the input s. In effect, this is a driven jump process for X^n, driven by all upstream components $X^1, \ldots, X^{n-1}$ and the input signal. Specifically, denoting $u = (x^1, \ldots, x^{n-1}, s)$ as the joint trajectory representing the history of all upstream components as well as the input signal, we can, as explained in Sec. II C, write the time-dependent transition rate matrix $Q_t(\cdot|u)$ for the stochastic dynamics of X^n and use Eq. (14) to compute $\mathcal{P}[x^n|u] = \mathcal{P}[x^n|x_i^1, \ldots, x_i^{n-1}, s]$. Using Eq. (47), this then allows us to compute $\mathcal{P}[x^n|s]$.

Finally, to compute the mutual information $I(\mathcal{S}; \mathcal{X}^n)$, e.g., using the estimate in Eq. (10), we additionally need to evaluate the marginal output probability $\mathcal{P}[x^n]$. This requires us to perform one additional integration over the space of input trajectories s:

$$\mathcal{P}[x^n] = \int \mathcal{D}[s] \mathcal{P}[s] \mathcal{P}[x^n|s]$$
$$= \langle \mathcal{P}[x^n|s] \rangle_{\mathcal{P}[s]}. \quad (48)$$

The corresponding Monte Carlo estimate is

$$\mathcal{P}[x^n] \approx \frac{1}{N} \sum_{i=1}^{N} \mathcal{P}[x^n|s_i]$$
$$\approx \frac{1}{N} \sum_{i=1}^{N} \frac{1}{M} \sum_{j=1}^{M} \mathcal{P}[x^n|x_{ij}^1, \ldots, x_{ij}^{n-1}, s_i], \quad (49)$$

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

In summary, the scheme to obtain $\mathcal{P}[x^n|u]$ in the presence of hidden intermediate components is analogous to that used for computing $\mathcal{P}[x]$ from $\mathcal{P}[x|s]$. In both cases, one needs to marginalize a distribution function by integrating out components. Indeed, the schemes presented here and in Sec. II B are bona fide schemes to compute the mutual information between the input s and either the trajectory of the output component x^n or the full output x. However, when the trajectories are sufficiently long or the

stochastic dynamics are sufficiently complex, the free-energy schemes of Sec. III may be necessary to enhance the efficiency of computing the marginalized distribution, $\mathcal{P}[\boldsymbol{x}]$ or $\mathcal{P}[\boldsymbol{x}^n|\boldsymbol{s}]$.

V. RESULTS

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 two instructive chemical reaction networks. We first consider a linearly coupled birth-death process. This system has already been studied previously using a Gaussian model [6], and by Duso and Zechner [30] using an approximate technique, and we compare our results to the results of 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 second example has been chosen to demonstrate the practical applicability of our technique. We use RR-PWS to compute the mutual information rate in the bacterial chemotaxis system, which is a prime example of a complex information processing system consisting of many reactions. Then, we compare the computed rate against recent experiments.

The code used to produce the PWS estimates was written in the Julia programming language [69] and has been made freely available [70]. For performing stochastic simulations, we use the DifferentialEquations.jl package [71], and biochemical reaction models are set up with help from the ModelingToolkit.jl package [72].

A. Coupled birth-death processes

As a first example system, we consider a simple birth-death process $\varnothing \rightleftharpoons X$ of species X, which is created at rate $\rho(t)$ and decays with constant rate μ 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

$$\rho(t) = \rho_0 s(t), \qquad (50)$$

where ρ_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 $\varnothing \rightleftharpoons S$ with production rate κ and decay rate λ.

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 a
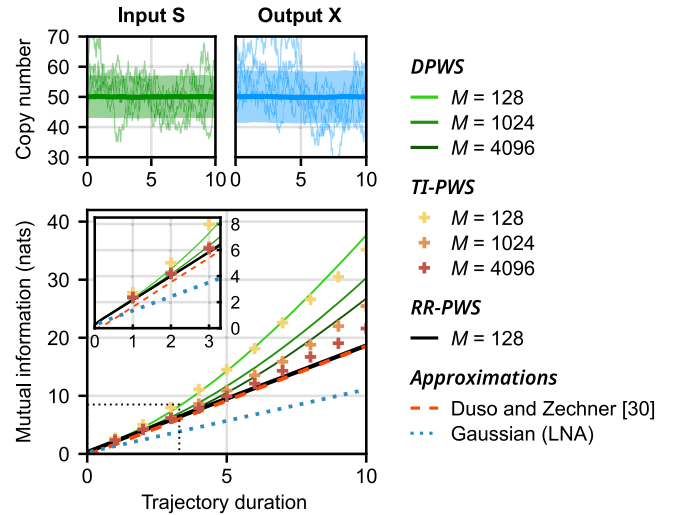

FIG. 4.   Comparison of different schemes to compute the mutual information as a function of trajectory duration for a simple coupled birth-death process with rates $\kappa = 50, \lambda = 1, \rho_0 = 10$, $\mu = 10$ and a steady-state initial condition. The top panels show example trajectories of the input and output as well as the mean (solid line) and standard deviation (shaded region). The lower panels show the mutual information 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 for convergence. Results for the three PWS variants are compared with the Duso and Zechner [30] estimate and with the linear noise approximation from Ref. [6]. 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 $\mathcal{P}[\boldsymbol{s}, \boldsymbol{x}]$.

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 it leads to nonzero mutual information already for $T = 0$.

Figure 4 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 [30]. To apply it, we used the code publicly provided by the authors [73], 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 [6], obtained using the linear-noise approximation (see Appendix F).

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 $\mathcal{P}[x]$. For longer trajectories, the estimate becomes increasingly dominated by rare trajectories, which make an exceptionally large contribution to the average of $\mathcal{P}[x]$. Missing these rare trajectories with a high weight tends to increase the marginal entropy $\mathrm{H}(\mathcal{X})$ [see Eq. (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 $\mathcal{P}[x]$. Similarly, for TI-PWS, the error decreases as we use more MCMC samples for the marginalization scheme. For the RR-PWS, however, already for $M = 128$ the estimate has converged; we verify 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 [30]. Only very small deviations are visible in Fig. 4. 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.

Figure 4 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. We leave a detailed analysis of this observation for future work.

The different approaches for computing the marginal probability $\mathcal{P}[x]$ lead to different computational efficiencies of the respective PWS schemes. In Fig. 5, 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 makes it possible to generate input trajectories s that are correlated with the output trajectories 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 s with nonvanishing likelihood $\mathcal{P}[x|s]$, thereby avoiding the sampling problem from DPWS. Moreover, sequential
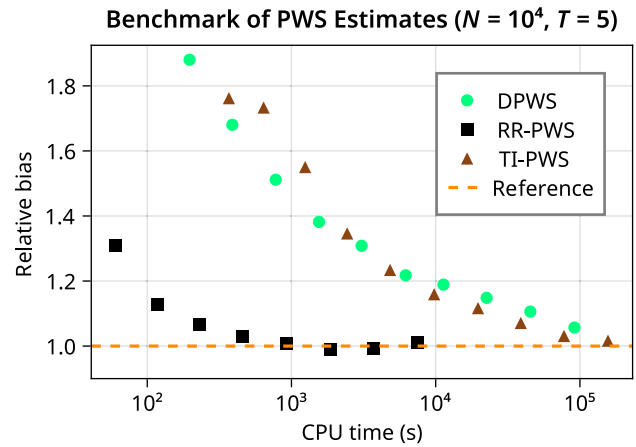

FIG. 5. Comparing estimation bias for the different PWS variants in relation to their CPU time requirements. Each dot represents a single mutual information estimate with $N = 10^4$ samples for trajectories of duration $T = 5$. Almost all the CPU time of a PWS estimate is spent on the computation of the marginal probability $\mathcal{P}[x]$. 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 represent estimates ranging from $M = 2^5$ to $M = 2^{14}$, and 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.

Monte Carlo techniques such as RR-PWS and FFS [64] have a considerable advantage over MCMC techniques in trajectory 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.

The coupled birth-death process represents a simple yet nontrivial system capable of information transmission. In the next section, we apply PWS to a more complex and realistic biochemical signaling network.

B. Bacterial chemotaxis

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. Briefly, *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 [74]. The rates of switching between these two states are controlled by the chemotaxis sensing system [Fig. 6(a)]. This system consists of receptors on the cell surface that detect the
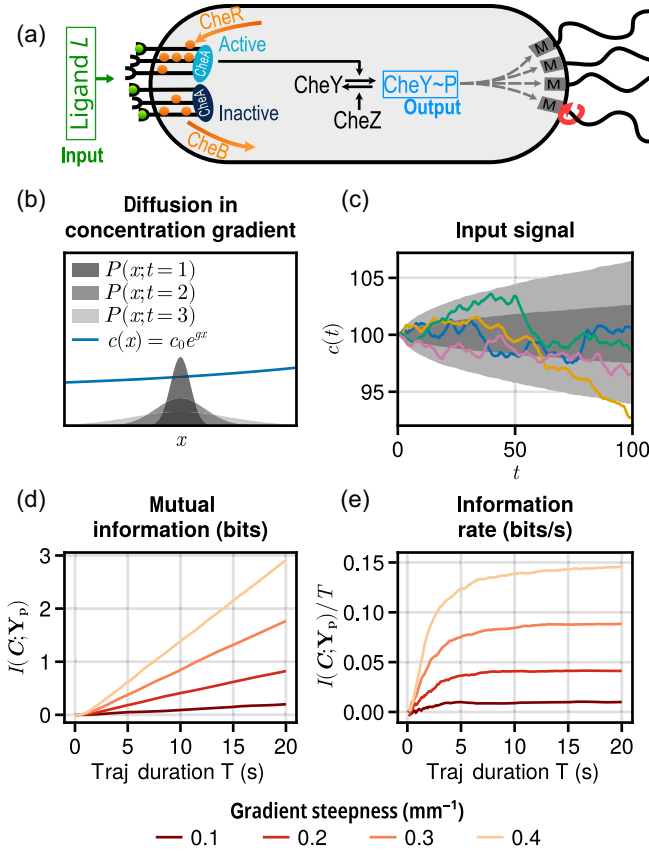
FIG. 6. Information transmission rate in the bacterial chemotaxis system. (a) Cartoon of the chemotaxis network of *E. coli*. Receptors form clusters with an associated CheA kinase. A cluster can either be active or inactive, depending on the number of bound ligands (green dots) and methylated sites (orange dots). Active CheA can phosphorylate CheY; phosphorylated CheY controls the rotation direction of the flagellar motors and thereby the movement of the bacterium. (b) Bacterium diffusing nearly freely in the x direction in a shallow gradient $c(x)$. 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. (c) Movement giving rise to the input statistics of the signal. The shaded regions indicate the 75th and 95th percentiles; example trajectories are displayed in color. This signal is nonstationary as its variance always keeps growing. (d) 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). (e) Information transmission rate defined as $I(\mathbf{C}_T, \mathbf{Y}_T)/T$ in the limit $T \to \infty$.

ligand and a downstream signaling network that processes this information by taking the time derivative of the signal, the ligand concentration. This derivative is taken via two antagonistic reactions, which occur on two distinct timescales. Attractant binding rapidly deactivates the receptor, while slow methylation counteracts this reaction by reactivating the receptor, leading to near perfect

adaptation [75–78]. Lastly, active receptors phosphorylate the downstream messenger protein CheY, which controls the tumbling propensity by binding the flagellar motors that propel the bacterium.

In our model, the receptors are grouped in clusters (Appendix D). Each receptor can switch between an active and an inactive conformational state, but, in the spirit of the Monod-Wyman-Changeux model [79], 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 molecules and be (de)methylated, and thus can 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 (Appendix D). 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 that drive (de)phosphorylation are not modeled explicitly. Table II in Appendix D gives 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. [80,81], 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. [82]. The translation of our model into a master equation is explained in Appendix D.

We first asked whether this model based on the current literature can reproduce the information transmission rate as recently measured by Mattingly *et al.* [50]. In what follows, we call this model the "literature-based" model.

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.* [50]. In our model, the input signal is the time-dependent ligand concentration $c(t)$ that is experienced by the swimming bacterium. In the experiments, the cells swim in a very shallow chemical gradient, such that their swimming behavior is, to a good approximation, identical to that in the absence of a gradient. The cell's movement can thus be modeled as a (persistent) random walk. Assuming that the gradient is oriented in the x direction, the autocorrelation of the x component of the velocity was experimentally found to be well described by $V(t) \equiv \langle \delta v_x(t) \delta v_x(0) \rangle \simeq \sigma_v^2 e^{-\lambda|t|}$, where σ_v^2 is the variance of the fluctuations in the velocity, and λ^{-1} is the correlation

time of these fluctuations [50]. We therefore model the cell's velocity as $\dot{v}_x = -\lambda v_x + \xi$, which, with $\langle \xi(t)\xi(t') \rangle = 2\sigma_v^2 \lambda \delta(t - t')$, gives rise to the measured correlation function $V(t)$. These cells swim in a shallow, exponential gradient $c(x) \propto e^{gx}$ with steepness g. The dynamics of the ligand concentration as experienced by the bacteria are then given by $\dot{c} = gcv_x$. The cell's own swimming dynamics described by v_x thus give rise to the input signal $c(t)$ of the biochemical network. It constitutes an exponential random walk, as illustrated inFigs. 6(b) and 6(c) (see also Appendix E).

In our model, the output is the concentration of phosphorylated CheY, while in the experiments of Mattingly *et al.* [50], 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, $Yp(t)$, 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 $Y_p(t)$ 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.* [50].

We use RR-PWS to exactly 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, $y_p(t)$, 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 Sec. III B. The receptor states are hidden internal states, and we use the technique of Sec. IV to integrate them out.

Figure 6(e) shows the PWS estimate of the information transmission rate for cells swimming in gradients of different steepness g. The information transmission rate is obtained from the PWS estimate of the trajectory mutual information $I(\mathbf{C}, \mathbf{Y_p}; T)$, with different trajectory durations T. As seen in Fig. 6(d), for short trajectories, the mutual information increases nonlinearly 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

$$R(\mathbf{C}, \mathbf{Y_p}) = \lim_{T \to \infty} \frac{I(\mathbf{C}, \mathbf{Y_p}; T)}{T}. \qquad (51)$$

We then compare our results for the information transmission rate of the literature-based model to those of Mattingly *et al.* [50]. Figure 7(c) shows that the model predictions differ from the experiments by a factor of approximately 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.* [50].

However, 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.* [50] had to employ to compute the information transmission rate from experimental data. Concerning the latter hypothesis, because of the curse of dimensionality and experimental constraints, Mattingly *et al.* [50] 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 $N(t)$ [6,50] (see also Appendix F). This Gaussian model is based on a linear noise assumption and cannot perfectly capture the true nonlinear dynamics of the biochemical network. This could be the cause for the observed discrepancies in the information rate. We have indeed already seen in Sec. V A that there can be substantial differences between exact computations and the Gaussian approximation for the trajectory mutual information.

To uncover the reason for the discrepancy, we first test 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.* [50]. However, we find that the response kernel $K(t)$ and the autocorrelation function of the noise $N(t)$ of our system are different. Figures 7(a) and 7(b) show that our model reproduces the timescales of $N(t)$ and $K(t)$ as measured
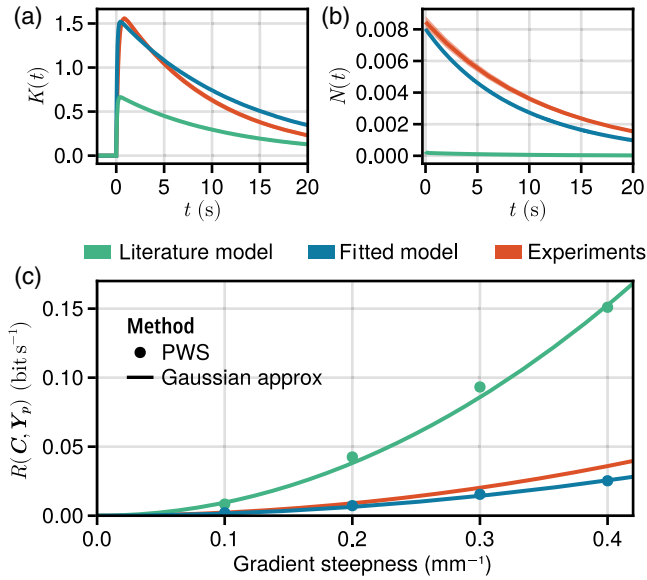
FIG. 7. Comparison of theoretical models with experimental data for bacterial chemotaxis system. Panels (a) and (b) show the response and noise kernels, respectively, for the model based on literature parameters (green), parameters fitted to experiments (blue), and experiments from Mattingly *et al.* [50] (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 ≈ 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 by Mattingly *et al.* [50].

experimentally. This result 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 σ_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 σ_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. 7(b) 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 [50].

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 [80,85], but the number of receptors per cluster has only been inferred indirectly from experiments [80,81]. The higher gain as measured experimentally by Mattingly *et al.* [50] indicates that N is larger than assumed in our model: With $N = 15$, our model can quantitatively fit $K(t)$ [Fig. 7(a)].

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 to quantitatively fit both $N(t)$ and $K(t)$ [50] (Fig. 7). This result suggests that the number of independent receptor clusters is smaller than hitherto believed while their size is larger.

Finally, 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.* [50]? 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.* [50]; see Fig. 7. Within the Gaussian framework (see Appendix F), the information transmission rate in our model is thus expected to be very similar to the experimentally measured one, and Fig. 7(c) shows that this is indeed the case. To quantify the accuracy of the Gaussian framework, we then recompute the information transmission rate for the revised model, using exact PWS (see Appendix H). We find 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.* [50].

VI. DISCUSSION

In this paper, we have developed 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.

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 [50] 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 co-workers in analyzing the data. In contrast, because PWS is exact, we knew the difference between theory and experiment was 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 was 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, $N \sim 6$ [80,81,86]. The early estimates of the cluster size were based on bulk dose-response measurements with a relatively slow ligand exchange, yielding $N \approx 6$ [80,86]. 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 [81]. Our estimate $N \approx 15$, based on fitting the response kernel $K(t)$ to that measured by Mattingly *et al.* [50], 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.* [50], is surprisingly low, $N_c \sim 10$, given the total number of receptors, $N_r \sim 10^3$–10^4 [82]. Interestingly, recent experiments indicate that the receptor array is poised near the critical point [87], 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 [21,88], although it has been argued that the sensing error of the *E. coli* chemotaxis system is minimized for independent receptors [89]. Mattingly *et al.* [50] have demonstrated that the chemotactic drift speed in shallow exponential gradients is limited by the information transmission rate [50], but whether the system has been optimized for information transmission and how the latter affects chemotactic performance in other spatiotemporal concentration profiles remain interesting questions for future work.

While we have focused on the computation of the mutual information between trajectories of systems governed by a master equation, this concept can be extended to other types of stochastic processes. The crux of PWS is the exact evaluation of the path likelihood $\mathcal{P}[x|s]$ from the master equation. Therefore, in order to use PWS with a different stochastic process, we require the ability to compute the path likelihood $\mathcal{P}[x|s]$ for its sample paths. Remarkably, for stochastic diffusion processes, which are described by a Langevin equation in a continuous state space, the trajectory probability can be computed by replacing the kernel $\mathcal{L}_t(s, x)$ in Eq. (15) with the Onsager-Machlup function [90]. In particular, while the path probability is not well defined for continuous sample paths, Adib [91] shows that the Onsager-Machlup function yields a correct expression for the path probability of time-discretized diffusion trajectories. Therefore, PWS can be extended to handle systems described by a Langevin equation. In such a PWS scheme, the Gillespie simulations are replaced by standard numerical integration techniques for Langevin equations (see, e.g., Ref. [58]), and the trajectory likelihood is evaluated on the fly using the Onsager-Machlup function.

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-neighbor estimators [23,24], decoding-based information estimates [26], or schemes that compute the mutual information from the data within the Gaussian framework [50]. PWS requires a (generative) model based on a master equation or Langevin equation. However, an increasingly popular approach is to estimate the mutual information from experimental data using hidden Markov models (HMM) [92–94]. While the construction of these HMM models is beyond the scope of this paper, PWS makes it possible to compute the mutual information between time-varying inputs and outputs exactly for these models.

We have applied PWS to compute the mutual information (rate) in the steady state, but PWS can be used equally well to study systems out of the steady state. For such systems, a (non)equilibrium initial condition $P(s_0, x_0)$ must be specified in addition to a well-defined nonstationary probability distribution of input trajectories $\mathcal{P}[s]$. These distributions are defined by the (experimental) setup and lead to a well-defined output distribution $\mathcal{P}[x]$ when the system is coupled to the input. Thus, a steady state is not a prerequisite for the application of PWS to study the trajectory mutual information.

Throughout the paper, we have considered systems in which the output does not feed back onto the input. In systems with feedback, the current output influences future input, which means that we cannot straightforwardly generate input trajectories according to $\mathcal{P}[s]$. Moreover, $\mathcal{P}[x|s]$, the central quantity of all three PWS methods, cannot be obtained straightforwardly. Nonetheless, PWS can be extended to systems with feedback as shown in Appendix C. As in free-energy calculations, the trick is to define a reference system for which the marginal probability distribution—and thus the "free energy"—is known and then compute the "free-energy difference" between that reference system and the system of interest.

Aside from DPWS, 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 $\mathcal{P}[\boldsymbol{x}]$, which corresponds to the computation of a partition function in statistical physics. RR-PWS and TI-PWS are based on techniques from polymer and rare-event sampling to make the computation of the marginal trajectory probability more efficient.

The different PWS variants share some characteristics yet 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 [44] and rare-event methods like DFFS or BG-FFS [64]. 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 [95] and the TPS [49], TIS [96], and RB-FFS [64] 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 [97].

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 a 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 τ. 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 input trajectory a time τ 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 τ earlier. In contrast, DPWS and TI-PWS generate full trajectories one at a time and are therefore more likely to capture these correlations.

Overall, PWS is a general framework for computing the mutual information between trajectories. We presented three variants of PWS for systems described by a master equation. In addition, we expect that other variants could be developed to improve efficiency for particular applications. Because of its flexibility and simplicity, we envision that PWS will become an important and reliable tool for studying information transmission in dynamic stochastic systems.

APPENDIX A: JUSTIFICATION OF THE PARTICLE FILTER FROM SEC. III B

Here, we justify the marginal probability estimate shown in Eq. (37); i.e., we show that the bootstrap particle filter used in Sec. III B provides a consistent estimate for the ratio of partition functions $\mathcal{P}[\boldsymbol{x}] = \mathcal{Z}[\boldsymbol{x}]/\mathcal{Z}_0[\boldsymbol{x}]$. The result that this estimate is also *unbiased* is more difficult to establish; a proof is given by Del Moral [98].

To make the derivations below easy to follow, 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 according to a target distribution when only samples from a different distribution are available. Second, we use these insights to explain how the resampling procedure used in the particle filter generates trajectories that are distributed approximately according to $\mathcal{P}[\boldsymbol{s}|\boldsymbol{x}]$, 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 $\mathcal{P}[\boldsymbol{x}]$.

1. Sampling and resampling

Sampling and then resampling is a strategy to use samples $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^M$ from a given prior distribution $f[\boldsymbol{s}]$ to generate *approximate* 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 $\hat{h}[s] = h[s] / \int \mathcal{D}[s] h[s]$. To generate samples from $\hat{h}[s]$, we assign each of the existing samples from $f[s]$ a normalized weight

$$W^k = \frac{g[s^k]}{\sum_{j=1}^{M} g[s^j]}. \qquad (A1)$$

Then, by sampling from the discrete set $\{s^1, \ldots, s^M\}$ according to the weights $W^1, \ldots, W^M$, we pick samples that are approximately distributed according to $\hat{h}[s]$. Indeed, for $M \to \infty$, the distribution of the resulting samples approaches the density $\hat{h}[s]$ [99]. We use resampling at each iteration of the algorithm of Sec. III B to regularly prune those trajectories with low overall contribution to the marginalization integral.

2. Distribution of trajectories in the particle filter

In the bootstrap particle filter, at each iteration, we start with a set of trajectories $s_{[0,i-1]}^1, \ldots, s_{[0,i-1]}^M$, which we assume are approximately distributed according to $\mathcal{P}[s_{[0,i-1]} | x_{[0,i-1]}]$. In each iteration of the particle filter, the goal is to produce a set of trajectories approximately distributed according to $\mathcal{P}[s_{[0,i]} | x_{[0,i]}]$. Clearly, by iterating such a procedure, we can generate a set of trajectories distributed approximately according to $\mathcal{P}[s_{[0,n]} | x_{[0,n]}]$ for any $n > 1$. Note that we always carefully use the phrase approximately distributed because, as explained above, for finite M, a resampling procedure cannot generate *exact* samples from a probability distribution (yet the estimate for $\mathcal{P}[x]$ remains unbiased regardless of how good these approximations are). We now take a closer look at one iteration of the particle filter.

We start with the set of trajectories with a time span $[\tau_0, \tau_{i-1}]$, denoted by $\{s_{[0,i-1]}^1, \ldots, s_{[0,i-1]}^M\}$. These trajectories are then propagated forward to time τ_i, by adding a new segment $s_{[i-1,i]}^k$ to the trajectory $s_{[0,i-1]}^k$ for $k = 1, \ldots, M$. Each new segment is generated from the distribution $\mathcal{P}[s_{[i-1,i]}^k | s_{[0,i-1]}^k]$ such that the propagation step results in a set of trajectories $\{s_{[0,i]}^1, \ldots, s_{[0,i]}^M\}$, distributed according to $f[s_{[0,i]}] = \mathcal{P}[s_{[0,i]} | x_{[0,i-1]}]$.

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

$$W_i^k = \frac{\mathcal{P}[x_{[i-1,i]} | x_{[0,i-1]}, s_{[0,i]}^k]}{\sum_{j=1}^{M} \mathcal{P}[x_{[i-1,i]} | x_{[0,i-1]}, s_{[0,i]}^j]}, \qquad (A2)$$

cf. Eq. (A1). Note that this is the same choice of weighting function as in Sec. III B, Eq. (34). 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

$$W_i^k = \frac{e^{-U_i^k}}{w_i}. \qquad (A3)$$

Why is this choice of weighting function the correct one? First, observe that resampling with the normalized weights of Eq. (A2) produces samples approximately distributed according to

$$\begin{aligned} h[s_{[0,i]}] &= f[s_{[0,i]}] g[s_{[0,i]}] \\ &= \mathcal{P}[s_{[0,i]} | x_{[0,i-1]}] \mathcal{P}[x_{[i-1,i]} | x_{[0,i-1]}, s_{[0,i]}]. \end{aligned} \qquad (A4)$$

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

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

$$g[s_{[0,i]}] = \frac{\mathcal{P}[s_{[0,i]} | x_{[0,i-1]}, x_{[i-1,i]}] \mathcal{P}[x_{[i-1,i]} | x_{[0,i-1]}]}{\mathcal{P}[s_{[0,i]} | x_{[0,i-1]}]}. \qquad (A5)$$

Notice that the first term of the numerator can be written as $\mathcal{P}[s_{[0,i]} | x_{[0,i]}]$. After inserting this result into Eq. (A4), we obtain

$$h[s_{[0,i]}] = \mathcal{P}[s_{[0,i]} | x_{[0,i]}] \mathcal{P}[x_{[i-1,i]} | x_{[0,i-1]}]. \qquad (A6)$$

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

$$\hat{h}[s_{[0,i]}] = \mathcal{P}[s_{[0,i]} | x_{[0,i]}]. \qquad (A7)$$

Consequently, this is the distribution that is approximated by the set of trajectories at the end of the ith 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 $\mathcal{P}[s|x]$.

3. Marginal probability estimate

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

$$\mathcal{P}[\mathbf{x}_{[i-1,i]}|\mathbf{x}_{[0,i-1]}] = \int \mathcal{D}[\mathbf{s}_{[0,i]}]\mathcal{P}[\mathbf{x}_{[i-1,i]},\mathbf{s}_{[0,i]}|\mathbf{x}_{[0,i-1]}]$$

$$= \int \mathcal{D}[\mathbf{s}_{[0,i]}]\mathcal{P}[\mathbf{s}_{[0,i]}|\mathbf{x}_{[0,i-1]}]g[\mathbf{s}_{[0,i]}]. \quad \text{(A8)}$$

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

$$\mathcal{P}[\mathbf{x}_{[i-1,i]}|\mathbf{x}_{[0,i-1]}] = \langle g[\mathbf{s}_{[0,i]}]\rangle_{\mathcal{P}[\mathbf{s}_{[0,i]}|\mathbf{x}_{[0,i-1]}]}. \quad \text{(A9)}$$

In principle, this average can be computed using a Monte Carlo scheme, with trajectories generated from $\mathcal{P}[\mathbf{s}_{[0,i]}|\mathbf{x}_{[0,i-1]}]$. Notice that at each iteration of the particle filter, we dispose of a set of trajectories $\mathbf{s}^1_{[0,i]}, \ldots, \mathbf{s}^M_{[0,i]}$, which are approximately distributed according to $\mathcal{P}[\mathbf{s}_{[0,i]}|\mathbf{x}_{[0,i-1]}]$ above. Therefore, we can compute the average, Eq. (A9), directly from the trajectories that are present for each iteration of the particle filter. With the notation from Sec. III B, using $g[\mathbf{s}^k_{[0,i]}] = \exp(-U^k_i)$, we thus obtain the estimate

$$\mathcal{P}[\mathbf{x}_{[i-1,i]}|\mathbf{x}_{[0,i-1]}] \approx \frac{1}{M}\sum_{k=1}^{M} e^{-U^k_i} = \frac{w_i}{M}. \quad \text{(A10)}$$

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

$$\mathcal{P}[\mathbf{x}] = \mathrm{P}(x_0)\mathcal{P}[\mathbf{x}_{[0,1]}|x_0]\cdots\mathcal{P}[\mathbf{x}_{[n-1,n]}|\mathbf{x}_{[0,n-1]}], \quad \text{(A11)}$$

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:

$$\hat{\mathcal{P}}[\mathbf{x}] = \mathrm{P}(x_0)\prod_{i=1}^{n}\frac{w_i}{M}, \quad \text{(A12)}$$

which is precisely Eq. (37).

APPENDIX B: MCMC SAMPLING IN TRAJECTORY SPACE

Thermodynamic integration PWS in Sec. III C relies on the computation of averages with respect to $\mathcal{P}_\theta[\mathbf{s}|\mathbf{x}] \propto \exp(-\mathcal{U}_\theta[\mathbf{s},\mathbf{x}])$. Sampling from these distributions using the SSA (Gillespie) algorithm is not possible. Instead, in this section, we show different ways 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 $\mathbf{s}'$ from a

given trajectory $\mathbf{s}$ with probability $T(\mathbf{s} \to \mathbf{s}')$. We accept the proposal using the Metropolis criterion with probability

$$A(\mathbf{s}',\mathbf{s}) = \min\left(1, e^{\mathcal{U}_\theta[\mathbf{s},\mathbf{x}]-\mathcal{U}_\theta[\mathbf{s}',\mathbf{x}]}\frac{T(\mathbf{s}' \to \mathbf{s})}{T(\mathbf{s} \to \mathbf{s}')}\right) \quad \text{(B1)}$$

to create a chain of trajectories with stationary distribution given by $\mathcal{P}_\theta[\mathbf{s}|\mathbf{x}] = e^{-\mathcal{U}_\theta[\mathbf{s},\mathbf{x}]}/\mathcal{Z}_\theta[\mathbf{x}]$ for $0 \leq \theta \leq 1$. 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 $\mathbf{s}'$ must be sufficiently different from the original trajectory $\mathbf{s}$, while at the same time, it should not be so radically different that the acceptance probability is drastically reduced. 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 naive) proposal kernel is to generate an entirely new trajectory $\mathbf{s}'$ independent of $\mathbf{s}$, by sampling directly from $\mathcal{P}[\mathbf{s}]$ using the SSA. Hence, the transition kernel is given by $T(\mathbf{s} \to \mathbf{s}') = \mathcal{P}[\mathbf{s}']$, and a proposal $\mathbf{s} \to \mathbf{s}'$ is accepted with probability

$$A(\mathbf{s}',\mathbf{s}) = \min\left(1, e^{\mathcal{U}_\theta[\mathbf{s},\mathbf{x}]-\mathcal{U}_\theta[\mathbf{s}',\mathbf{x}]}\frac{\mathcal{P}[\mathbf{s}]}{\mathcal{P}[\mathbf{s}']}\right)$$

$$= \min\left(1, \frac{\mathcal{P}[\mathbf{x}|\mathbf{s}']^\theta}{\mathcal{P}[\mathbf{x}|\mathbf{s}]^\theta}\right), \quad \text{(B2)}$$

where the second line follows by inserting the definition of $\mathcal{U}_\theta[\mathbf{s},\mathbf{x}]$ given in Eq. (40). Although this simple scheme to completely regenerate an entire trajectory and accept or reject according to $A(\mathbf{s}',\mathbf{s})$ creates correctly distributed trajectories, it should not be used in simulations to compute $\mathcal{P}[\mathbf{x}]$. Indeed, we get a better estimate of $\mathcal{P}[\mathbf{x}]$ by just using the same number of independent sample trajectories from $\mathcal{P}[\mathbf{s}]$ and using the brute-force scheme in Eq. (8) without taking the detour of thermodynamic integration 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 [100]. By not regenerating the entire trajectory, the new trajectory $\mathbf{s}'$ is correlated with the original trajectory $\mathbf{s}$, and correlation, in general, improves 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 τ along the existing trajectory $\mathbf{s}$ is randomly selected, and a new trajectory segment is regrown from this point to the end, resulting in the proposal $\mathbf{s}'$. Since the new segment is generated according to the input statistics given

by $\mathcal{P}[s_{[T-\tau,T]}]$, the acceptance probability for the proposed trajectory is given by Eq. (B2). If the input dynamics given by $\mathcal{P}[s]$ are time reversible, we can also perform a backward shooting move. Here, the beginning of 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. (B2). 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. This is indeed the MCMC sampler used to obtain the TI-PWS results for the coupled birth-death process in Sec. V A.

For long trajectories, it can prove to be a problem that the middle section is too inflexible when the trail moves only regenerate either the beginning or the end of a trajectory. Therefore, one could additionally try to incorporate midsection regrowth to make sure that the middle parts of the trajectory also become flexible. To regrow a middle segment with duration τ of a trajectory s, we have to generate a new segment of duration τ according to the stochastic dynamics given by $\mathcal{P}[s]$ 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 τ are called stochastic bridging processes.

The simplest way to generate trajectories from a bridging process is by generating a trajectory segment of length τ from the normal stochastic process and rejecting the segment if it does not hit the correct end point [101]. 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 [102]. For jump processes, bridges can be simulated using particle filters [103], by a weighted stochastic simulation algorithm (wSSA) [104], or using random time discretization (uniformization) [101].

Further techniques to create trajectory-space MCMC samplers have been developed in the literature. Crooks [105] describes a scheme to create MCMC moves for trajectories evolving in nonequilibrium dynamics by making MCMC moves to change the trajectories' noise histories. In the particle Markov chain Monte Carlo (PMCMC) algorithm, trail trajectories are generated using a particle filter and accepted with an appropriate Metropolis criterion [106]. Another class of efficient samplers for Markov jump processes can be built using uniformization [107].

APPENDIX C: DEALING WITH FEEDBACK

Although, so far, we have assumed the stochastic dynamics of the input to be independent of the generated output trajectories, 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. Above, the PWS scheme was derived with this assumption. In this appendix, we drop the assumption and explicitly consider systems where the produced output perturbs the input, i.e., systems where the output feeds back onto the input. In the following, we first discuss the additional problems that arise when computing the mutual information for a system with feedback, and subsequently, we present a modified version of PWS that can be used to compute the trajectory mutual information for these systems.

1. Computing the mutual information with feedback between input and output

All PWS schemes presented above require the computation of the trajectory likelihood $\mathcal{P}[x|s]$, a quantity that is not readily available for systems with feedback. Indeed, as already mentioned in Sec. II C 1, for a given input trajectory 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 $\mathcal{P}[x|s]$ 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 $\mathcal{P}[x|s]$ in systems with feedback, we often can still compute the joint probability density $\mathcal{P}[s,x]$ 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 $\mathcal{P}[s,x]$, 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 $\mathcal{P}[s,x]$ can be obtained by expressing the mutual information as

$$\mathrm{I}(\mathcal{S}, \mathcal{X}) = \int \mathcal{D}[s] \int \mathcal{D}[x] \mathcal{P}[s, x] \ln \frac{\mathcal{P}[s, x]}{\mathcal{P}[s] \mathcal{P}[x]}. \qquad (C1)$$

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

$$\mathrm{I}(\mathcal{S}, \mathcal{X}) = \left\langle \ln \frac{\mathcal{P}[s, x]}{\mathcal{P}[s] \mathcal{P}[x]} \right\rangle_{\mathcal{P}[s, x]}, \qquad (C2)$$

which we want to estimate via a Monte Carlo average using samples from $\mathcal{P}[s, x]$. We see that while we do not need to evaluate the likelihood $\mathcal{P}[x|s]$, we now need to explicitly compute the joint density $\mathcal{P}[s, x]$ and two marginal densities, $\mathcal{P}[s]$ and $\mathcal{P}[x]$, for each Monte Carlo sample $(s, x) \sim \mathcal{P}[s, x]$. 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:

$$\mathcal{P}[s] = \int \mathcal{D}[x] \mathcal{P}[s, x] \qquad (C3)$$

and

$$\mathcal{P}[x] = \int \mathcal{D}[s] \mathcal{P}[s, x]. \qquad (C4)$$

However, these marginalization integrals cannot be directly computed with the techniques described so far. Note that while in Sec. III we discussed in detail how to compute such marginalization integrals, all methods presented there require the evaluation of the likelihood $\mathcal{P}[x|s]$ and cannot be used directly. Therefore, in the following subsection, we discuss how to compute marginalization integrals for systems with feedback.

Additionally, as discussed in Sec. IV, 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. (44).

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 to apply the brute-force Monte Carlo estimate Eq. (8) or the other, more advanced techniques from Sec. III A to systems with feedback. Nevertheless, if the system with feedback can be decomposed into a noninteracting part and an interacting part that include 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

$$\mathcal{P}[x] = \int \mathcal{D}[s] \mathcal{P}[s, x] \qquad (C5)$$

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

Formulated in the language of statistical physics, all of the techniques of Sec. III A are estimators of the free-energy difference $\Delta \mathcal{F}[x] = \mathcal{F}[x] - \mathcal{F}_0[x]$ between two ensembles described by potentials $\mathcal{U}[s, x]$ and $\mathcal{U}_0[s, x]$. Previously, for systems without feedback, we chose these potentials to be $\mathcal{U}_0[s, x] = -\ln \mathcal{P}[s]$ and $\mathcal{U}[s, x] = -\ln \mathcal{P}[s, x]$ 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 $\mathcal{P}[x]$.

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

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

$$\mathcal{U}[s, x] = \mathcal{U}_S[s] + \mathcal{U}_X[x] + \Delta \mathcal{U}[s, x], \qquad (C6)$$

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

$$\mathcal{Z}_0[x] = \int \mathcal{D}[s] e^{-\mathcal{U}_0[s, x]} \qquad (C7)$$

is known. In other words, we need to choose the decomposition in Eq. (C6) such that the partition function Eq. (C7) 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 $\mathcal{U}_0$:

$$-\ln\mathcal{P}[\boldsymbol{x}] = \mathcal{F}[\boldsymbol{x}] = \mathcal{F}_0[\boldsymbol{x}] + \Delta\mathcal{F}[\boldsymbol{x}], \qquad \text{(C8)}$$

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 Sec. III A.

As an example for finding a decomposition like Eq. (C6), 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 of 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 noninteracting system, such interactions between the input and output must be absent. Thus, a noninteracting version of the reaction system contains no single reaction that involves both S and X. We now describe how we can use that noninteracting 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 noninteracting version both S and X obey independent dynamics that are characterized by individual master equations, both $\mathcal{P}_0[\boldsymbol{s}]$ and $\mathcal{P}_0[\boldsymbol{x}]$ can be individually computed using Eq. (13). Thus, in this case, the noninteracting 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 noninteracting version that can be used to obtain the reference potential $\mathcal{U}_0[\boldsymbol{s}, \boldsymbol{x}]$. Using the techniques described in Sec. III A, 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. (C2).

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 noninteracting version of the system. However, barring a decomposition into interacting and noninteracting potentials, there is generally no unambiguous choice of the reference potential $\mathcal{U}_0[\boldsymbol{s}, \boldsymbol{x}]$ to compute the marginal probability $\mathcal{P}[\boldsymbol{x}]$. 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 Sec. III A to compute marginal probability $\mathcal{P}[\boldsymbol{x}]$. Thus, the specific choice of $\mathcal{U}_0[\boldsymbol{s}, \boldsymbol{x}]$ is system specific.

APPENDIX D: STOCHASTIC CHEMOTAXIS MODEL

We developed a stochastic chemotaxis model that describes individual reactions using a master equation framework. In our model, receptors are organized in clusters. To each cluster, we assign a probability of being active that depends on the ligand concentration. Additionally, we explicitly model the methylation and demethylation events of each receptor which affect the activity of a cluster. The cluster activity, in turn, determines its ability to phosphorylate the protein CheY. Phosphorylated CheY binds to the molecular motors driving the flagella, which alter the cell's tumbling rate. However, we do not model this downstream effect in our model.

1. MWC model

Receptors are organized in clusters on the cell surface. In our model, each cluster consists of N receptors. The ligand binding dynamics to a cluster is cooperative, and in the spirit of the Monod-Wyman-Changeux (MWC) model [79,108], we model this cooperativity by coupling the ligand binding dynamics to conformational switching dynamics of the receptors. Moreover, the energetic cost of two receptors in the same cluster being in different conformational states is prohibitively large. Thus, all receptors within a cluster of size N switch conformations in concert, so we can meaningfully speak of an active or an inactive cluster. A typical value for the cluster size is reported to be $N = 6$ by Shimizu *et al.* [80]. 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 [80]. Methyl groups can be attached to a receptor by the

TABLE II. 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 main text), 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.* [50]; we note that changing N and N_c also requires updating k_A to keep the fraction ϕ_Y of phosphorylated CheY constant.

Parameter	Value		Description
a_v	157.1	$\mu m^2\,s^{-2}$	Variance of up-gradient velocity [50]
λ	0.862	s^{-1}	Velocity correlation decay constant [50]
c_0	100	μM	Mean ligand concentration
N	6		Number of receptor units per cluster [80]
N_c	400		Number of receptor clusters [82]
M	4		Number of methylation sites per receptor [80]
N_Y	10 000		Total copy number of CheY proteins (phosphorylated and unphosphorylated) [82]
K_a	2900	μM	Ligand dissociation constant of active receptors [81]
K_i	18	μM	Ligand dissociation constant of inactive receptors [81]
k_R	0.1	s^{-1}	Methylation rate [50,80]
k_B	0.2	s^{-1}	Demethylation rate [50,80]
k_A	0.015	s^{-1}	Phosphorylation rate [83,84]
k_Z	10.0	s^{-1}	Dephosphorylation rate [83,84]
ϕ_Y	0.17		Steady-state fraction of phosphorylated CheY [83]
m_0/N	0.5		Receptor methylation level at zero ligand concentration [80]
δf_m	−2.0	$k_B T$	Free energy change of active conformation from attachment of one methyl group [80]

protein CheR and are removed by the protein CheB. We model the receptors' methylation dynamics following the model of Barkai and Leibler [75], where CheB can only demethylate active receptors. Additionally, to ensure exact adaptation, in our model, CheR can only attach methyl groups to inactive receptors, as in Ref. [109].

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,

$$p_a(c, m) = \frac{1}{1 + e^{-f(c,m)}}, \qquad (D1)$$

where

$$f(c, m) = N \ln\left(\frac{1 + c/K_i}{1 + c/K_a}\right) + \delta f_m(m - m_0). \qquad (D2)$$

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.* [80]. Their experimental results indicate that $\delta f_m = -2k_B T$, $m_0 = -N/2$. Kamino *et al.* [81] report ligand dissociation constants of $K_a = 2900\ \mu M$ for active receptors and $K_i = 18\ \mu M$ for inactive Tar receptors (for MeASP). Note that, in the equations, we assume units such that $k_B T = 1$.

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

$$\frac{dm}{dt} = (1 - p_a(c, m))k_R - p_a(c, m)k_B. \qquad (D3)$$

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

$$a_0 = \frac{k_R}{k_R + k_B}. \qquad (D4)$$

The steady-state methylation $m^\star$ can be obtained from Eqs. (D1) and (D2) by solving $p_a(c, m^\star) = a_0$:

$$m^\star = m_0 + \frac{N \ln\left(\frac{1+c/K_i}{1+c/K_a}\right) + \ln\left(\frac{1-a_0}{a_0}\right)}{-\delta f_m}. \qquad (D5)$$

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

$$\frac{dm}{dt} = -\frac{m(t) - m^\star}{\tau_m}, \qquad (D6)$$

where τ_m is the characteristic timescale of the methylation dynamics. We find τ_m by expanding p_a [Eq. (D1)] around $m = m^\star$:

$$p_a(c, m) = p_a(c, m^\star) + \left.\frac{\partial p_a}{\partial m}\right|_{m^\star} (m - m^\star) + \mathcal{O}(m^2)$$

$$= a_0[1 - \delta f_m(1 - a_0)(m - m^\star)] + \mathcal{O}(m^2), \qquad (D7)$$

and then plugging this first-order expansion into Eq. (D3), we get

$$\frac{dm}{dt} = \frac{\delta f_m(m - m^\star)}{k_R^{-1} + k_B^{-1}}. \tag{D8}$$

Thus, we find that for small perturbations, the timescale for methylation to approach the steady state is given by

$$\tau_m = \frac{k_R^{-1} + k_B^{-1}}{-\delta f_m}. \tag{D9}$$

Thus, the parameters k_R and k_R control two important characteristics of the methylation system: the adapted activity a_0 and the methylation timescale τ_m. Shimizu *et al.* [80] report an adapted activity of $a_0 = 1/3$, and based on experimental data [50,80], we assume a methylation timescale of $\tau_m = 10$ s. Our parameter choice, which is consistent with both of these observations, is $k_R = 0.075$ s^{-1} and $k_B = 0.15$ 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 s^{-1} for an attractant response and 22 s^{-1} for a repellent response [84]. Based on these data, we use the approximate dephosphorylation rate $k_Z = 10$ 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

$$\phi_Y = \frac{a_0 N_c k_A}{k_Z + a_0 N_c k_A}. \tag{D10}$$

In the fully adapted state, the phosphorylated fraction was found to be $\phi_Y \approx 0.16$ [83]. Hence, we infer a phosphorylation rate of $k_A = k_Z \phi_Y/(a_0 N_c(1 - \phi_Y)) = 0.015$ 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.* [50], we use a larger phosphorylation rate due to the smaller number of clusters, N_c.

2. Reaction kinetics

Since the timescale of conformational switching of active and inactive receptors and ligand binding is much faster [110] than the timescale of phosphorylation or methylation, we do not 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 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 $C_m \to C_{m+1}$, (b) the demethylation of a receptor $C_m \to C_{m-1}$, (c) the phosphorylation of CheY $C_m + Y \to C_m + Y_p$, and (d) the single dephosphorylation reaction $Y_p \to Y$. Thus, because of 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 $C_m \to C_{m+1}$ is given by

$$k_{m+}(c, m) = (1 - p_a(c, m))k_R. \tag{D11}$$

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

$$k_{m-}(c, m) = p_a(c, m)k_B, \tag{D12}$$

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.

Similarly, only active receptors can phosphorylate the CheY protein using the receptor-associated kinase CheA; therefore, we model phosphorylation as a reaction $C_m + Y \to C_m + Y_p$ with propensity

$$k_{Y \to Y_p}(c, m) = p_a(c, m)k_A, \tag{D13}$$

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

APPENDIX E: STOCHASTIC DYNAMICS OF THE INPUT SIGNAL FOR CHEMOTAXIS

We assume an *Escherichia coli* bacterium is swimming in a static nutrient concentration gradient $c(x)$. Following Mattingly *et al.* [50], an exponential gradient $c(x) = c_0 e^{gx}$ with gradient steepness g is assumed.

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.* [50], we assume that in a shallow gradient, the bacterial swimming dynamics are, to a good approximation, the same as in the absence of a gradient. Their experimental evidence shows that the velocity fluctuations in the absence of a gradient are described by an exponentially decaying autocorrelation function:

$$V(t) = \langle v_x(0)v_x(t)\rangle = a_v e^{-\lambda|t|}. \tag{E1}$$

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

$$\frac{dv_x}{dt} = -\lambda v_x + \sigma\xi(t), \tag{E2}$$

where $\sigma = \sqrt{2a_v\lambda}$, 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

$$\frac{d^2x}{dt^2} = -\lambda\frac{dx}{dt} + \sigma\xi(t). \tag{E3}$$

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. 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

$$\frac{dc}{dt} = \frac{\partial c}{\partial x}\frac{\partial x}{\partial t} = gc(t)v_x(t). \tag{E4}$$

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 $c(x)$.

In the PWS simulations, we use $c(t)$ directly as the input to our system. Yet, for the Gaussian approximation, we need to use a different input signal because the chemotaxis system does not respond linearly to $c(t)$. Instead, Mattingly et al. [50] show that the chemotaxis system responds approximately linearly to an input $s(t)$ defined by

$$s(t) = \frac{d}{dt}\ln c(t) = gv_x(t). \tag{E5}$$

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

$$C_{ss}(t) = \langle s(\tau)s(t+\tau)\rangle = g^2 V(t). \tag{E6}$$

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

$$P_{ss}(\omega) = g^2 V(\omega) = g^2\frac{2a_v\lambda}{\omega^2 + \lambda^2}. \tag{E7}$$

We use this same input below in Appendix F 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 $s(t)$.

APPENDIX F: MUTUAL INFORMATION RATE FOR THE GAUSSIAN CHEMOTAXIS SYSTEM

In Ref. [111], it is shown that the mutual information for a discrete-time Gaussian system can be computed using

$$I(\mathcal{S}, \mathcal{X}) = \frac{1}{2}\ln\left[\frac{|\Sigma_{ss}||\Sigma_{xx}|}{|Z|}\right], \tag{F1}$$

with

$$Z = \begin{pmatrix} \Sigma_{ss} & \Sigma_{xs} \\ \Sigma_{sx} & \Sigma_{xx} \end{pmatrix}. \tag{F2}$$

Here, Σ_{ss} and Σ_{xx} are the (auto)covariance matrices of the input and the output, respectively, whereas Σ_{sx} and Σ_{xs} contain the cross-covariances. The matrix elements are thus given by $\Sigma_{\alpha\beta}^{ij} = \langle\alpha(t_i)\beta(t_j)\rangle = C_{\alpha\beta}(t_i - t_j)$, where $C_{\alpha\beta}(t)$ denotes the (cross-)correlation functions of the system's input and output variables.

In continuous time, the information transmission rate $R(\mathcal{S}, \mathcal{X})$ of a Gaussian system in a steady state can be computed exactly from the spectral density functions of the system:

$$R(\mathcal{S}, \mathcal{X}) = -\frac{1}{4\pi}\int_{-\infty}^{\infty} d\omega \ln\left[1 - \frac{|P_{sx}(\omega)|^2}{P_{ss}(\omega)P_{xx}(\omega)}\right]. \tag{F3}$$

Here, the power spectral density $P_{\alpha\beta}(\omega)$ is the Fourier transform of $C_{\alpha\beta}(t)$, defined as

$$P_{\alpha\beta}(\omega) = \int_{-\infty}^{\infty} dt e^{-i\omega t} C_{\alpha\beta}(t). \tag{F4}$$

The information rate in the Gaussian framework can thus be computed by obtaining the required (cross-)correlation functions. In their experiments with E. coli bacteria, Mattingly et al. [50] do not 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 proceed by discussing the three kernels individually.

First, $V(t)$ 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 Appendix E, the swimming dynamics of the bacteria determine the statistics of the input signal $s(t) = (d/dt)\ln c(t)$, 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 $C_{ss}(t) = g^2 V(t)$.

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

$$K(t) = \theta(t)\langle a(t) - a_0 \rangle \ln \frac{c_s}{c_0}, \qquad (F5)$$

where we assume the input concentration jumps instantaneously from c_0 to c_s at time $t = 0$. Note that $\theta(t)$ is the Heaviside step function. 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 unit stimulus $s(t) = \delta(t)$; i.e., $K(t)$ is the Green's function of the system. The stochastic response $a(t)$ to an arbitrary time-dependent signal $s(t)$ can be written as a convolution of $K(t)$ with $s(t)$,

$$a(t) = a_0 + \int_{-\infty}^{t} dt' K(t - t') s(t') + \eta_a(t), \qquad (F6)$$

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

$$C_{sx}(t) = \langle s(\tau) x(\tau + t) \rangle = \int_{-\infty}^{t} dt' K(t - t') C_{ss}(t'). \qquad (F7)$$

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, $N(t) = \langle x(\tau) x(\tau + t) \rangle = \langle \eta_a(\tau) \eta_a(\tau + t) \rangle$, where we assume that $s(t) = 0$.

We now rewrite Eq. (F3) for the mutual information rate in terms of the three kernels described above. Thus, we need to express the power spectra $P_{\alpha\beta}(\omega)$ in terms of the Fourier-transformed kernels $V(\omega)$, $K(\omega)$, and $N(\omega)$. In Appendix E, we already showed that $P_{ss}(\omega) = g^2 V(\omega)$. The cross-power spectrum is given by $P_{sx}(\omega) = K(\omega) P_{ss}(\omega)$, which follows from Eq. (F7). Finally, from Ref. [6], 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. (F3), which yields
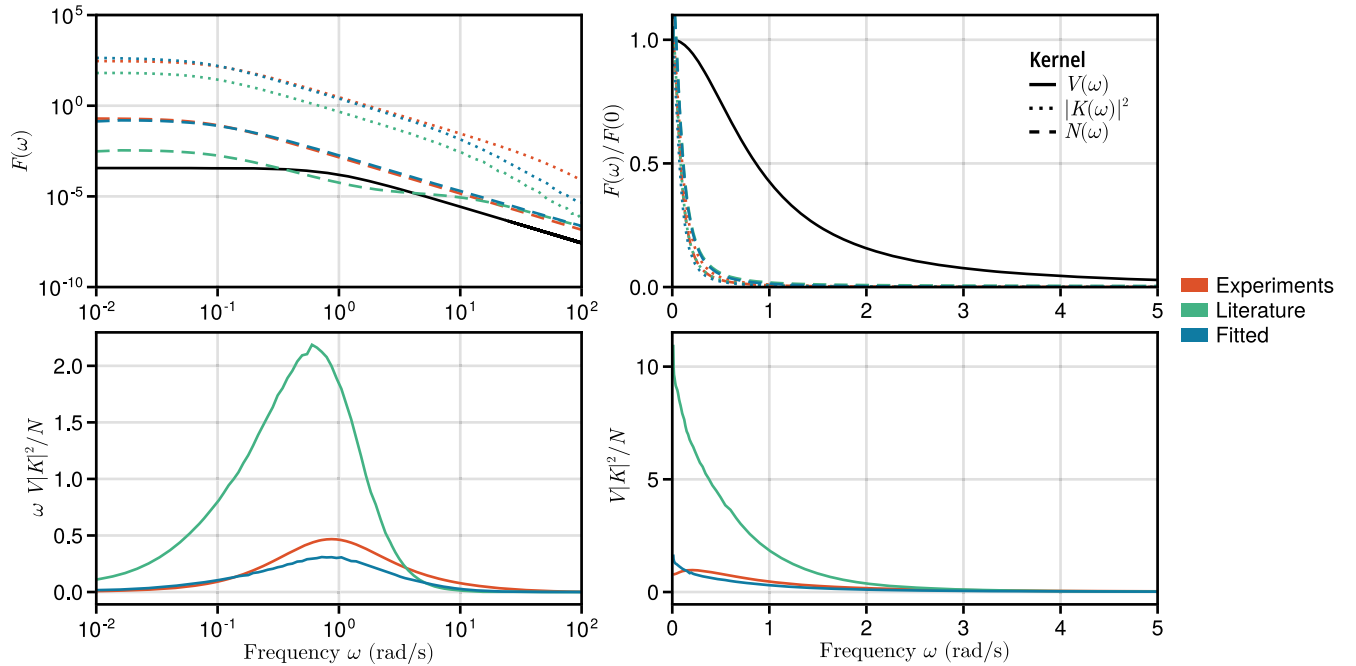

FIG. 8. Fourier representation of the kernels for computing the information transmission rate in the Gaussian approximation, the velocity power spectrum $V(\omega)$ [units $(\text{mms}^{-1})^2$], the squared frequency response $|K(\omega)|^2$, and the noise power spectrum $N(\omega)$. The top-left panel shows the individual Fourier kernels as a function of frequency ω for the different models. In the top-right panel, 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 panel, the area under the curves represents the proportionality between the squared gradient steepness g^2 and the information rate (units bit s^{-1} mm). In the bottom-left plot, the integrand is multiplied by ω; thus, with log scaling of the axes, the area under the curve is equal to the integral.

$$R(\mathcal{S}, \mathcal{X}) = \frac{1}{4\pi} \int_{-\infty}^{\infty} d\omega \ln \left(1 + \frac{g^2 V(\omega) |K(\omega)|^2}{N(\omega)} \right). \quad \text{(F8)}$$

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

$$R(\mathcal{S}, \mathcal{X}) = \frac{g^2}{4\pi} \int_{-\infty}^{\infty} d\omega \frac{V(\omega) |K(\omega)|^2}{N(\omega)} + \mathcal{O}(g^4). \quad \text{(F9)}$$

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.* [50] obtain the relevant kernels $V(\omega)$, $K(\omega)$, and $N(\omega)$ from experiments by fitting phenomenological models to their single-cell data. How we obtain these kernels for our chemotaxis model is described in Appendix G.

Figure 8 shows the Fourier representations of the relevant kernels, $V(\omega)$, $K(\omega)$, and $N(\omega)$. We computed these kernels for the three different systems: the literature-based model, the fitted model, and the experimental system of Mattingly *et al.* [50]. Because the kernels are different, so are the Gaussian information rates that we obtain. The results are shown and discussed in the main text.

APPENDIX G: QUANTITATIVE COMPARISON WITH EXPERIMENTS

We wanted to test whether our theoretical model ("literature-based model") reproduces the behavior of the experimental system studied by Mattingly *et al.* [50]. To do so, we measured the same kernels for our model as were measured experimentally. Indeed, the simulation protocol we used for measuring the kernels was directly modeled after the experimental protocol [50].

1. Measuring the response kernel

To compute the response kernel, we record the response of our chemotaxis model to a step input. To do this, we first adapt the system to the ligand concentration $c_0 = 100$ μM for $t_0 = 50$ s and then instantaneously increase the concentration from c_0 to $c_s = c_0 + 0.1 c_0$. We then record the response of the system in the 200 s following this step increase at a time resolution of 0.01 s. Note that we do not directly obtain the average receptor activity $a(t)$ from the simulations. Since in the PWS simulations we compute the mutual information between the ligand trajectory $c(t)$ and the output trajectory of the phosphorylated CheY, $y_p(t)$, we estimate $a(t)$ from the phosphorylation level of CheY. Specifically, we record the fraction $f(t)$ between phosphorylated and unphosphorylated CheY, i.e., $f(t) = [Y_p]/[Y]$. This fraction serves as a proxy for the activity $a(t)$. Indeed, because the copy number of CheY is relatively large, we can estimate the activity as

$$a(t) = \frac{k_Z}{k_A N_c} f(t). \quad \text{(G1)}$$

We then obtain an estimate for the response kernel $K(t) = \ln(c_s/c_0)\langle a(t - t_0) - a(t_0) \rangle$ by averaging the recorded activity traces over 10^5 recorded trajectories.

2. Output noise statistics

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 $c_0 = 100$ μM for a very long time of 1×10^4 s. The result is a time trace of the activity $a(t)$, which we again obtain from the fraction $f(t)$ using Eq. (G1). We discretize this time trace at a resolution of 0.01 s. This results in a time series $\boldsymbol{a} = (a_1, ..., a_N)^T$, where $a_i = a(t_i)$. To estimate the correlations in the time series, we subtract the mean 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 autocorrelation 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.

3. 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 Appendix E. We obtain $K(\omega)$ and $N(\omega)$ numerically via a discrete Fourier transform of the corresponding measured 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 take samples at times $t_0, ..., t_{N-1}$ at a sampling frequency of $f_s = 100$ s^{-1}. Then, we use the discrete Fourier transform (DFT) to obtain approximations for $K(\omega)$ and $N(\omega)$ as follows. The DFT coefficients $\tilde{K}_k$ of the time discrete response kernel are given by

$$\tilde{K}_k = \sum_{n=0}^{N-1} K_n e^{-i2\pi nk/N}, \quad \text{(G2)}$$

where $k = 0, 1, ..., N - 1$. 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

$$\omega_k = \frac{2\pi f_s k}{N}, \quad k = 0, 1, ..., N - 1, \quad \text{(G3)}$$

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, 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

$$h_n = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi n}{N-1}\right)\right], \quad n = 0, 1, \ldots, N-1. \quad \text{(G4)}$$

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

$$k_n = K_n h_n, \quad n = 0, 1, \ldots, N-1. \quad \text{(G5)}$$

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 applied to $N(t)$ as well to obtain the coefficients $\tilde{n}_k$. We can then evaluate the information rate using Eq. (F9) 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

$$R(\mathcal{S}, \mathcal{X}) = \frac{g^2}{4\pi}\sum_{k=0}^{N-1}\Delta\omega\frac{V(\omega_k)|\tilde{k}_k|^2}{\tilde{n}_k}. \quad \text{(G6)}$$

APPENDIX H: 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 $N = 15$

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

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

receptors, matches the experimental kernels of Mattingly et al. [50]. 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 Appendix F. Here, we briefly describe 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 decreasing from 400 to 50. The rate for the model of interest, with $N = 15$ and $N_c = 9$, was then obtained by fitting these data to a simple polynomial and then extrapolating to $N_c = 9$.

In Fig. 9, we show the information rate for a range of values of N_c, and for different gradient steepness 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 $R(N_c) = aN_c - bN_c^2$
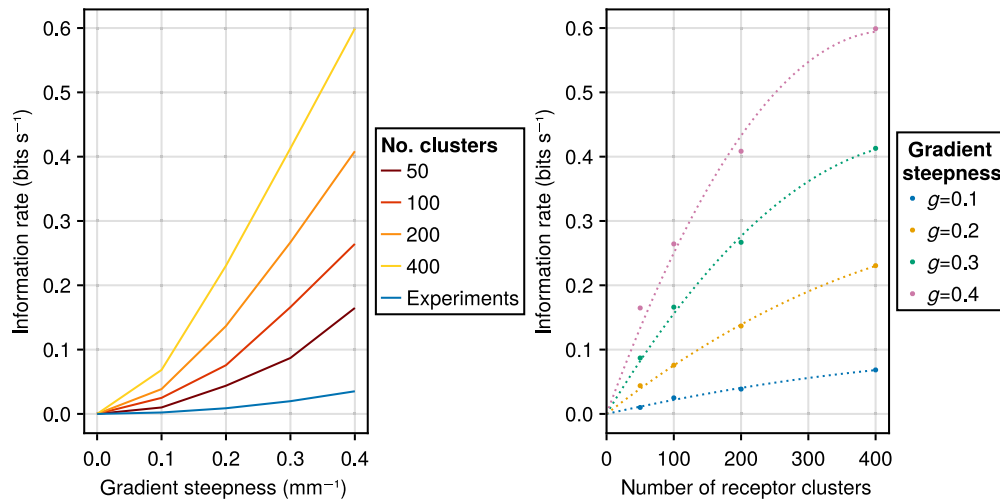

FIG. 9. Information rate as a function of the number of receptor clusters N_c. The cluster size is fixed at $N = 15$. The left panel shows the increase of information rate as a function of gradient steepness for different values of N_c, including a line for the experimental data from Mattingly et al. [50]. The right panel shows the same data but highlights the increase of the information rate 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.

with positive coefficients a, b to the data. We provide the fit coefficients for different gradient steepness g in Table III. From these fits, we can obtain the extrapolated information rates for $N_c = 9$ that are shown in the main text.

———

[1] G. Tkačik and A. M. Walczak, *Information transmission in genetic regulatory networks: A review*, J. Phys. Condens. Matter **23**, 153102 (2011).

[2] G. Tkačik and W. Bialek, *Information processing in living systems*, Annu. Rev. Condens. Matter Phys. **7**, 89 (2016).

[3] D. MacKay, *Information Theory, Inference, and Learning Algorithms* (Cambridge University Press, Cambridge, England, 2003).

[4] C. E. Shannon, *A mathematical theory of communication*, Bell Syst. Tech. J. **27**, 379 (1948).

[5] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. (John Wiley & Sons, New York, 2006).

[6] F. Tostevin and P. R. ten Wolde, *Mutual information between input and output trajectories of biochemical networks*, Phys. Rev. Lett. **102**, 218101 (2009).

[7] P. Fiedor, *Networks in financial markets based on the mutual information rate*, Phys. Rev. E **89**, 052801 (2014).

[8] J. L. Massey, *Causality, Feedback and Directed Information*, in *Proceedings 1990 International Symposium on Information Theory And Its Applications, Hawaii, USA, 1990* (1990), pp. 303–305.

[9] T. Schreiber, *Measuring information transfer*, Phys. Rev. Lett. **85**, 461 (2000).

[10] S. M. Block, J. E. Segall, and H. C. Berg, *Adaptation kinetics in bacterial chemotaxis*, J. Bacteriol. **154**, 312 (1983).

[11] J. E. Segall, S. M. Block, and H. C. Berg, *Temporal comparisons in bacterial chemotaxis*, Proc. Natl. Acad. Sci. U.S.A. **83**, 8987 (1986).

[12] C. Marshall, *Specificity of receptor tyrosine kinase signaling: transient versus sustained extracellular signal-regulated kinase activation*, Cell **80**, 179 (1995).

[13] J. Purvis and G. Lahav, *Encoding and decoding cellular information through signaling dynamics*, Cell **152**, 945 (2013).

[14] 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**, 1370 (2014).

[15] 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*, Proc. Natl. Acad. Sci. U.S.A. **115**, 6088 (2018).

[16] S. P. Strong, R. Koberle, R. R. de Ruyter van Steveninck, and W. Bialek, *Entropy and information in neural spike trains*, Phys. Rev. Lett. **80**, 197 (1998).

[17] L. Paninski, *Estimation of entropy and mutual information*, Neural Comput. **15**, 1191 (2003).

[18] R. Cheong, A. Rhee, C. J. Wang, I. Nemenman, and A. Levchenko, *Information transduction capacity of noisy biochemical signaling networks*, Science **334**, 354 (2011).

[19] G. Tkačik, C. G. Callan, and W. Bialek, *Information flow and optimization in transcriptional regulation*, Proc. Natl. Acad. Sci. U.S.A. **105**, 12265 (2008).

[20] 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**, 39 (2014).

[21] M. Meijers, S. Ito, and P. R. ten Wolde, *Behavior of information flow near criticality*, Phys. Rev. E **103**, L010102 (2021).

[22] F. Rieke, D. Warland, R. de Ruyter van Steveninck, and W. Bialek, *Spikes: Exploring the Neural Code* (MIT Press, Cambridge, Massachusetts, 1999).

[23] A. Kaiser and T. Schreiber, *Information transfer in continuous processes*, Physica (Amsterdam) **166D**, 43 (2002).

[24] A. Kraskov, H. Stögbauer, and P. Grassberger, *Estimating mutual information*, Phys. Rev. E **69**, 066138 (2004).

[25] S. A. Cepeda-Humerez, J. Ruess, and G. Tkačik, *Estimating information in time-varying signals*, PLoS Comput. Biol. **15**, e1007290 (2019).

[26] Y. Gao, I. Kontoyiannis, and E. Bienenstock, *Estimating the entropy of binary time series: Methodology, some theory and a simulation study*, Entropy **10**, 71 (2008).

[27] A. Borst and F. E. Theunissen, *Information theory and neural coding*, Nat. Neurosci. **2**, 947 (1999).

[28] M. Hledík, T. R. Sokolowski, and G. Tkačik, *A tight upper bound on mutual information*, in *2019 IEEE Information Theory Workshop (ITW)* (2019), pp. 1–5.

[29] P. J. Thomas and A. W. Eckford, *Capacity of a simple intercellular signal transduction channel*, IEEE Trans. Inf. Theory **62**, 7358 (2016).

[30] 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)* (IEEE, Piscataway, NJ, 2019), pp. 6610–6615.

[31] A.-L. Moor and C. Zechner, *Dynamic information transfer in stochastic biochemical networks*, Phys. Rev. Res. **5**, 013032 (2023).

[32] M. Delbrück, *Statistical fluctuations in autocatalytic reactions*, J. Chem. Phys. **8**, 120 (1940).

[33] D. A. McQuarrie, *Kinetics of small systems. I*, J. Chem. Phys. **38**, 433 (1963).

[34] D. A. McQuarrie, C. J. Jachimowski, and M. E. Russell, *Kinetics of small systems. II*, J. Chem. Phys. **40**, 2914 (1964).

[35] M. B. Elowitz and S. Leibler, *A synthetic oscillatory network of transcriptional regulators*, Nature (London) **403**, 335 (2000).

[36] W. Feller, *Die Grundlagen der Volterraschen Theorie des Kampfes ums Dasein in wahrscheinlichkeitstheoretischer Behandlung*, Acta Biotheoretica **5**, 11 (1939).

[37] J.-M. Park, E. Muñoz, and M. W. Deem, *Quasispecies theory for finite populations*, Phys. Rev. E **81**, 011902 (2010).

[38] J. Cremer, A. Melbinger, and E. Frey, *Growth dynamics and the evolution of cooperation in microbial populations*, Sci. Rep. **2**, 281 (2012).

[39] W. Weidlich and M. Braun, *The master equation approach to nonlinear economics*, J. Evol. Econ. **2**, 233 (1992).

[40] T. Lux, *Herd behaviour,* Bubbles and Crashes, Econ. J. **105,** 881 (1995).

[41] D. Helbing, *Traffic and related self-driven many-particle systems,* Rev. Mod. Phys. **73,** 1067 (2001).

[42] C. Castellano, S. Fortunato, and V. Loreto, *Statistical physics of social dynamics,* Rev. Mod. Phys. **81,** 591 (2009).

[43] J. I. Siepmann, *A method for the direct calculation of chemical potentials for dense chain systems,* Mol. Phys. **70,** 1145 (1990).

[44] P. Grassberger, *Pruned-enriched Rosenbluth method: Simulations of θ polymers of chain length up to 1 000 000,* Phys. Rev. E **56,** 3682 (1997).

[45] D. Frenkel and B. Smit, *Understanding Molecular Simulation,* 2nd ed. (Academic Press, San Diego, 2002).

[46] 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,* J. Chem. Phys. **81,** 3188 (1984).

[47] A. Gelman and X.-L. Meng, *Simulating normalizing constants: From importance sampling to bridge sampling to path sampling,* Stat. Sci. **13,** 163 (1998).

[48] R. M. Neal, *Annealed importance sampling,* Stat. Comput. **11,** 125 (2001).

[49] P. G. Bolhuis, D. Chandler, C. Dellago, and P. L. Geissler, *TRANSITION PATH SAMPLING: Throwing ropes over rough mountain passes, in the dark,* Annu. Rev. Phys. Chem. **53,** 291 (2002).

[50] H. H. Mattingly, K. Kamino, B. B. Machta, and T. Emonet, *Escherichia coli chemotaxis is information limited,* Nat. Phys. **17,** 1426 (2021).

[51] N. G. van Kampen, *Stochastic Processes in Physics and Chemistry,* 3rd ed. (Elsevier, Amsterdam, 2007).

[52] M. F. Weber and E. Frey, *Master equations and the theory of stochastic path integrals,* Rep. Prog. Phys. **80,** 046601 (2017).

[53] A. M. Walczak, A. Mugler, and C. H. Wiggins, *A stochastic spectral analysis of transcriptional regulatory cascades,* Proc. Natl. Acad. Sci. U.S.A. **106,** 6529 (2009).

[54] K.-Y. Kim and J. Wang, *Potential energy landscape and robustness of a gene regulatory network: Toggle switch,* PLoS Comput. Biol. **3,** e60 (2007).

[55] D. T. Gillespie, *A general method for numerically simulating the stochastic time evolution of coupled chemical reactions,* J. Comput. Phys. **22,** 403 (1976).

[56] A. Prados, J. J. Brey, and B. Sánchez-Rey, *A dynamical Monte Carlo algorithm for master equations with time-dependent transition rates,* J. Stat. Phys. **89,** 709 (1997).

[57] V. H. Thanh and C. Priami, *Simulation of biochemical reactions with time-dependent rates by the rejection-based algorithm,* J. Chem. Phys. **143,** 054104 (2015).

[58] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations* (Springer, Berlin, Heidelberg, 1992).

[59] 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 s that can arise from the dynamics given by $\mathcal{P}[s]$, while $\mathrm{H}(\mathcal{S}|\mathcal{X})$ is related to the number of distinct trajectories s that could have led to a specific output 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 x. Now, if we generate trajectories according to the dynamics given by $\mathcal{P}[s]$, with overwhelming probability, we generate a trajectory s that is not compatible with the output trajectory x, and therefore, $\mathcal{P}[x|s] \approx 0$. Hence, the effective number of samples M_{eff} is much smaller than the actual number of generated trajectories M, i.e., $M_{\mathrm{eff}} \ll M$. We therefore only expect the estimate in Eq. (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.

[60] M. Müller and W. Paul, *Measuring the chemical potential of polymer solutions and melts in computer simulations,* J. Chem. Phys. **100,** 719 (1994).

[61] M. N. Rosenbluth and A. W. Rosenbluth, *Monte Carlo calculation of the average extension of molecular chains,* J. Chem. Phys. **23,** 356 (1955).

[62] 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,** 107 (1993).

[63] T. Prellberg and J. Krawczyk, *Flat histogram version of the pruned and enriched Rosenbluth method,* Phys. Rev. Lett. **92,** 120602 (2004).

[64] R. J. Allen, D. Frenkel, and P. R. ten Wolde, *Simulating rare events in equilibrium or nonequilibrium stochastic systems,* J. Chem. Phys. **124,** 024102 (2006).

[65] N. B. Becker, R. J. Allen, and P. R. ten Wolde, *Nonstationary forward flux sampling,* J. Chem. Phys. **136,** 174118 (2012).

[66] L. Martino, V. Elvira, and F. Louzada, *Effective sample size for importance sampling based on discrepancy measures,* Signal Processing **131,** 386 (2017).

[67] A. Doucet and A. M. Johansen, *A tutorial on particle filtering and smoothing: Fifteen years later,* in *Oxford Handbook of Nonlinear Filtering,* edited by D. Crisan and B. Rozovskii (Oxford University Press, New York, 2011).

[68] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, *Equation of state calculations by fast computing machines,* J. Chem. Phys. **21,** 1087 (1953).

[69] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, *Julia: A fresh approach to numerical computing,* SIAM Rev. **59,** 65 (2017).

[70] M. Reinhardt, *PathWeightSampling.jl (v0.1.0)* (2021), 10.5281/zenodo.6334035.

[71] C. Rackauckas and Q. Nie, *DifferentialEquations.jl—A performant and feature-rich ecosystem for solving differential equations in Julia,* J. Open Res. Software **5,** 15 (2017).

[72] Y. Ma, S. Gowda, R. Anantharaman, C. Laughman, V. Shah, and C. Rackauckas, *ModelingToolkit: A composable graph transformation system for equation-based modeling*, arXiv:2103.05244.

[73] See https://github.com/zechnerlab/PathMI/tree/302f03e51-ad195adc6be39fa9618886c76590cc4.

[74] H. Berg and E. Purcell, *Physics of chemoreception*, Biophys. J. **20,** 193 (1977).

[75] N. Barkai and S. Leibler, *Robustness in simple biochemical networks*, Nature (London) **387,** 913 (1997).

[76] R. G. Endres and N. S. Wingreen, *Precise adaptation in bacterial chemotaxis through "Assistance Neighborhoods,"* Proc. Natl. Acad. Sci. U.S.A. **103,** 13040 (2006).

[77] G. Lan, P. Sartori, S. Neumann, V. Sourjik, and Y. Tu, *The energy–speed–accuracy trade-off in sensory adaptation*, Nat. Phys. **8,** 422 (2012).

[78] P. Sartori and Y. Tu, *Free energy cost of reducing noise while maintaining a high sensitivity*, Phys. Rev. Lett. **115,** 118102 (2015).

[79] J. Monod, J. Wyman, and J.-P. Changeux, *On the nature of allosteric transitions: A plausible model*, J. Mol. Biol. **12,** 88 (1965).

[80] 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*, Mol. Syst. Biol. **6,** 382 (2010).

[81] K. Kamino, J. M. Keegstra, J. Long, T. Emonet, and T. S. Shimizu, *Adaptive tuning of cell sensory diversity without changes in gene expression*, Sci. Adv. **6,** eabc1087 (2020).

[82] M. Li and G. L. Hazelbauer, *Cellular stoichiometry of the components of the chemotaxis signaling complex*, J. Bacteriol. **186,** 3687 (2004).

[83] V. Sourjik and H. C. Berg, *Receptor sensitivity in bacterial chemotaxis*, Proc. Natl. Acad. Sci. U.S.A. **99,** 123 (2002).

[84] 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*, Proc. Natl. Acad. Sci. U.S.A. **99,** 12669 (2002).

[85] M. D. Lazova, T. Ahmed, D. Bellomo, R. Stocker, and T. S. Shimizu, *Response rescaling in bacterial chemotaxis*, Proc. Natl. Acad. Sci. U.S.A. **108,** 13870 (2011).

[86] B. A. Mello and Y. Tu, *Effects of adaptation in maintaining high sensitivity over a wide range of backgrounds for Escherichia coli chemotaxis*, Biophys. J. **92,** 2329 (2007).

[87] 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 (2023), 10.1101/2022.12.04.518992.

[88] 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*, Proc. Natl. Acad. Sci. U.S.A. **112,** 11508 (2015).

[89] M. Skoge, Y. Meir, and N. S. Wingreen, *Dynamics of cooperativity in chemical sensing among cell-surface receptors*, Phys. Rev. Lett. **107,** 178101 (2011).

[90] L. Onsager and S. Machlup, *Fluctuations and irreversible processes*, Phys. Rev. **91,** 1505 (1953).

[91] A. B. Adib, *Stochastic actions for diffusive dynamics: Reweighting, sampling, and minimization*, J. Phys. Chem. B **112,** 5910 (2008).

[92] L. E. Baum and T. Petrie, *Statistical inference for probabilistic functions of finite state Markov chains*, Ann. Math. Stat. **37,** 1554 (1966).

[93] L. Paninski, Y. Ahmadian, D. G. Ferreira, S. Koyama, K. R. Rad, M. Vidne, J. Vogelstein, and W. Wu, *A new look at state-space models for neural data*, J. Comput. Neurosci. **29,** 107 (2010).

[94] 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*, Nat. Commun. **12,** 1272 (2021).

[95] J. I. Siepmann and D. Frenkel, *Configurational bias Monte Carlo: A new sampling scheme for flexible chains*, Mol. Phys. **75,** 59 (1992).

[96] T. S. van Erp, D. Moroni, and P. G. Bolhuis, *A novel path sampling method for the calculation of rate constants*, J. Chem. Phys. **118,** 7762 (2003).

[97] T. J. H. Vlugt and B. Smit, *On the efficient sampling of pathways in the transition path ensemble*, Phys. Chem. Comm. **4,** 11 (2001).

[98] P. Del Moral, *Nonlinear filtering: Interacting particle resolution*, C. R. Acad. Sci. I **325,** 653 (1997).

[99] A. F. M. Smith and A. E. Gelfand, *Bayesian statistics without tears: A sampling-resampling perspective*, Am. Stat. **46,** 84 (1992).

[100] C. Dellago, P. G. Bolhuis, and D. Chandler, *Efficient transition path sampling: Application to Lennard-Jones cluster rearrangements*, J. Chem. Phys. **108,** 9236 (1998).

[101] A. Hobolth and E. A. Stone, *Simulation from endpoint-conditioned, continuous-time Markov chains on a finite state space, with applications to molecular evolution*, Ann. Appl. Stat. **3,** 1204 (2009).

[102] F. van der Meulen and M. Schauer, *Bayesian estimation of discretely observed multi-dimensional diffusion processes using guided proposals*, Electron. J. Stat. **11,** 2358 (2017).

[103] A. Golightly and D. J. Wilkinson, *Bayesian inference for Markov jump processes with informative observations*, Stat. Appl. Gen. Mol. Biol. **14,** 169 (2015).

[104] C. S. Gillespie and A. Golightly, *Guided proposals for efficient weighted stochastic simulation*, J. Chem. Phys. **150,** 224103 (2019).

[105] G. E. Crooks, *Path-ensemble averages in systems driven far from equilibrium*, Phys. Rev. E **61,** 2361 (2000).

[106] C. Andrieu, A. Doucet, and R. Holenstein, *Particle Markov chain Monte Carlo methods*, J. R. Stat. Soc. Ser. B **72,** 269 (2010).

[107] V. Rao and Y. W. Teh, *Fast MCMC sampling for Markov jump processes and extensions*, J. Mach. Learn. Res. **14,** 3295 (2013).

[108] V. Sourjik and H. C. Berg, *Functional interactions between receptors in bacterial chemotaxis*, Nature (London) **428,** 437 (2004).

[109] C. J. Morton-Firth, T. S. Shimizu, and D. Bray, *A free-energy-based stochastic simulation of the tar receptor complex*, J. Mol. Biol. **286,** 1059 (1999).

[110] 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*, Nat. Commun. **4,** 2881 (2013).

[111] F. Tostevin and P. R. ten Wolde, *Mutual information in time-varying biochemical systems*, Phys. Rev. E **81,** 061917 (2010).