Energy-based scheme for reconstruction of piecewise constant signals observed in the movement of molecular machines

Joachim Rosskopf,[1] Korbinian Paul-Yuan,[2] Martin B. Plenio,[1] and Jens Michaelis[2]
[1]*Institute of Theoretical Physics, Ulm University, Ulm, Germany*
[2]*Institute of Biophysics Ulm University, Ulm, Germany*
(Received 3 December 2015; revised manuscript received 12 May 2016; published 29 August 2016)

Analyzing the physical and chemical properties of single DNA-based molecular machines such as polymerases and helicases requires to track stepping motion on the length scale of base pairs. Although high-resolution instruments have been developed that are capable of reaching that limit, individual steps are oftentimes hidden by experimental noise which complicates data processing. Here we present an effective two-step algorithm which detects steps in a high-bandwidth signal by minimizing an energy-based model (energy-based step finder, EBS). First, an efficient convex denoising scheme is applied which allows compression to tuples of amplitudes and plateau lengths. Second, a combinatorial clustering algorithm formulated on a graph is used to assign steps to the tuple data while accounting for prior information. Performance of the algorithm was tested on Poissonian stepping data simulated based on published kinetics data of RNA polymerase II (pol II). Comparison to existing step-finding methods shows that EBS is superior in speed while providing competitive step-detection results, especially in challenging situations. Moreover, the capability to detect backtracked intervals in experimental data of pol II as well as to detect stepping behavior of the Phi29 DNA packaging motor is demonstrated.

DOI: 10.1103/PhysRevE.94.022421

I. INTRODUCTION

Single molecule measurements of molecular motors make it possible to study the motion of individual enzymes. The studies range from enzymes making comparably large steps, e.g., motor proteins like myosin V [1] and kinesin [2], to DNA-based molecular machines which make steps on the scale of single nucleotides [3–6]. Experimental techniques to study these systems range from single molecule fluorescence localization [7] to optical and magnetic tweezers [8]. Most of these measurements represent the underlying dynamics as one-dimensional time series of positional changes. The enzymatic reactions which fuel this motion appear as stochastic events resulting in steplike movements [9] obliterated by noise. Nowadays state of the art optical tweezers experiments make it possible to study the movement of enzymes with a resolution down to single base pairs [3,10]. For example, studies on the $\varphi 29$ bacteriophage ring ATPase [11–13] used the information from step-detection data to propose a complete model of the mechanochemical cycle. However, oftentimes analysis schemes rely on low-pass smoothed data.

Indeed, the problem of finding steps is not only limited to studies of movement of enzymes but appears in a wide range of biomolecular experiments from fluorescence resonance energy transfer trajectories [14], to steps in membrane tether formation [15], or the opening of ion channels [16], just to name a few.

Consequently, there is a rich amount of signal processing techniques available to recover piecewise constant signals from noisy data. Due to the stochastic nature of enzymatic stepping, the number of steps is often not known *a priori*. Therefore, different step-finding algorithms have been developed [17–20].

One class of algorithms determines steps from single molecule data based on statistical hypothesis testing in a moving window. A prominent example is the so-called t test, which is based on Student's t test [18]. In this algorithm a step is recorded when the hypothesis that two normally distributed random variables have the same mean is violated. The mean is calculated with respect to a certain time window, which is an input parameter that can be eliminated by sweeping through various window sizes. Thus, the t test is conceptually simple. However, for situations with small step sizes and as a result comparatively large noise, increasing window sizes are required, limiting efficient step detection.

Hidden Markov models (HMMs) have been developed for situations with poor signal-to-noise ratio [21,22]. In HMM the signal is modeled as a Markov process with transitions between discrete states obliterated by Gaussian white noise. Thus, in the HMM analysis of stepping data, transition probabilities of a Markov process are obtained from a maximum likelihood estimation and the steps are reconstructed using the Viterbi algorithm [23]. A HMM for processive molecular motor data requires many states to model the possible positions on the template, making it computationally expensive. Performance can be improved by cutting the signal at a predefined amplitude and transforming positions to periodic coordinates to limit the necessary number of states [22]. HMMs proved to be excellent tools for pattern recognition in many fields. However, in addition to being computationally demanding, they rely on assumptions about the hidden stepping process and about the noise model.

Another popular class of step-finding algorithms reconstruct the underlying step signal by successively introducing new steps until a stop criterion is met [24,25]. One commonly used approach is developed by Kalafut and Visscher (K & V). It positions every new step such that the Bayesian information criterion with respect to the noisy data is minimized [24]. Whether this is a valid assumption for change-point problems is a topic of current research [26]. The algorithm does not require user input and stops when the addition of new steps is unfavorable according to the Bayesian information criterion.

The K & V algorithm is a member of the larger class of step-finding methods which minimize a certain energy

function [19]. However, since the K & V algorithm only adds new steps and does not remove previously found steps, it is not guaranteed that the global energy minimum is found [19]. Finding the global minimum is possible if these energy functions are convex. In this case, efficient algorithms can be used that yield good approximations to the underlying step signal [27]. However, for poor signal-to-noise ratio, these convex energy functions are too simplistic to optimally detect steps, resulting in an overfitting of the data; i.e., more steps are detected than are actually present. Thus, if steps are hidden in noise, such algorithms behave as efficient filter functions, and accurate step detection requires an additional second stage on the filtered, i.e., denoised data [20].

Here we present a two-stage approach, termed energy-based step finding (EBS), where both stages are based on the minimization of energy functions. In a first stage, we denoise the signal with a highly efficient and fast optimization algorithm. The algorithm minimizes a convex energy function in a process called total variation denoising (TVDN). We show that an optimal denoising can be found making the process effectively parameter free. There is no further assumption about the noise necessary. For actual step detection, we proceed in the second stage of EBS with combinatorial clustering (CC) of the denoised data into steps. Such an approach is already in common use in the computer vision community [28] and is both computationally efficient and fast. The energy functions used in CC belong to a more general class which allows the incorporation of prior knowledge such as the step size of the stepper to make the algorithm more accurate. We tested EBS with simulated data that were created based on experimental data of RNA polymerase II (Pol II) movement. We compare the performance of EBS on the same simulated data to (i) a t test, (ii) to the variable step size HMM, and (iii) to the K & V algorithm.

The analysis reveals that EBS performs faster and more accurately. We therefore applied the algorithm to detect steps in experimental data of the bacteriophage $\varphi 29$ packaging motor and to determine pauses of Pol II transcription elongation in high-resolution optical tweezers experiments.

II. METHODS

A. Energy-based model

Starting from a large and noisy trajectory of motor protein movement, we use an EBS. To reveal the steps produced by the underlying biological system hidden in noise, one has to identify piecewise constant parts in the data set. This is done by taking the N-element noisy input data and creating an N-element output set of steps. Therefore, one needs to penalize variations within neighboring variables in the signal. In contrast, one needs to increase the energy if the free variables deviate too much from the measured signal. This is reflected in the energy function

$$E(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{N} D(|x_i - y_i|) + \sum_{i=1}^{N} S(|x_i - x_{i-1}|), \quad (1)$$

where $\boldsymbol{y}$ and $\boldsymbol{x}$ are the N-element input data and output variable vectors, respectively. Minimizing the energy function is the conceptual baseline of our approach. It consists of terms where variables interact with the input data $D(\cdot)$, as well as nearest neighbor interaction between two adjacent variables $S(\cdot)$. Unfortunately, depending on the actual shape of these terms the optimization problem can get prohibitively computationally expensive [28]. One of the design goals of EBS was to work efficiently for large data sets on commodity hardware. Therefore, we chose an approach which in the first stage denoises and smoothens the signal by minimizing a simple convex energy function, solving the TVDN problem. The result of this stage is the set of denoised steps. Each step is characterized by its amplitude and length. From now on we call the combination of amplitude and length a tuple. The amount of tuples remains comparably low even for a significantly increased sampling rate. This makes EBS well suited for high-bandwidth data consisting of a huge number of data points. Afterwards we use this smaller set of tuples and minimize a more sophisticated energy function in the CC stage, which is defined on a discrete level set and incorporates a step height prior. A flowchart of this two stage process is shown in Fig. 1.

B. Total variation denoising

In the first stage of EBS, we separate noise from the actual stepping signal. This stage works on the full and noisy one-dimensional (1D) input data set $\boldsymbol{y} = y_1, \ldots, y_N \in \mathbb{R}^N$, which can be quite large ($N > 10^7$ data points). To denoise, we minimize an energy function known as the TVDN

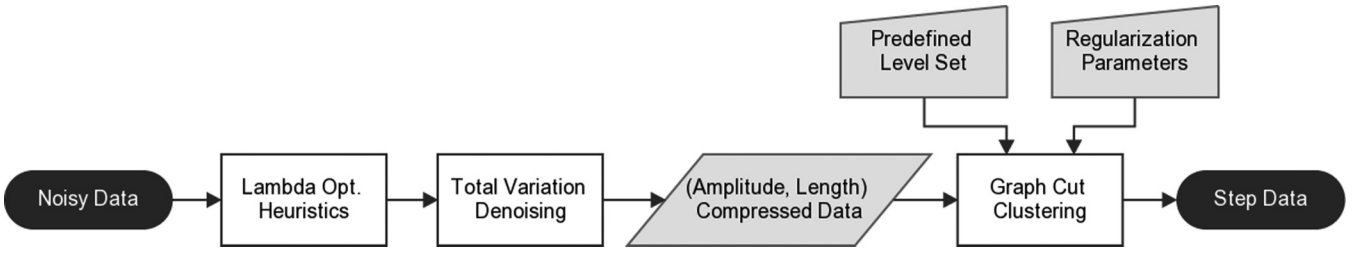


FIG. 1. Flowchart of the two-stage process for finding steps in noisy stepping data. The input data is first denoised via solving the convex TVDN problem. This process requires no intervention, as the regularization parameter λ is determined automatically. This results in a lower-dimensional, compressed representation of tuples (amplitude, length). This discrete data is then handed to a graph cut algorithm which solves a combinatorial clustering problem on a graph. The graph cut allows further customization by the use of regularization parameters ρ_i and a predefined level set.

problem [29],

$$\boldsymbol{x}^\star = \underset{x\in\mathbb{R}^N}{\operatorname{argmin}}\, q(\boldsymbol{x},\boldsymbol{y}) + p(\boldsymbol{x}) = \underset{x\in\mathbb{R}^N}{\operatorname{argmin}}\, \frac{1}{2}\sum_{i=1}^{N}|x_i - y_i|^2 + \lambda\sum_{i=1}^{N-1}|x_i - x_{i+1}|, \quad (2)$$

where the optimal solution $\boldsymbol{x}^\star$ represents the denoised signal. The $\{y_i\}$ and $\{x_i\}$ are the ith entry of the time-discrete input and solution vector, respectively. This optimal solution is a trade-off between, on the one hand, prior knowledge that the enzymatic steps yield piecewise constant signals, which is introduced by $p(\boldsymbol{x}) = \lambda\sum|x_i - x_{i+1}|$, a function which penalizes introducing steps, and, on the other hand, the term $q(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2}\sum|x_i - y_i|^2$, which penalizes deviations of the resulting solution from the input signal. The regularization parameter λ is important for the solution $\boldsymbol{x}^\star$ and controls the relative weight of the two terms. The unique solution of this problem requires no assumption about the characteristics of the noise. Therefore, the denoising step works well in case of Gaussian white noise, as well as more complicated colored noise.

The energy function in Eq. (2) is strictly convex, which means, regardless of the input data $\boldsymbol{y}$, there exists one unique solution $\boldsymbol{x}^\star$ (see, e.g., [30]). We have applied a fast algorithm for solving the TVDN problem (see the Appendix) which can easily handle millions of data points in a few milliseconds [31]. The algorithm scans forward through the signal. During this it tries to extend segments of the signal with the same amplitude until optimality conditions derived from the TVDN problem are violated. If this happens, the method backtracks to a position where a new step can be introduced, revalidates the current segment until this position, and starts a new segment (see the Appendix).

An open problem in the context of TVDN for step detection is how to choose the regularization parameter λ such that as few true steps are lost (false negatives) as possible but still the data is not overfitted (false positives). We propose a heuristic method to choose an optimal value for λ, termed λ_h, automatically. To motivate these heuristics, we have a closer look to the two limits naturally imposed on λ. For $\lambda_{\min} = 0$ the TVDN algorithm perfectly reproduces the input signal such that $\boldsymbol{x}^\star = \boldsymbol{y}$. On the other hand, the upper bound of sensible values is marked by $\lambda = \lambda_{\max}$. Above this threshold the solution of Eq. (2) is constant $x_i^\star = \text{const}$ for all i. The value of $\lambda_{\max}$ can be derived analytically from the underlying Fenchel-Rockafellar [32] problem (see the Appendix).

There exists a transition in TVDN while varying the regularization parameter from a stable minimization into the overfitting regime. Thus, by lowering λ from $\lambda_{\max}$ to $\lambda_{\min}$ one observes a sudden increase of steps produced by TVDN (Fig. 2). This marks the point where the TVDN minimization breaks down and the solution starts to fit noise. The breakdown also persists while varying sampling frequency or rate of steps as well as signal-to-noise ratio (see the Appendix). To choose the optimal value of λ, λ_h, we use a line-search algorithm which detects the sudden increase in the slope of the number of steps in the resulting signal and uses a slightly larger value. The sole input to this algorithm is the analytically determined value of $\lambda_{\max}$. Therefore, the λ_h heuristic provides us with a stable parameter-free means to choose an optimized TVDN regularization parameter.

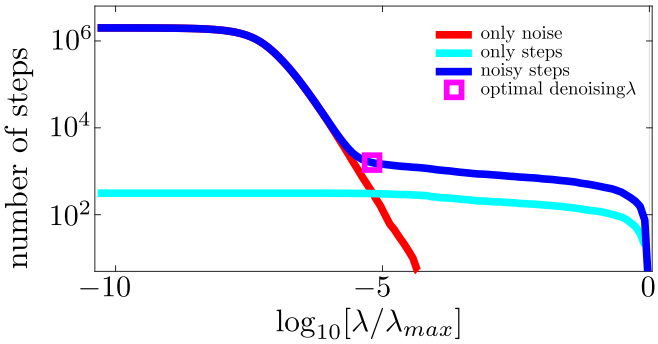


FIG. 2. Breakdown of the TVDN model dependent on $\lambda/\lambda_{\max}$. The number of produced steps and plateaus vs different λ. For small $\lambda/\lambda_{\max}$ solving the TVDN problem reproduces the input signal and the number of steps equals the number of data points. For big $\lambda/\lambda_{\max}$ the number of steps is significantly lower. The point $\lambda_h/\lambda_{\max}$ (magenta) before the number of steps increases suddenly marks the value of the TVDN regularization parameter that we choose in our heuristic. Plotted is a constant signal with added Gaussian white noise (red), a signal with exponentially distributed dwell times and Gaussian white noise (blue), and the same signal without white noise (cyan).

C. Performance characteristics of the total variation denoising algorithm

Our implementation of 1D total variation denoising is based on the C code published together with [31]. This publication also provides a detailed outline of the TVDN algorithm, description of its working principles, and the optimality condition it adheres to.

Typically, TVDN is addressed by fixed-point methods [33]. These methods reach the minimal theoretically possible algorithmic complexity [34]. A different kind of approach [31] uses the local nature of the total variation denoising filter and provides a very fast, memory efficient, noniterative way to solve Eq. (2). Although the theoretical complexity of this algorithm is worse compared to fixed-point methods, it actually achieves competitive or even faster results on signals which exhibit piecewise constant characteristics. For practical situations the complexity class of the algorithm can be assumed to be $O(N)$. Thus, for such signals denoising of 10^6 data points takes around 30 ms on a recent 2.5-GHz processor.

After successful TVDN of the signal, $\boldsymbol{x}^\star \in \mathbb{R}^N$ consists of M steps. A step is characterized by a discontinuity between two neighboring plateaus with different amplitude a. It is beneficial to represent the signal not in the basis of indexed amplitudes $x_i^\star$, but instead to use tuples $(a,w)_j, j \in 1,\ldots,M$. Where a_j is the amplitude and w_j is the length of the jth plateau. By this change of representation the number of elements of the data set is typically reduced from several millions to a few thousand. This increases computational efficiency due to the fact that the complexity of following algorithms depends on the number of elements in the data set. Therefore, a compressed signal consisting of tuples opens up the possibility to apply sophisticated step-detection algorithms to the data. The problem can now be cast as a Markov random field

[35] and can be tackled by a CC method as presented in the following section.

D. Graph cut and α expansion used for combinatorial clustering to minimize energy functions

As stated above, the input to the second stage of EBS are tuples of amplitude and corresponding length (a,w) of the compressed signal. To reveal the actual steps, these tuples have to be clustered on a discrete set of levels by minimizing an energy function. This means that a combinatorial version of an energy function similar to Eq. (1) has to be optimized. The length of a plateau plays the role of a weighting factor changing the contribution of a single tuple or a pair of tuples to the total energy. With these modifications a general energy loss function takes the form

$$E(\boldsymbol{\xi}|(\boldsymbol{a},\boldsymbol{w})) = \sum_{i\in\mathcal{V}} \mathcal{Q}_i(\xi_i|a_i,w_i) + \sum_{(i,j)\in\mathcal{E}} \mathcal{P}_{i,i+1}(\xi_i,\xi_{i+1}|a_i,a_{i+1},w_i,w_{i+1}), \tag{3}$$

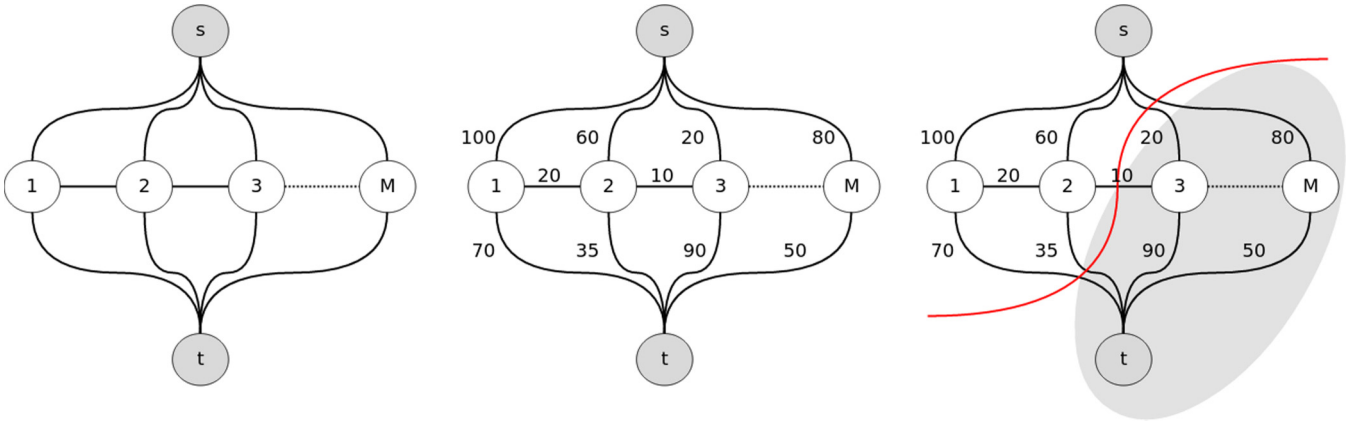
where the possible ξ_i are taken from a set of levels $\mathcal{L}$. The value of the data term $\mathcal{Q}_i(\cdot)$ depends on deviations of ξ_i from the input. The pairwise term $\mathcal{P}_{i,j}(\cdot,\cdot)$ encodes interaction potentials between neighboring plateaus. Essentially, the problem means to cluster the tuples $(\boldsymbol{a},\boldsymbol{w})$ to discrete levels, such that the joint configuration $\boldsymbol{\xi}$ minimizes $E(\boldsymbol{\xi})$.

An elegant solution can be found by mapping the problem onto a graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$, consisting of vertices $\mathcal{V}$ and edges $\mathcal{E}$. For the simple binary case, where the tuples have to be assigned to only two levels, termed source s and terminal t, both of these levels as well as all tuples represent vertices $\mathcal{V}$. $\mathcal{E}$ denotes the set of edges connecting the vertices [Fig. 3(a)] and each edge carries a capacity $c_i \geqslant 0$ [Fig. 3(b)]. Therefore, there are two types of edges: those connecting neighboring tuples and those edges connecting tuples to levels. The capacities of the former are encoded in the pairwise term $\mathcal{P}_{i,j}(\cdot,\cdot)$ and the latter are represented by the data term $\mathcal{Q}_i(\cdot)$. In the process of assigning a level ξ_i to tuples the graph cut algorithm solves the following binary decision problem: Is the assignment to level t more favorable than assignment to level s in terms of the energy function? In the graphical representation this assignment is represented by a cut through edges of neighboring tuples and edges between tuples and the s and t level [Fig. 3(c)].

Due to the well known max-flow min-cut theorem of graph theory, the optimal energy coincides with the smallest sum of capacities of the edges one has to cut from the graph to disconnect s from t [37]. The cut splits the graph $\mathcal{G}$ in two subgraphs: The part $\mathcal{S}$, which is connected to the vertex s, and the part $\mathcal{T}$, which is connected to t. The algorithm we apply solves this problem in polynomial time (see the Appendix).

To make max-flow min-cut usable for the above described assignment of multiple different levels ξ_i, it has to be embedded into an outer procedure. For this we use the α-expansion algorithm [28,38]. It finds provably good approximate solutions by iteratively solving graph cut problems on graphs representing the binary decision as to whether to alter the previous assigned level configuration [28]. For a multilevel problem new levels are added successively in a random order. That means, once the graph has been optimized for i levels and the new $i+1$th level is introduced, t corresponds to the assignment to the predefined level set and s to the new level. Again capacities for all edges are computed. With the new graph cut, vertices in the subgraph $\mathcal{S}$ get assigned their new level; the other vertices connected to $\mathcal{T}$ keep their previously assigned level. After having introduced all levels, in order to minimize the energy even more, the assignment can be optimized by iteratively reintroducing the complete level set. This iteration stops when the overall energy is not decreasing anymore (see the Appendix).

In general, finding the level configuration which coincides with minimal energy requires at least nondeterministic polynomial time. Graph cut algorithms provide the advantage to solve the problem in polynomial time, with the constraint to be just applicable to energies which exhibit a strong local minimum [39]. This is the case if the pairwise terms $\mathcal{P}_{ij}$ of the energy



(a) Build Graph Structure (b) Assign Capacities to Edges (c) Find Optimal Cut

FIG. 3. Illustration of the graph cut algorithm. (a) The initial graph structure we use to model the step signal. The nodes $i \in \{1 \ldots M\}$ represent the variables ξ_i. (b) In a second step, energies are mapped to capacities of edges. (c) The Boykov-Kolmogorov graph cut algorithm [36] finds the max flow and cuts the graph into two subgraphs $\mathcal{G} = \mathcal{S} \cup \mathcal{T}$, where $\mathcal{S}$ is the part connected to s and $\mathcal{T}$ is the remaining part connected to t.

function satisfy

$$\mathcal{P}_{ij}(\beta,\gamma) + \mathcal{P}_{ij}(\alpha,\alpha) \leqslant \mathcal{P}_{ij}(\beta,\alpha) + \mathcal{P}_{ij}(\alpha,\gamma) \tag{4}$$

for arbitrary levels $\alpha,\beta,\gamma \in \mathcal{L}$. This is also known as submodularity or regularity condition.

E. Energy function for step finding

To perform CC we have to specify the energy function, Eq. (3), as well as the level grid. The levels can be chosen arbitrarily and, depending on the problem, provide an elegant way to introduce prior information. Often molecular motors move in discrete steps with known step size. In this case, the spacing of the level grid can be chosen to match the known step size. If such information is not known *a priori* or steps are expected to be nonuniform, the levels have to be chosen with a refinement that corresponds to the required numerical accuracy, i.e., with a sufficiently small spacing.

To determine the relative importance of the terms $\mathcal{Q}$ and $\mathcal{P}$ in Eq. (3), we introduce the parameters ρ_D, ρ_S, and ρ_P which regularize the detected steps.

The data terms $\mathcal{Q}_i$ penalize deviations of the proposed level amplitude ξ_i to the original tuple amplitude a_i at the vertex v_i,

$$\mathcal{Q}_i = \rho_D w_i |\xi_i - a_i|, \tag{5}$$

where ρ_D is a regularization parameter determining the importance of the data term and w_i is the weight of the current tuple. The most prominent plateaus are likely to be discovered by TVDN and contribute a tuple with a large weight. Thus, in order to preserve these plateaus, the data term also depends on the weight w_i.

For the case of an equidistant level set $\mathcal{P}_{ij}$ consists of two different terms, a smoothing term and a term that favors steps of a certain size. The first and simpler pairwise energy uses a Potts model [40] to increase the energy whenever two assigned levels ξ_i and ξ_{i+1} differ,

$$\mathcal{P}^{\text{Potts}}_{i,i+1} = \rho_S(w_i + w_{i+1})[1 - \delta(\xi_i,\xi_{i+1})], \tag{6}$$

where $\delta(x,y) = 1$ if $x = y$ and $\delta(x,y) = 0$ else. Here ρ_S is the smoothing parameter determining the energetic penalty for differing adjacent levels. The Potts model satisfies the submodularity condition, Eq. (4) [39]. A larger regularization parameter ρ_S boosts clustering of the signal and therefore combines steps. There is no other *a priori* bias towards combining steps due to the CC algorithm itself.

The second more sophisticated contribution to the pairwise term in Eq. (3) favors level changes of specific size between adjacent sites.

This second pairwise term is optional if step sizes are uniform and it serves the purpose to introduce that prior information. Lowering the regularization parameter ρ_P gives rise to the introduction of new steps with a special step height. The complete pairwise term $\mathcal{P}_{i,j}$ thus includes prior information about step heights and is given by

$$\mathcal{P}_{i,i+1} = \mathcal{P}^{\text{Potts}}_{i,i+1} + \rho_P[1 - \delta(\xi_i,\xi_{i+1})][1 - \delta(|\xi_i - \xi_{i+1}|,\epsilon)], \tag{7}$$

with an expected average step height ϵ determined by the underlying process. The depth of the jump height prior potential is given by the jump height parameter ρ_P. Contrary to Eq. (6), we chose this term to not depend on the weights of the adjacent sites to regularize step sizes independently of the corresponding dwell times.

Note that not all pairwise terms constructed by Eq. (7) strictly fulfill the submodularity condition, Eq. (4). Therefore, we applied an extension to the graph construction procedure proposed in [41] to circumvent a submodularity violation (see the Appendix). The procedure truncates the energy until it satisfies (4). The procedure is applicable to any energy function and provides a provably good approximation for a single expansion move. For the complete α expansion the procedure is applicable if most of the terms are submodular [41]. This is fulfilled by all signals presented below: mean fraction of nonsubmodular terms $(0.27 \pm 0.08)\%$.

F. Simulation method and definition of parameters for algorithm comparison

In order to quantify positional and temporal accuracy of the steps detected by EBS, we use simulated data of noisy steps, which are generated in a two stage process. In a first stage we generate a piecewise constant signal according to a simplified Pol II stepping model where a step is the product of an enzymatic process with a certain net rate. This model contains an elongation state with forward steps of 1 bp in size generated using an effective stepping rate k_{elong}. We also account for backtracked states which can be entered by a backward step of 1 bp [42,43] with a rate $k_{b,1}$. In a backtracked state Pol II can step forward or backward by 1 bp with the rates k_f or k_b, respectively.

Second, we simulate experimental noise including effects of confined Brownian motion of trapped microspheres. To accurately reflect the experiment, we take into account changes in tether length and tether stiffness due to the motion of the enzyme. We apply a harmonic description of the trapping potentials and assume that the DNA linker can be described by a spring constant k_{DNA} determined by the wormlike chain model (see the Appendix).

In real experiments, the equilibrium position of the trapped microspheres is influenced by drift, which leads to colored noise characteristics on long time scales. Sources of drift are, for example, pointing or power fluctuations of the trapping laser or temperature drifts. To analyze the influence of drift on the detected step signal, we simulate drift as a confined Brownian motion with a very slow time constant ($\sim$10 min) and a diffusion constant of 10 nt^2/s. This represents a stochastically fluctuating baseline which is added to the simulated steps. Furthermore, the drift signal is assumed to be small enough not to affect kinetic parameters of the stepping simulation. Using these parameters, the simulation produces drifts of around 1 nm on a time scale of $\sim$1 min [Fig. 7(a)], which can be even outperformed by current high-resolution instruments [3,43].

We simulated a slow, an intermediate, and a fast scenario with elongation rates of $k_{\text{elong}} = 4.1$ Hz, $k_{\text{elong}} = 9.1$ Hz, and $k_{\text{elong}} = 25.1$ Hz, respectively. For the slow scenario, we generated $N = 2.5 \times 10^5$ data points with time increments corresponding to a 5-kHz sampling frequency. Simulated signals of the intermediate scenario consist of $N = 10^5$ data points with 2-kHz sampling frequency. The computed standard

deviation in both scenarios is 5.5 bp at the given sampling frequency. For the fast scenario we chose $N = 5 \times 10^4$ data points and a 1-kHz sampling rate. Moreover, in the fast scenario we use higher noise amplitudes with a computed standard deviation of 10.0 bp at the 1-kHz sampling frequency.

For EBS analysis of the noisy steps we have to choose the parameters ρ_D, ρ_S, and ρ_P as well as the level spacing for CC. Since our task is to optimize Eq. (3), we are only interested in relative values of the data and interaction function. Thus, we can arbitrarily set $\rho_D = 100$. ρ_S and ρ_P are parameters that have to be defined by the user. The smoothing parameter ρ_S has to be large enough to cluster small steps but small enough not to miss simulated steps. To this end, simulated data can be used to optimize parameters such that as many steps as possible are recovered but only few false positives are created (see the Appendix). We choose $\rho_S = 2$, $\rho_P = 50$ and use a level grid spacing of 1 bp, i.e., the simulated step size.

In order to compare different step-finding algorithms, we need to define a criterion when a detected step occurs at the correct time point. A detected step is classified correct whenever its temporal position lies with $\pm\Delta_{\text{window}}$ of the simulated step. We choose the window size such that $\Delta_{\text{window}} = 1/(5k_{\text{elong}})$. This allows for a small temporal shift of the detected steps with respect to a simulated step. The window is small enough to minimize classification of a step as correct by chance but large enough to make the step detection robust against numerical error.

The definition of correct steps is further used to introduce two quantities that characterize step-detection performance. The recall is defined by the number of correct steps divided by the number of simulated steps and provides information about the completeness of the recovered steps. The recall's value is meaningful only in combination with a second quantity called precision. Precision is defined by the number of correct steps divided by the number of detected steps, which is essentially the probability that a detected step is in the above defined time interval around a simulated step.

G. Detecting backtracked regions

In transcription elongation periods of forward motion are oftentimes interrupted by backward steps. This so-called backtracking is important *in vivo* for regulating transcription and therefore it is desirable to accurately detect backtracks in order to better understand regulation. Dwell times between detected steps are assigned to the set of backtracked states when they lead to a backward step. A backtracked pausing interval ends at a forward step that transfers Pol II back to the elongation state. At high noise and for fast steps we do not expect that our method will perfectly find all backtracking events present. For example, short backtracks can be omitted resulting in a long dwell time between two forward transitions in the detected steps. However, since the rates of backtracking are slow compared to elongation rates, we can correct for the missed detection of a backward step by a statistical hypothesis testing of dwell times, assuming that forward stepping follows an exponential waiting time distribution (see the Appendix). The corresponding mean dwell time can be estimated from the dwell time histogram of forward steps. Thus, dwell times which violate this hypothesis are also considered as backtracked intervals, even if the actual backtracking step is not detected.

A typical method for this separation is a Savitzky-Golay filtered velocity threshold pause detection (SGVT). SGVT finds backtracked regions in Savitzky-Golay smoothed data from histograms of instantaneous velocities [44]. These histograms show a pause peak around zero velocity and an elongation peak. One typically defines a velocity threshold by computing the mean plus one (or two) standard deviation(s) of the pause peak which is used to characterize paused regions in transcription data. A sensible choice for typical Pol II experiments of the Savitzky-Golay filter parameter is to use third order polynomials and a frame size of 2.5 s [4]. We compare the performance of the SGVT algorithm to EBS in determining backtracks.

III. RESULTS AND DISCUSSION

A. Reliable implementation of EBS

We developed the EBS algorithm to determine steps in the trajectories of molecular motors (the software package can be downloaded at https://github.com/qubit-ulm/pwcs) and tested this algorithm on simulated data of Pol II stepping using published rates (see the Methods section and the Appendix). We first simulated data using the intermediate scenario (see the Methods section). We simulated a trajectory of 50 s (i.e., 10^5 data points) resulting in 291 steps. In our simulation noise amplitudes are much larger than the 1-bp steps of the simulated step signal (Fig. 4). TVDN efficiently removes noise and produces a set of 587 plateaus [Fig. 4(a)]. The TVDN data approximate the simulated step signal, but often decomposes a simulated step into several smaller steps.

How well a particular algorithm can detect steps is best tested by computing the recall, i.e., the number of correct

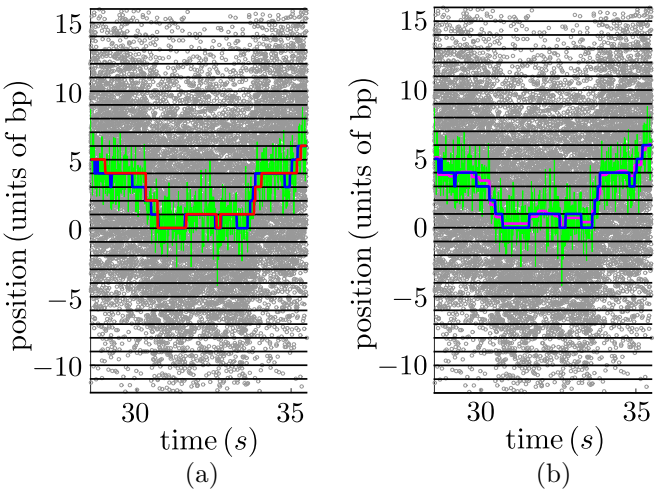


FIG. 4. EBS algorithm correctly detects steps in presence of high noise. (a) Noise reduction after application of TVDN. (b) Step detection using combinatorial clustering. Shown is a magnified interval of the simulated noisy data (gray points), boxcar averaged noisy data (20 times reduced, green), simulated steps (blue), denoised signal from TVDN [magenta (a)], and detected steps after application of combinatorial clustering by graph cut [red (b)].

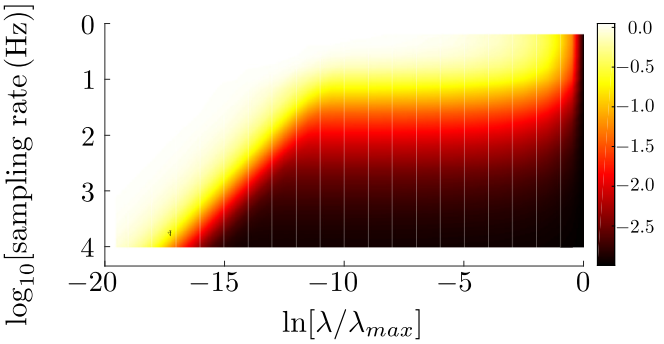


FIG. 5. Overfitting transition depends on sampling frequency. Each step signal has 1016 Poissonian distributed steps sampled with different frequencies and covered by noise. The signals are 100 s long. The color bar shows the value of $\ln(n/N)$, where N is the total number of data points and n the number of denoised steps.

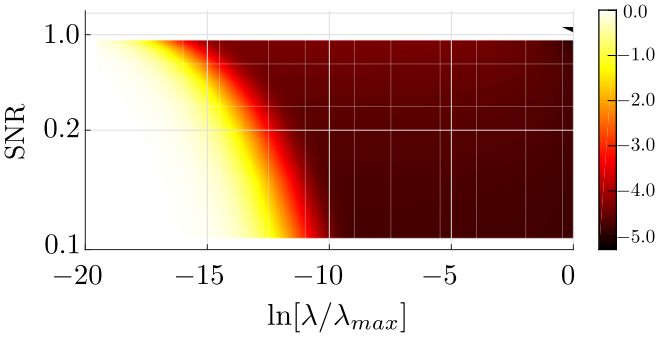


FIG. 6. Influence of SNR on over fitting transition. Each step signal has 980 Poissonian distributed steps with different noise amplitudes at 6 kHz. Every signal is 100 s long and consists of 6×10^5 data points. The scale of the color bar is chosen analog to Fig. 5.

steps divided by the number of simulated steps, and the precision, the number of correct steps divided by the number of detected steps (methods). For ideal step detection both recall and precision have to be close to one. A step finder which exhibits low recall but high precision tends to underfit the simulated step signal. On the other hand, high recall but low precision is a sign of overfitting. Both underfitting and overfitting are undesired since they may significantly distort statistical properties calculated from the detected step signal.

For the simulated data the computed recall of TVDN is 69%, which is fairly high. However, this comes at the cost of a low precision of 34%. In the second step of EBS we use CC to cluster the denoised data to predefined levels of integer multiples of the known step size of 1 bp. This results in a total of 176 found steps and thus many steps in the TVDN data are removed [Fig. 4(b)]. TVDN visually traces the simulated data very well [Fig. 4(a)], but overfits the signal; i.e., there are many more detected plateaus than simulated steps. In this example, the CC algorithm performs much better; due to the high noise not all steps are recovered [Fig. 4(b)]. Some simulated steps were missed or fused to steps of double size. Compared to the TVDN the computed recall is slightly reduced, but the precision of 51% is much higher, showing that the data are fitted more accurately. The quality of the performance of CC depends on the value of the prior potential parameters ρ_S, ρ_P tuned to optimize precision and recall (see the Appendix).

B. Stability and scalability of λ_h heuristics

The λ_h heuristic is the starting point of finding steps which are corrupted by noise and here we analyze the applicability of this scheme on simulated data. In general, we do not expect that this scheme returns good results for arbitrarily large noise amplitudes or sampling frequencies on the order of stepping rates. The dependencies on noise amplitudes and sampling frequencies for Poisson distributed steps (forward stepping with rate constant 10 Hz) covered by noise can be best summarized in the following phase diagrams (Figs. 5 and 6). As for the data shown in Fig. 2 we compute the number of produced steps after TVDN for different denoising parameter λ. For a signal of 100 s length with 1016 Poisson distributed steps, we vary the sampling frequency and keep the standard deviation of noise constant at 4.4 bp (Fig. 5). For each sampling frequency the number of produced steps is normalized to the number of simulated data points. Furthermore, we vary the standard deviation of noise and keep the sampling frequency constant at 6 kHz (Fig. 6).

In the overfitting regime (white), the number of steps of the denoised signal equals the number of data points. At $\lambda/\lambda_{\max} = 1$ the denoised signal is constant without any steps. At a sampling frequency $f = 10$ kHz the number of steps as a function of λ has a clear transition between overfitting and underfitting and resembles the data shown in Fig. 2 (blue). As the sampling frequency is lowered the transition is shifted more and more towards $\lambda_{\max}$. Below a sampling frequency of 100 Hz the number of produced steps are gradually increasing until there are as many steps as data points, as was already observed for the data in Fig. 2 (red curve). If the sampling frequency is this low, the λ_h heuristic is not applicable anymore since TVDN breaks down and just imitates noise. At 100 Hz there are, on average, ten data points for each plateau. Since steps are Poissonian distributed, many steps have plateaus that consist of less than 10 data points and are thus hardly distinguishable from noise.

For decreasing signal-to-noise ratio (SNR) we get a similar shift of the phase boundary towards $\lambda_{\max}$ for worse SNR; Fig. 6.

C. The influence of drift on EBS performance

Actual measured data exhibit drift stemming from the instrument. To analyze the influence of drift on step-detection performance of EBS, simulated data (slow scenario) are used with different amplitudes of a stochastic drift. An example of such a drift can be seen in Fig. 7(a). It produces a deviation of 16.6 nt in a time interval of 40 s compared to the simulation without drift [Fig. 7(a), green arrow]. This slightly influences the λ-versus-number-of-steps curve of the TVDN heuristic [Fig. 7(b)]. In an intermediate regime of $\lambda/\lambda_{\max}$ the additional low-frequency fluctuations produce slightly more steps when drift is present [Fig. 7(b), red] compared to the same noisy step signal without drift [Fig. 7(b), blue].

Although EBS can eliminate most of these drift induced TVDN steps, some false positives remain which decreases the algorithms step-detection precision. We analyzed the

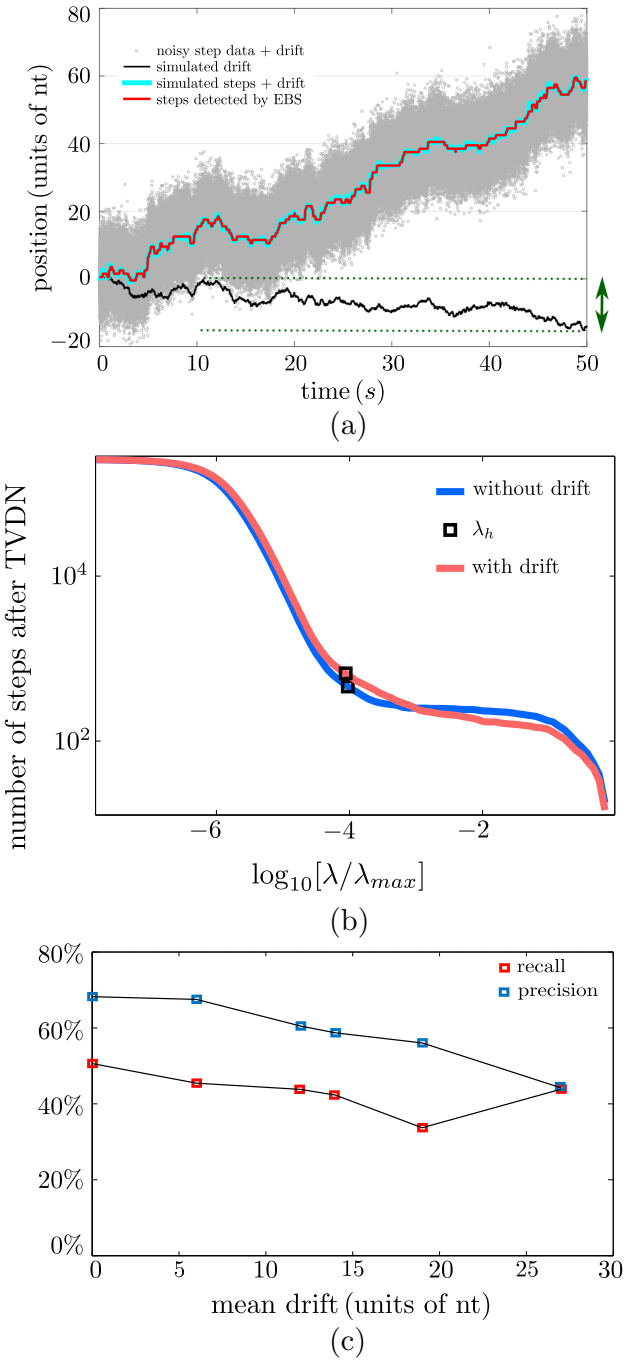


FIG. 7. Detection of steps in noisy signals corrupted by drift. (a) Shown is the noisy signal with drift (gray dots), the drift (black), simulated steps + drift (cyan), and the detected steps (red). The drift is modeled by an Ornstein-Uhlenbeck process ($1/k = 9$ min, $D = 10$ nt^2/s) and mimics low-frequency instrumental fluctuations around an equilibrium position. Here, over the simulated time interval of 50 s the signal has a drift of 16.6 nt (peak to peak, indicated by green arrow dotted lines). (b) TVDN heuristic of noisy signals with simulated drift is preserved. In the intermediate regime where λ_h is located, the λ-heuristic curve of the noisy signal with drift (red) is slightly above the curve of the same signal. (c) With increasing drift steps are detected less precisely. Shown is the precision (blue) and recall (red) of peak-to-peak difference of the drift [mean of 25 signals, simulated according to the slow scenario; SEM bars (not shown) are smaller than the size of the squares].

influence of drift on the precision by successively increasing the diffusion constant D of the drift simulation ($D \in \{0.0, 1.7, 5.9, 10.0, 34.0, 117.0\}$ nt^2/s). For every value of D, 25 signals were simulated according to the slow scenario and the mean precision of step detection was plotted against the mean peak-to-peak difference of the drift of each signal [Fig. 7(c)]. For relatively small drift (6 nt) precision decreased by only 1% and thus the influence of such a drift is rather negligible. Only the comparably large drift of 27 nt decreases the precision to 44% and thus introduces much more false positives than for signals without drift.

D. EBS outperforms existing algorithms

In the following we compare the performance of the EBS algorithm to commonly used algorithms for detecting steps in the trajectory of motor proteins, namely, a t test [18] (using an implementation that sweeps over different window sizes, [12]), the Kalafut and Visscher algorithm (K & V) [24] (using the implementation from [20]), and the variable step size hidden Markov model (HMM) [22].

In order to quantitatively compare the results of the algorithms, we chose the slow, intermediate, and fast scenarios (see the Methods section). To get statistically meaningful results, we simulated 25 time traces for each scenario. Input parameters of the step-detection algorithms were adjusted once for each simulation scenario (see the Appendix). After the analysis the detected steps were compared to the simulated input steps by computing recall and precision according to our criterion of correctly recovered steps (methods and Fig. 8).

For the slow scenario, around half of the simulated steps could be recovered by each of the four algorithms [Fig. 8(a)]. While the K & V algorithm recovers the fewest of the simulated steps (recall, 42%), the much larger precision (87%) shows that there are comparably few false positives among the detected steps. The other algorithms exhibit a somewhat smaller precision (t test 61%, HMM 64% and EBS 68%), but a higher recall (t test, 50%; HMM, 45%; and EBS, 51%). This means that more detected steps are misplaced or shifted with respect to the simulated steps. Hence, for these conditions all four algorithms work well and recover a similar amount of steps in a close vicinity of the simulated steps. However, the K & V algorithm is a little more conservative towards placement of new steps, thus increasing the precision but lowering the recall.

In the intermediate scenario stepping rates are faster, which clearly reduces the recall for the t test (15%). This effect is less dramatic for the HMM (30%), K & V (26%), and EBS (30%). The precision of HMM (67%) and K & V (65%) are at a similar level followed by EBS (57%) and t test (50%).

The fast scenario exhibits even faster steps and higher noise amplitudes and thus is the most difficult simulation setting considered here. The t test recovers 12% of the simulated steps at around half the precision of the other algorithms showing the worst performance. The performance also decreased for the other three algorithms. However, compared to HMM (10%) and K & V (8%), EBS recovers approximately twice as many correct steps (19%) at a comparable precision (HMM, 37%; K & V, 50%; and EBS, 42%).

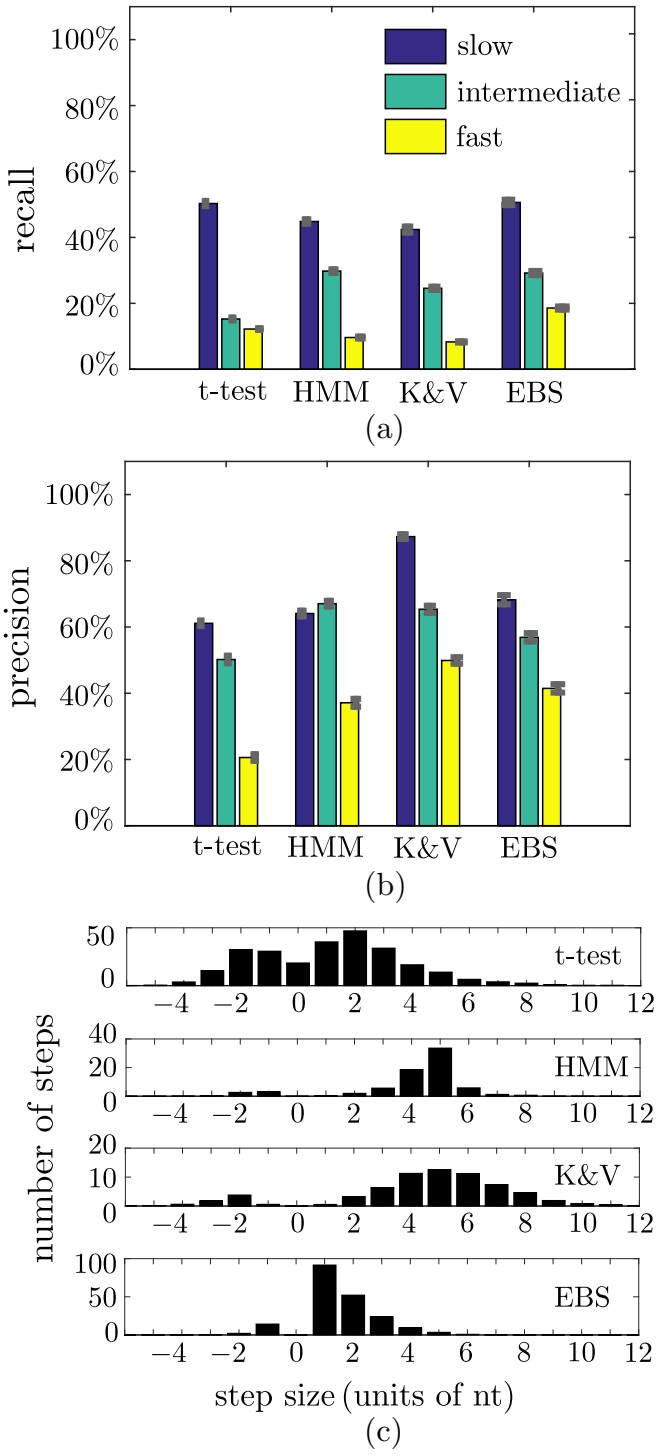


FIG. 8. Performance of step-detection algorithms with respect to slow scenario (blue bars), intermediate scenario (green bars), and fast scenario (yellow bars). Panel (a) shows, in percent of the total number of simulated steps, the number of detected steps. (b) Percentage of correct steps among the simulated steps. Error bars are SEM. Panel (c) shows average step size histograms with 1 bp binning of the detected steps of the fast scenario for t test, HMM, K & V, and EBS (from upper to lower histogram).

The correct timing of a detected step, as described by the computed values of precision and recall is only one important aspect of step detection. It is also important to test whether the recovered step-size distribution resembles the simulated stepping behavior [Fig. 8(c)]. For the fast scenario, due to the lower bandwidth and faster stepping rates, the algorithms do not reproduce the simulated step size well. Here all algorithms tend to fuse 1-bp steps to steps of larger size which explains the smaller number of found steps compared to number of simulated steps [Figs. 8(a) and 8(b)]. While the t test is showing a broad distribution of step sizes and both HMM and K & V detect mostly steps of size larger than 2 bp, the step size distribution obtained by EBS resembles the expected distribution most closely. In contrast, for the slow scenario step-size histograms show a majority of the expected 1-bp steps for all algorithms considered here (see the Appendix, Fig. 18). Therefore, in comparison with the fast scenario it becomes evident how much the noise influences the step size distributions. Compared to the other algorithms, the denoising stage of EBS is the most robust.

Important statistical properties of the underlying chemical cycle of a motor protein are often obtained from the distribution of dwell times, i.e., the duration between adjacent steps. To analyze the quality of the detected dwell times in the fast scenario, we compute dwell time histograms (25 ms binning) from the detected steps of each algorithm and compare them to the distribution of simulated dwell times (Fig. 9). While the EBS derived dwell time histogram has a shape similar to that of the actual simulation input, the other algorithms fail to recover the general shape of the histogram. This observation is also reflected in the rate constants of a double exponential fit to the dwell time distribution (Fig. 9, red curve). Here rate constants extracted from the steps detected by EBS deviate about a factor of two from those extracted directly from the simulated distribution. In contrast, the rate constants determined by the other algorithms deviate by several orders of magnitude when determining the pausing rate and are also considerably worse compared to EBS in determining the elongation rate.

How well the detected dwell time distributions reproduce the simulation can also be quantified by the Kullback-Leibler divergence (Fig. 10). As expected, the Kullback-Leibler divergence of EBS is smaller compared to the other algorithms. Moreover, due to the slower stepping rates and smaller noise amplitudes in the slow scenario, dwell time histograms of detected steps are more similar to the distribution of the simulation than in the fast scenario (Fig. 10, blue bars).

In summary, with properly adjusted parameters, none of the algorithms overfits the highly noisy data since precision exceeds recall in all four cases and thus there are fewer detected steps than simulated steps [Figs. 8(a) and 8(b)]. Nevertheless, the low recall performance means that step-detection accuracy is strongly compromised for the lower bandwidth signal of the fast scenario and directly extracting information of the underlying enzymatic cycles of elongation from dwell time fluctuations would result in errors.

E. EBS is orders of magnitude faster than competing algorithms

Moreover, we also compared the run times for all four algorithms on signals which contain 2.5×10^5 data points and ~600 simulated steps. We chose rate constants according to the intermediate scenario and recorded the respective run times (t_{run}). EBS is the fastest algorithm with run times of

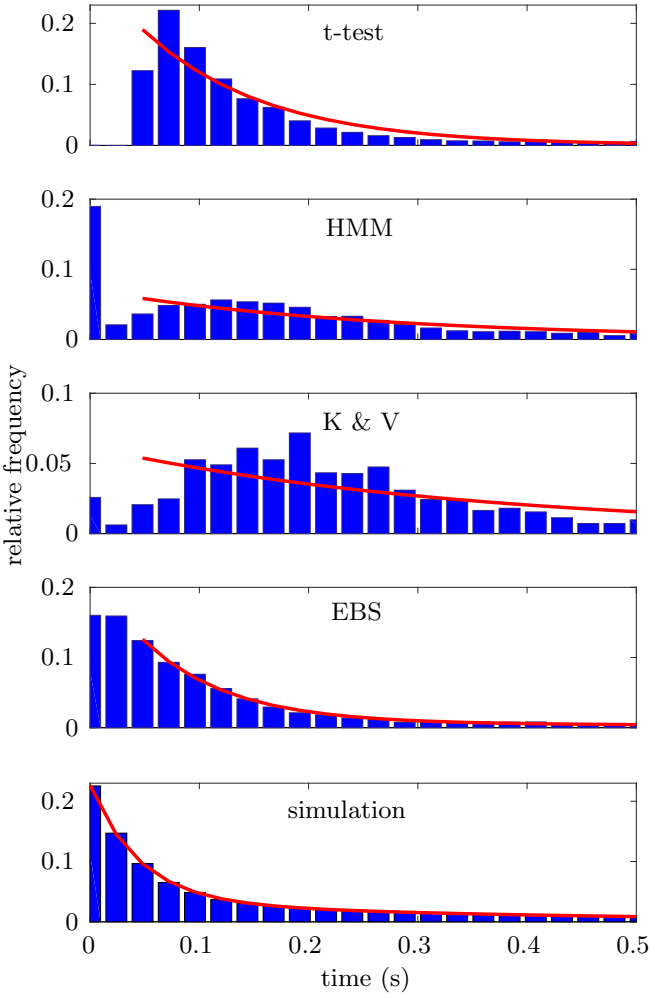


FIG. 9. Dwell time distribution of detected steps in the fast scenario. Displayed are dwell time histograms (blue bars) of the t test, HMM, K & V, and EBS algorithm (from top to bottom panel). For better comparison, the histogram of simulated dwell times is depicted in the lowest panel. Each dwell time histogram is fitted by a double exponential decay (red line), which yields the following rates ($k_{\text{pause}}/k_{\text{elong}}$ in Hz): t test (0.017/8.9), HMM (0.088/3.9), K & V (0.01/2.8), EBS (1.3/13), simulation (2.8/22). Note that in case of the detected dwell times the first two bars are not taken into account since steps with short dwell times are likely to be skipped by step-detection algorithms.

$\sim$5 s. The t test is 150 times slower, the K & V is 500 times slower, and the HMM is 1000 times slower (see the Appendix, Table II). EBS is fast enough that even very high-bandwidth signals with 10^7 data points ($\sim$900 simulated steps) can be compressed very quickly, yielding a run time of only $\sim$3 min (see the Appendix). Therefore, EBS can process much more data points at comparably short run time and is essentially limited only by the available memory size (see the Appendix). The ability to quickly process a large number of data points can be used to increase the accuracy of step finding when the signal is sampled with higher rates. For example, when using kinetics of the intermediate scenario, the recall can be increased at similar precision from 30% with 2-kHz sampling rate to 40% with 200 kHz (see the Appendix).

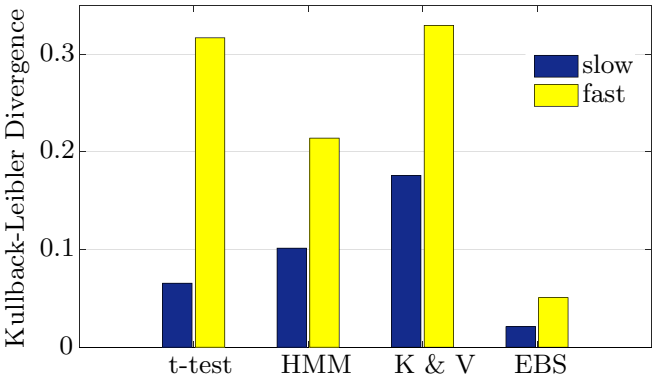


FIG. 10. The Kullback-Leibler divergence of dwell time histograms of the detected dwell time distribution with respect to the simulated one.

In summary, in the slow and intermediate scenarios the algorithms under consideration perform similarly in the total number of steps found as well as in the number of correct steps. In the fast scenario where elongation rates are faster, bandwidth is lower, and noise amplitudes are higher EBS shows better results. Moreover, when using EBS, the results of the fast scenario could be improved by higher sampling rates, which gives more data points for each plateau while still preserving comparably short run times. Thus, the EBS method especially excels for signals obtained from long measurement time, high bandwidth, and poor signal-to-noise ratio.

F. EBS detects substeps in experimental data of φ29 DNA packaging

Experimental data of Pol II transcription at saturating nucleotide concentrations yield rates comparable to the fast scenario. For these conditions the EBS as the best performing algorithm would be able to correctly detect only $\sim$19% of all simulated steps. Therefore, in order to better test the step-finding properties of EBS on actual experimental data, one would need to reduce the stepping rates or apply the algorithm to a motor protein with larger step size. A prominent example for such a process is the packaging of DNA by the bacteriophage φ29 motor, which makes steps of 10 bp, which consist of a burst of four steps with a size of 2.5 bp each [12].

We have applied EBS to experimental stepping data of φ29 recorded with a bandwidth of 2.5 kHz using opposing forces of around 5 pN [12,13]. We used 2.5 bp for the level grid spacing as well as for the jump height prior [ϵ in Eq. (7)]. The standard deviation of the experimental noise at this sampling frequency was found to be $\approx$3.8 bp. For this motor at low forces of a few pN a fast burst of four 2.5-bp steps is followed by a long dwell time [Fig. 11(a)]. The presence of 2.5-bp steps had previously been identified at large forces leading to a slow down of the 2.5-bp steps [12]. At the forces of 5 pN the previously applied t test had failed to resolve the 2.5-bp steps. In contrast, some of the steps are detected by EBS (Fig. 11 and Appendix Fig. 21).

G. EBS detects pausing of Pol II

While the EBS algorithm is not able to determine a large fraction of steps of Pol II at saturating nucleotide

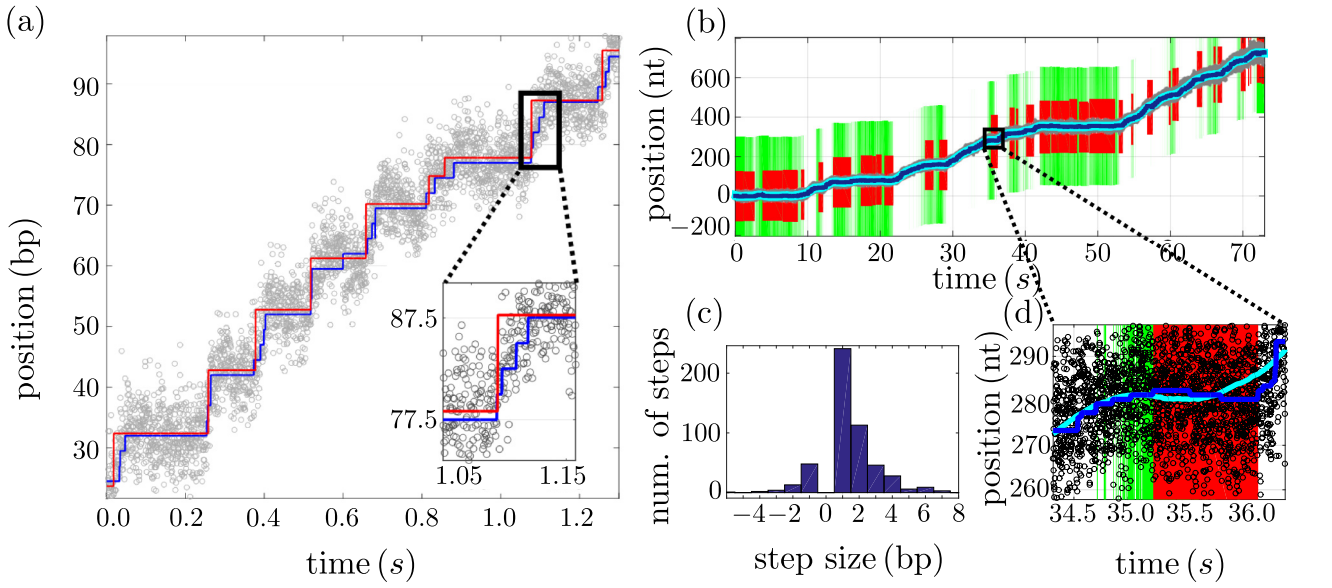


FIG. 11. (a) 2.5-bp substeps in $\varphi 29$ bacteriophage data (circles) measured in an optical tweezers experiment, EBS (blue) and t test (red). (b) Paused regions in experimental Pol II transcription elongation data. Shaded regions indicate paused intervals found by the SG-filtering method with a velocity threshold of two standard deviations of the pause peak (green) and EBS (red). 1 kHz sampled transcription data (gray), SG-filtered data of polynomial order 3 and frame size: 2.5 s (cyan) and step-detection result of our method (blue). (c) Step size histogram of detected steps by EBS in the Pol II data shown in (b). (d) Magnification of a detected pause. Shown is EBS signal (blue), SG-filtered signal (black), measured data at 1 kHz (black circles), and paused regions (SG, green; EBS, red).

concentrations given published noise levels, it can be used to investigate pausing of the enzyme.

We use EBS to detect pauses (see the Methods section) in experimental data from single molecule transcription elongation data of Pol II [45]. Further, we compare the pauses extracted from EBS step data to the result of SGVT (methods) which is a commonly applied method from the literature [4]. The signal consists of $N \sim 7 \times 10^4$ data points and was recorded with a sampling frequency of 1 kHz. The noise amplitude has an estimated average standard deviation of $\sim$10 bp. Thus, the experimental data is comparable to the fast scenario. We used both SGVT as well as EBS to detect pauses and backtracks of the enzyme (see the Methods section, Fig. 11). When comparing the results from both algorithms one finds that most long pauses do overlap, while differences are observed for the detected short pauses.

In order to get a better understanding of how well the two algorithms perform, we again use simulated data with parameters for stepping rates and sampling frequency according to the fast scenario (see the Appendix). In accordance with previously published discussions on backtracked pauses [46] we distinguish long ($t > t_p$) and short pauses ($t < t_p$) by a time scale $t_p = 1/\sqrt{k_f k_b} = 0.8$ s. All simulated long pauses were found by EBS (100%) and the total length of long pauses compared to simulated long pauses was 113%. Also the SGVT found almost all long pauses (98%) with 94% of the total duration of simulated long pauses. Both methods did not falsely assign long pauses and thus the result of finding long pauses in step-detected data and in SG filtered data largely agrees.

However, concerning short pauses, EBS outperforms SGVT in recall (EBS, 61%; SGVT, 38%) and precision (EBS, 92%; SGVT, 57%).

Especially for experiments with near base pair resolution and slow elongation rates (i.e., $k_{\text{elong}} \sim k_f, k_b$), SG filtered data is not suitable to distinguish between pauses and natural waiting times of elongation and hence step detection becomes the only option. For these experiments pause-detection accuracy is very high and allows the analysis of dwell time fluctuations. This provides further insights into enzymatic reaction cycles such as DNA sequence dependent dynamics [47].

IV. CONCLUSION AND OUTLOOK

We have presented an energy-based step-finding scheme composed of a denoising stage that uses TVDN followed by a CC analysis. The CC stage uses a graph cut algorithm and provides the possibility to include prior information. For biomotors with unknown step size, CC can be performed without step size prior terms. If the detected steps exhibit a dominant step size, a second application of EBS with this prior information can improve results. The EBS algorithm outperforms current schemes for detecting steps or pausing events in time trajectories of molecular motors. In the case of high-noise data it had the highest recall with comparable precision. The higher step-detection performance of EBS is also reflected in the step size and dwell time distributions which better reproduce the simulated distributions. In particular, for the fast scenario where the recall is rather low, further analysis of the dwell time distribution returns useful rate constants in contrast to the rates extracted from dwell time distributions of the competing algorithms. In addition EBS is much faster than competing algorithms.

In particular, the high computational speed of EBS becomes an advantage when multiple executions of the algorithm are necessary. One example for an extension of EBS with multiple

executions is an iteratively adapting level grid which could be used for signals with unknown step heights. Similar schemes are already available for HMMs [14] and were successfully applied to FRET data [48]. For EBS, this could be implemented by methods from multimodel fitting [49]. Another example could be to expand EBS to allow for drift correlation. As is, EBS is relatively insensitive to drift so that drifts on the order of 10 bp/min have a negligible effect on step finding (see the Results and Discussion section) [3]. However, one could explicitly correct for drift by using a decorrelation scheme as previously developed [50]. Again, this would necessitate multiple execution of EBS.

A reason the proposed EBS method exhibits competitive performance is a favorable representation of information in the signal, which led to the two stage process. Here we have used TVDN to build a fast and unbiased denoising scheme while still preserving the step features of the underlying signal. This was possible by using a drastically improved algorithm for solving the one-dimensional TVDN problem which allows us to choose the regularization parameter λ_h automatically. In fact, TVDN with this λ_h performs often very well in tracing the actual signal even under noisy conditions. Consequently, if TVDN is used as first stage, the choice of the regularization parameter is very important and can significantly influence the performance of further steps.

Previously, TVDN has been applied in a step-detection algorithm of the rotary flagella motor movement [20,27]. The method to determine the parameter λ developed here could be directly applied to this problem, thus increasing the accuracy of the denoising scheme. Nonetheless, a more rigorous theoretical examination of the sudden change from over- to underfitting of TVDN which led to our heuristics remains to be done. Donoho *et al.* [51] have reviewed the observation that sudden breakdowns of model selection or robust data fitting occur in high-dimensional data analysis and signal processing. They further refined this finding for compressed sensing in [52], which is a class of l_1 regularized convex optimization problems. It remains an interesting question if similar theoretical statements can be established for TVDN.

There exist different ways to solve the subsequent clustering problem for step detection. For example, when step sizes are uniform and the signal is periodic, such as for the above mentioned rotary bacterial flagella motor, a Fourier transform-based technique with nonlinear thresholding in frequency space can be used [20].

In contrast, the presented CC algorithm is broadly applicable to nonperiodic signals. We found that our implementation of CC is very well suited to cluster the output of the compression since it provides a framework to include prior information and it applies to a broad class of step signals, including steps with nonuniform sizes. Further, there are comparably fast algorithms available to solve relevant energy functions. In fact, we found that our algorithm scaled approximately quadratically in the number of tuple and linearly in the size of the predefined level set in our applications (see the Appendix). The penalizing energy scheme can be extended in an intuitive way to other prior information. For example, a histogram prior could yield a global energy term that favors certain step sizes and dwell time histograms.

The adjustment of the regularization parameters of the ρ_i energy function can be guided by comparing results with simulated stepping data. This choice is not dependent on noise due to the preceding application of TVDN. Further, by using weights in the energy terms, the regularization parameters can be applied to different data sets of the same underlying stepping process.

Both, the TVDN stage as well as the clustering stage provide the possibility to harness parallelization to gain speedups. A long high-bandwidth trajectory could be divided into smaller time intervals, which could then be treated in parallel. Of course, one would need to find a way to take care of the boundaries between the intervals, e.g., by shifting the time intervals and merging the data. This extension would also make a quasionline processing of measurement data possible, where new intervals are successively ingested.

EBS was successfully applied to detect pauses by Pol II as well as 2.5-bp steps in the packaging of DNA by the bacteriophage φ29 motor. However, while some steps could be found, the larger the noise and smaller the step size, the fewer correct steps are found. To make full use of the advantages of EBS higher-bandwidth data is needed. Moreover, shorter tether length, smaller beads, or stiffer handles provided by DNA origami [53] increase resolution and thus improve step detection.

In summary, the EBS method fills the gap of tools which are able to handle high-bandwidth data with many data points, as well as very noisy data under quite general assumptions. Regardless of the difference in TVDN and graph cut the energy-based model provides an intuitive access for the user of the method.

ACKNOWLEDGMENTS

We thank Marcus Jahnel for providing the experimental Pol II data, Gheorghe Chistol for providing the experimental φ29 phage packaging data, and Jeffrey R. Moffitt and Gheorghe Chistol for MATLABcode of the t-test algorithm. This work was supported by the EU Integrating project SIQS, the ERC Synergy Grant BioQ, as well as the ERC starting grant Remodeling and an Alexander von Humboldt Professorship. Unless otherwise stated, computations were performed on the computational resource bwUniCluster funded by the Ministry of Science, Research and the Arts Baden-Württemberg and the Universities of the State of Baden-Württemberg, Germany, within the framework program bwHPC.

J.R. and K.P-Y. contributed equally to this article. J. R., K.P.-Y., M.B.P., and J.M. designed research and wrote the manuscript. K.P.-Y. and J.R. developed algorithms and analyzed data.

APPENDIX

1. Determination of λ_{max} in TVDN

In this section we want to show how to determine the value of λ_{max} in TVDN analytically. The λ_{max} value determines the value of the regularization parameter λ in Eq. (2) above which the solution $x^\star$ remains constant and therefore contains no steps anymore. The information in the following sections is

twofold: First, derive general expressions for the Fenchel-Rockafellar-Dual problem and the forward-backward splitting applied to TVDN; second, we then derive a condition for $\lambda_{\max}$ from the Fenchel-Rockafellar dual problem and provide an analytical solution. Furthermore, we give hints on the special (tridiagonal) structure of the involved operators. The definitions in the next sections follow the work of [54].

a. Fenchel-Rockafellar dual problem and forward-backward splitting

Independent of the problem of our work, we start with a function $f(x)$ which is convex, proper, and lower semicontinuous. Then

$$\forall\, u \in \mathbb{R}^n, \quad f^*(u) = \underset{x\in\mathbb{R}^N}{\text{maximize}} \langle x,u\rangle - f(x) \tag{A1}$$

is called its Legendre-Fenchel dual function [54]. f^* is also convex, and it holds $(f^*)^* = f$. A further specialization is useful in the context of our work, as the TVDN problem consists of a minimization of two composed convex functions

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} f(x) + g[A(x)], \tag{A2}$$

where $A \in \mathbb{R}^{(p\times n)}$ and the convex functions $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^p \to \mathbb{R}$. We assume that $f^* \in C^1$ and therefore there exists a Lipschitz continuous gradient.

Due to the Fenchel-Rockafellar theorem, covered in Chapter 15 of [54], the following problems are equivalent:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} f(x) + g[A(x)] = -\underset{u\in\mathbb{R}^p}{\text{minimize}} f^\dagger(-A^\dagger u) + g^\dagger(u), \tag{A3}$$

where $\dagger$ denotes the adjoint function. The unique solution of the primal problem $x^\star$ can be recovered from a solution of the dual problem $u^\star$, which has not to be necessarily unique:

$$x^\star = \nabla f^\dagger(-A^\dagger u^\star). \tag{A4}$$

We use an additional assumption which is not a constraint for the TVDN problem: g is simple. That means one can compute a closed-form expression for the so-called proximal mapping

$$\text{prox}_{\gamma g}(x) = \underset{z\in\mathbb{R}^n}{\text{argmin}} \tfrac{1}{2}\|x - z\|^2 + \gamma g(z) \quad \forall\, \gamma > 0. \tag{A5}$$

Further, due to Moreau's identity, $g^\dagger$ is also simple [54].

Now having the connection between primal and dual problem at hand, this means one has to solve again a composite problem of a convex and a simple function,

$$\underset{u\in\mathbb{R}^p}{\text{minimize}} F(u) + G(u), \tag{A6}$$

with $F(u) = f^\dagger(-A^\dagger u)$ and $G(u) = g^\dagger(u)$.

A typical method to do proximal minimization is forward-backward splitting (see, e.g., Chap. 27 of [54]). The dual update is given by

$$u^{(\ell+1)} = \text{prox}_{\gamma G}[u^{(\ell)} - \gamma \nabla F(u^{(\ell)})]. \tag{A7}$$

In this update step, $\gamma < L/2$, where L is the Lipschitz constant. The primal iterates are given by

$$x^{(\ell)} = \nabla F(-A^\dagger u^{(\ell)}). \tag{A8}$$

The above general statements and theorems are taken from the tool set of convex analysis. For further background, see, e.g., [32] or [54]. In the following we discuss more problem specific expressions.

b. Application to total variation denoising

In a continuous picture the total variation of a smooth function $\phi : \mathbb{R} \to \mathbb{R}$ is defined as

$$J(\phi) = \int \|\nabla\phi(s)\| ds. \tag{A9}$$

In the discretized version one has to consider a discretized gradient operator $A : \mathbb{R}^n \to \mathbb{R}^p$ with $p = n - 1$,

$$J(x) = \|Ax\| = \sum_i u_i, \tag{A10}$$

where $u_i = x_{i+1} - x_i$ and therefore A taking the following form:

$$A = \begin{pmatrix} 1 & -1 & 0 & \dots & 0 \\ 0 & 1 & -1 & & \vdots \\ \vdots & & \ddots & \ddots & \\ & & & & -1 \\ 0 & \dots & & & 1 \end{pmatrix}. \tag{A11}$$

Using this and taking into account that the divergence and gradient operator are minus adjoint of each other ($\langle \nabla f, g\rangle = -\langle f, \nabla\cdot g\rangle$) the adjoint of the discrete gradient operator $A^\dagger$ is minus the discrete divergence:

$$A^\dagger = \begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & & \vdots \\ 0 & 1 & \ddots & \ddots & \\ \vdots & & \ddots & & 1 \\ 0 & \dots & & 1 & -2 \end{pmatrix}. \tag{A12}$$

Therefore, the divergence highly resembles a typical Laplace filter from signal processing. This leads for a single entry to $u_i - u_{i-1} = x_{i+1} - 2x_i + x_{i-1}$. For the deviation of $\lambda_{\max}$, we assume the boundary conditions that $u_0 = 0$ and $u_n = 0$.

For noise removal [and to get the connection to Eq. (2)] the following problem has to be solved:

$$x^\star = \underset{x\in\mathbb{R}^n}{\text{argmin}} \tfrac{1}{2}\|x - y\|^2 + \lambda J(x). \tag{A13}$$

To make use of the material so far, choose the following composition:

$$f(x) = \tfrac{1}{2}\|x - y\|^2 \quad \text{and} \quad g(u) = \lambda\|u\|. \tag{A14}$$

After that, one has to translate $f(x)$ and $g(x)$ into their dual representations $f^\dagger(u)$ and $g^\dagger(u)$ by using the following relations.

(i) For the case where $f(x) = 1/2\|Ax - y\|$ and $A \in \mathbb{R}^{n\times n}$ can be inverted, then

$$f^\dagger(u) = \tfrac{1}{2}\|(A^\dagger)^{-1}u + y\|^2. \tag{A15}$$

(ii) For the case where $f(x) = \|x\|_p = \sum_i (|x_i|^p)^{\frac{1}{p}}$ is a p norm, then the dual function corresponds with the indicator

function ι_C of the convex set C:

$$f^\dagger(u) = \iota_{\|\cdot\|\leqslant 1} \quad \text{where} \quad \frac{1}{q} + \frac{1}{p} = 1. \tag{A16}$$

Using that, we get the following dual representation of the dual functions $F(u) + G(u)$ for the TVDN problem in the Fenchel-Moreau-Rockafellar formulation:

$$F(u) = \tfrac{1}{2}\|y - A^\dagger u\|^2 - \tfrac{1}{2}\|y\|^2 \quad \text{and}$$
$$G(u) = \iota_C(u), \quad \text{where} \quad C = \{u : \|u\|_\infty \leqslant \lambda\}. \tag{A17}$$

The solution to the dual problem $u^\star$ can be obtained by solving

$$u^\star \in \underset{\|u\|\leqslant\lambda}{\operatorname{argmin}} \|y - A^\dagger u\|, \tag{A18}$$

and by applying Eq. (A4) the solution to the primal problem $x^\star$:

$$x^\star = y - A^\dagger u^\star. \tag{A19}$$

What is missing for concrete expression for the forward-backward iterations is first a closed form for the gradient of F, which is given by

$$\nabla F(u) = A(A^\dagger u - y). \tag{A20}$$

Second, it is possible for the proximal operator of G, which is the orthogonal projection on set C,

$$\operatorname{prox}_{\gamma G} u = \frac{u}{\max(1, \|u\|/\lambda)}, \tag{A21}$$
$$\gamma < \frac{2}{\|A^\dagger A\|} = \frac{1}{4}. \tag{A22}$$

Inserting the above statements into the general dual update step from Eq. (A7), one gets the following expression:

$$\begin{aligned} u^{(l+1)} &= \operatorname{prox}_{\gamma G}[u^{(l)} - \gamma \nabla F(u^{(l)})] \\ &= \frac{u^{(l)} - \gamma \nabla F(u^{(l)})}{\max\left(1, \frac{\|u^{(l)} - \gamma \nabla F(u^{(l)})\|}{\lambda}\right)} \\ &= \frac{u^{(l)} - \gamma A(A^\dagger u^{(l)} - y)}{\max\left(1, \frac{\|u^{(l)} - \gamma A(A^\dagger u^{(l)} - y)\|}{\lambda}\right)}. \end{aligned} \tag{A23}$$

.

c. *Derivation of $\lambda_{\max}$ from the proximal iteration*

Finding a maximal regularization parameter λ is equal to finding a a criterion, such that the dual iterations remain constant $\forall\, l$:

$$u^{(l+1)} \stackrel{!}{=} u^{(l)}. \tag{A24}$$

By using Eq. (A19) one can see that this will lead to a steady state solution $x_i^\star = \text{const}\ \forall\, i$. For simplicity, assume $\tilde{\lambda} = \lambda/\gamma$. Starting from the proximal iteration we find that in case of $\tilde{\lambda} \leqslant \|u^{(l)} - \nabla F(u^{(l)})\|$ the problem simplifies to

$$u^{(l+1)} = u^{(l)} - Ay + AA^\dagger u^{(l)}. \tag{A25}$$

To satisfy the constant condition from Eq. (A24), the $u^{(l)}$ has to be in the solution of

$$AA^\dagger u = Ay. \tag{A26}$$

The shape of $AA^\dagger$ is the following:

$$AA^\dagger = \begin{pmatrix} -3 & 3 & -1 & \dots & 0 \\ 1 & -3 & 3 & \ddots & \vdots \\ 0 & 1 & \ddots & \ddots & -1 \\ \vdots & & \ddots & -3 & 3 \\ 0 & \dots & & 1 & -2 \end{pmatrix}. \tag{A27}$$

Linear equations with a tridiagonal affine transform $AA^\dagger$ can be efficiently solved for example an algorithm proposed by Rose [55].

Still missing is a treatment of the primal iteration step $x^{(l+1)} = y - A^\dagger u^{(l+1)}$. The connection to the λ in the original TVDN problem is given such that, the Karush-Kuhn-Tucker conditions are still valid for our steady state solution (A26). This means that every $u^{(l)}$ in the dual solution has to satisfy

$$u_k^\star \in [-\lambda, \lambda]. \tag{A28}$$

To ensure this, we have to choose

$$\lambda_{\max} = \|u\|_\infty, \tag{A29}$$

which gives as a clear statement how to choose a maximal λ.

2. Algorithm implementing the λ_h heuristics

As outlined in the Methods section of the paper, we use a sudden increase of resulting steps when decreasing the regularization parameter λ in the TVDN problem shown in Eq. (2) from $\lambda_{\max}$ to determine λ_h. In the following, we want to describe the heuristic method we used to choose the value of λ_h. The starting point for the algorithm is the value of $\lambda_{\max}$ on a curve like the one depicted in Fig. 2. The iterative method shown in Algorithm 1 approximates the point of steepest ascent in an λ-n diagram, where n is the number of steps by searching an interval where the slope exceeds the slope of the secant of $\{0, \lambda_{\max}\}$. The function $N(\lambda)$ counts the number of steps after the TVDN minimization for a given value of λ.

This simple method gave us stable results for a variety of our test signals, either simulated or experimentally gathered. In the following section we have a closer look into the stability of the effect of sudden increase of steps.

Algorithm 1 Outline of our line-search algorithm to determine λ_h

$\lambda, n \leftarrow \lambda_{\max}, N(\lambda_{\max})$
$\lambda^+, n^+ \leftarrow \frac{\lambda_{\max}}{2}, N(\lambda_{\max}/2)$
$\delta_{\text{start}} \leftarrow \frac{|N(0) - N(\lambda_{\max})|}{\lambda_{\max}}$
while less than max. iterations **do**
  $\delta \leftarrow \frac{|n^+ - n|}{\lambda^+ - \lambda}$
  if $\delta > \delta_{\text{start}}$ **then**
    break
  end if
  $\lambda, n \leftarrow \lambda^+, n^+$
  $\lambda^+, n^+ \leftarrow \frac{\lambda^+}{\rho}, N(\lambda^+)$
end while
return $\lambda_h \leftarrow \lambda^+$

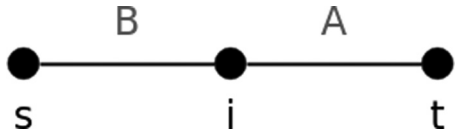


FIG. 12. Situation in an Markov random field concerning a single variable v_i and edges A, B to special variables s and t relevant for the data term.

3. Mapping of energies on edge capacities

In the process of assigning a level ξ_i to vertex v_j, the above mentioned graph cut algorithm solves a binary decision problem, whether the assignment of a new level is more favorable in terms of the energy loss function or not. The binary outcome of the decision is reflected in the graph structure by introducing two special vertices, where t is associated with keeping the old level and s is associated with assigning the proposed level. The energy values of the data term $\mathcal{Q}_i$ as well as the pairwise term $\mathcal{P}_{i,i+1}$ and their different combinations of keeping the current level or assigning a new level are mapped to capacities of edges in the Markov random field. In this section this mapping is explained stemming theoretical foundations outlined by Kolmogorov *et al.* in [56]. The graph cut algorithm solves this problem in polynomial time for a certain set of useful energy functions [36,56,57].

In Fig. 12 the situation for the data term is depicted. Here the mapping is easy, as the energy for a single variable v_i for the current level E_0 is mapped to edge A. The energy E_1 for a new level is mapped to edge B.

The situation for the pairwise term $\mathcal{P}_{i,j}$ is more complicated and depicted in Fig. 13. Here two variables v_i and v_{i+1} are involved, which leads to four different energy combinations $E_{0,0}$, $E_{0,1}$, $E_{1,0}$, and $E_{1,1}$ are possible. Here $E_{0,0}$ is associated with the energy value if both variables get assigned a new level. In contrast $E_{1,1}$ represents the energy of both variables keeping their current levels. The two other combinations represent the case when one variable keeps the current label and the other gets the new level assigned. For $E_{0,1}$ the variable $i+1$ keeps its level; for $E_{1,0}$ this is the case for the variable i.

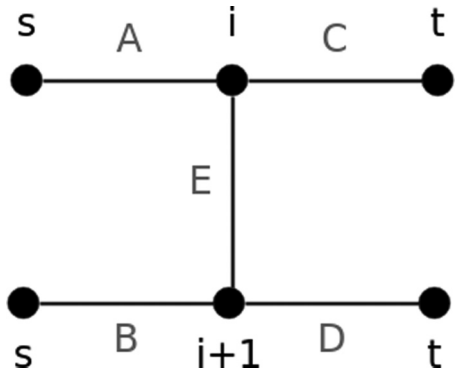


FIG. 13. Two neighboring variables in a Markov random field and edges relevant for the pairwise term. Here the two s vertices represent the same vertex in the graph and are just drawn separated to make the diagram clearer. The same is true for the t vertices.

The four energies can be represented in the following way:

$$\begin{Vmatrix} E_{0,0} & E_{0,1} \\ E_{1,0} & E_{1,1} \end{Vmatrix} = \begin{Vmatrix} a & b \\ c & d \end{Vmatrix} = \begin{Vmatrix} a & a \\ d & d \end{Vmatrix} + \begin{Vmatrix} 0 & b-a \\ c-d & 0 \end{Vmatrix}. \tag{A30}$$

The first summand on the right hand side is mapped to terminal capacities. This means that the capacity a is associated with the edge C, and the capacity d is associated with the edge B. The second summand maps to the edge E and gets the capacity $b-a+c-d$.

At this point the above mentioned strategy to circumvent a violation of submodularity is applied if $E_{0,0}+E_{1,1} > E_{0,1}+E_{1,0}$. Then in turn $E_{0,1}$ and $E_{1,0}$ is increased and $E_{0,0}$ is decreased by a small amount until the submodularity condition Eq. (4) is satisfied. Details and limitation of this approach can be found in [41].

4. α-Expansion algorithm outline

Finding a solution $\boldsymbol{\xi}^\star$ that minimizing Eq. (3) is a problem that is, in general, non-deterministic polynomial-time (NP) hard to solve for $|\mathcal{L}| \geqslant 3$. The iterative α-expansion algorithm outlined in Algorithm 2 finds provably good approximate solutions to this problem.

Algorithm 2 α-Expansion outline

1: $\xi' \leftarrow$ arbitrary labeling of sites
2: **while** not converged **do**
3: **for all** $\alpha \in \mathcal{L}$ **do**
4: $\xi^\alpha \leftarrow \underset{\xi}{\operatorname{argmin}} E(\xi, \xi')$
5: **if** $E(\xi^\alpha) < E(\xi')$ **then**
6: $\xi' \leftarrow \xi^\alpha$
7: **end if**
8: **end for**
9: **end while**

In each iteration the algorithm updates or moves the current labeling ξ' if it has found a better configuration. To achieve this, in each iteration, a new, randomly chosen label $\alpha \in \mathcal{L}$ is introduced and each site v_i has the choice to stay with the previous label or adopt the new proposed label α. The binary optimization problem is solved via a graph cut (line 4 of Algorithm 2). This step is called α expansion due to the fact that the number of nodes with the label α assigned could grow during this phase. The outer iteration stops if no new label assignments happened within two cycles. The α-expansion algorithm was initially published by Boykov *et al.* in [28].

5. Relaxation of submodularity condition by truncating energy

The submodularity condition (4) imposes structure on the energy minimization problem which allows stronger algorithmic results. In this sense the concept of submodularity plays a similar role for discrete, combinatorial clustering as convexity plays for continuous optimization.

The max-flow min-cut algorithm we use for minimization relies on that the supplied energy function satisfying the submodularity condition. Unfortunately, the pairwise term

(7) does not strictly satisfy the submodularity condition (4). Therefore, we adopted a truncation scheme proposed by Rother *et al.* in [41]. The truncation procedure for a single term can be summarized as follows: Either $\mathcal{P}_{ij}(\beta,\gamma)$ decreased or $\mathcal{P}_{ij}(\beta,\alpha)$ or $\mathcal{P}_{ij}(\alpha,\gamma)$ are increased until the submodularity condition is satisfied. This procedure is applicable to any energy function, and provides a provably good approximation for a single expansion move. The authors of [41] limit suitability for the case only a limited amount of terms are nonsubmodular.

In principle, there exist more sophisticated graph cut algorithms that alter the mapping of the combinatorial values of the energy to capacities of the edges of the graph [39]. In the same work, the authors compare performance of their more complicated optimization scheme for nonsubmodular energies to truncating the energy as we did. For a small percentage of terms violating the submodularity condition no severe degradation of the performance was found so we stayed with the simpler method, as it is more accessible and easier to reason about. The implications of nonsubmodular terms highly depend on the underlying data set and the chosen pairwise energy function. If, like in case of our label prior term (7) the nonsubmodular case is a rare event, the simple truncation procedure has a positive impact. Thus, the submodularity violation is a problem that has rather theoretical implications than practical importance for our applications.

6. Scaling of graph cut

When analyzing high-bandwidth noisy time traces of the movement of molecular motors the CC step often limits run time performance. Most of all, performance is influenced by system size, i.e., the number of tuples and number of levels in the label grid set. To analyze the scaling behavior for these two influences numerically, we simulated ten noisy Poisson step signals for each system size and label grid set and record computation times. Figure 14 shows mean and standard deviations as error bars. In Fig. 14(a) system size was increased from 250 to 3000 tuples and the number of levels offered to the combinatorial optimization problem was kept constant to around 800 levels. In this case computational time is expected to scale mostly with the complexity of the Boykov-Kolmogorov max flow algorithm which has a worst case complexity of $\mathcal{O}(|\text{edges}| \cdot |\text{nodes}|^2 \cdot C)$ [36], where C is the cost of the minimal cut and |edges| and |nodes| are, respectively, the number of edges and nodes in the graph. For the type of graphs considered here, for each additional tuple in the input data set we have to add two edges which would give a worst case complexity of roughly $\mathcal{O}(N^3C)$. However, computation times fit well to a quadratic function meaning that for our signals the scaling behavior is better than the worst case complexity [Fig. 14(a), red curve].

The second case is shown in Fig. 14(b). When the system size is fixed (here, 750 tuples) and the number of labels increases (here, from 10^3 to ca. 10^4) by refining the label grid subsequently, the corresponding run times increase linearly. This is in agreement with the theory behind multilabel graph cut problems [36]. The α expansion offers new labels one by one in a random order until all labels were used and the iteration stops. Thus, the observed linear scaling in the number of labels is also expected from theory.

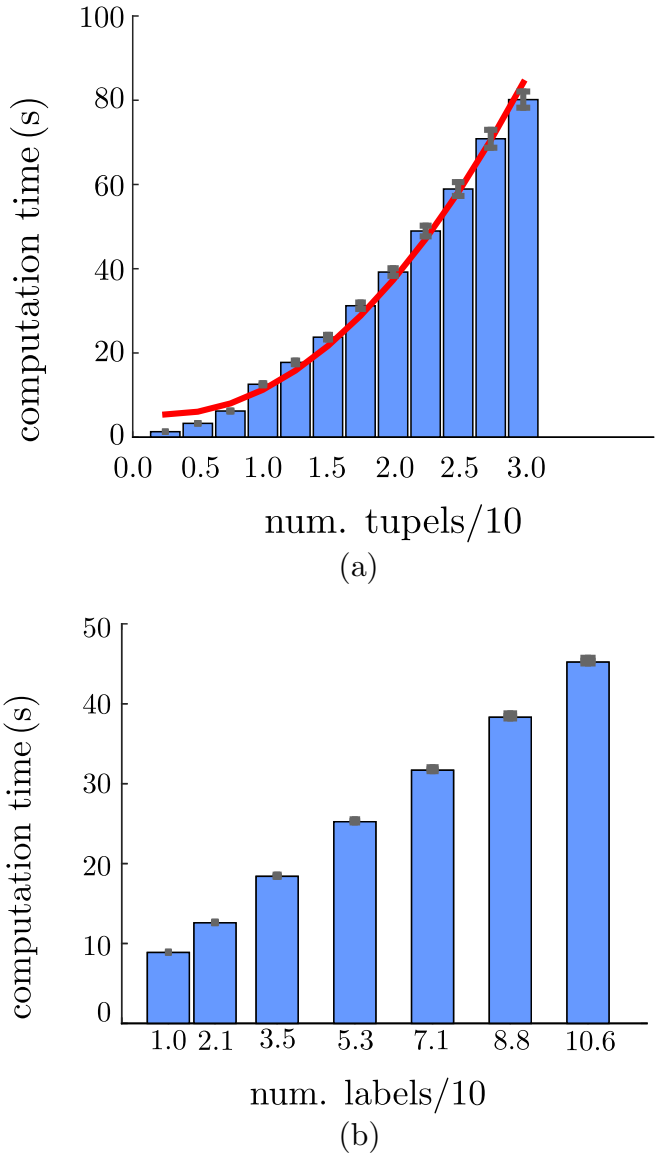


FIG. 14. Mean run time performance of combinatorial clustering stage for 10 simulated noisy step signals. (a) Graph cut computation time versus number of tuples for a label grid of 800 levels. Second order polynomial fit to computation times (red curve). (b) Graph cut computation time versus label grid size for a fixed system size of 750 tuples. Linear scaling of performance with increasing number of levels in the label grid set. The error bars are SEM.

It is important to point out that due to the TVDN compression the expression above is an improvement for this type of step signals (high bandwidth, number of data points $\sim 10^5$, but comparably few steps <1000) compared to the Fourier transform accelerated HMM implementation [22]: $\mathcal{O}(mn^2N\log_2 m)$, where m is the number of position states, n the number of molecular states, and N the number of data points. Moreover, the direct comparison of run times and memory consumption given in the main text shows that our algorithm is advantageous regarding computational resources compared to existing algorithms.

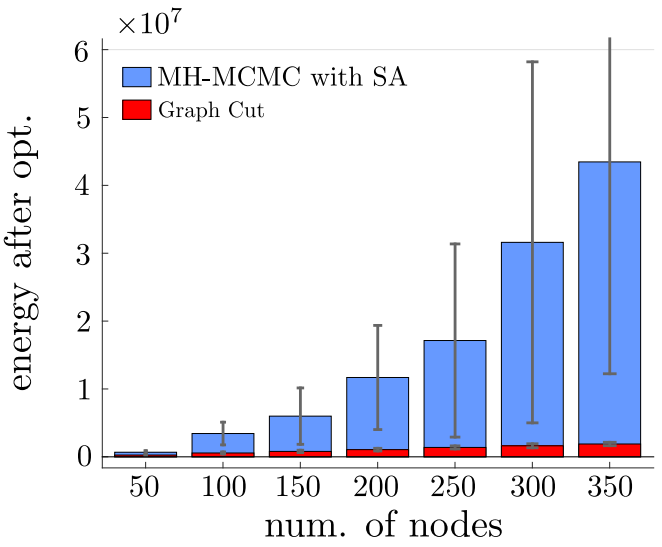


FIG. 15. Graph cut MCMC comparison. Energies of optimal solutions with increasing system size for MCMC and graph cut algorithms.

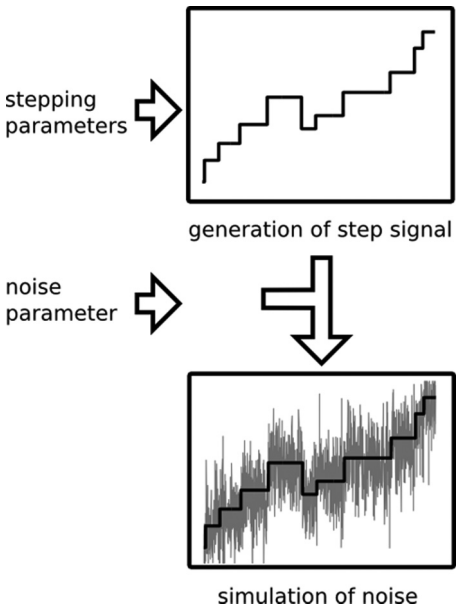


FIG. 16. Step and noise simulation procedure. State transition model and corresponding stepping rates determine the probability distribution from which a piecewise constant signals (first inset) is sampled. In a second step noise is simulated with the piecewise constant signals as the mean (second inset).

7. Comparison of graph cut and Markov chain Monte Carlo

Since Markov chain Monte Carlo (MCMC) methods are standard techniques to optimize an energy functional with Potts model terms like Eq. (7), we compare the graph cut method with a Metropolis-Hastings (MH) sampling and simulated annealing (SA) optimization algorithm [58]. In each iteration we randomly generate a proposal assignment of labels. The new assignment of a site is accepted or rejected according to the standard MH rules. Moreover, a logarithmic temperature schedule is used for SA. The temperature parameter is introduced as commonly done: $p(\mathbf{x}) \propto \exp[-E(\mathbf{x})/T]$. If an accepted proposal has smaller energy than all previous ones, it becomes the new configuration that minimizes Eq. (3). To compare the quality of the step-detection result, we computed the energy, Eq. (3), with prior terms, Eq. (7), for the energy minimizing solutions of graph cut and the MCMC method (Fig. 15). For a system size below 350 tuples, computation times of the graph cut algorithm were always below 10 s. Since MCMC is computationally more complex, longer computation times were used for MCMC, i.e., 45 min, which allowed for 12 iterations of a SA temperature cycle. Each cycle consists of 20 subsequent cooling steps and in each step we iterate through 20 000 proposals. In spite of the significantly higher computational cost, the MCMC solutions always have higher energies compared to graph cut and the excess energy increases for larger systems. This shows that MCMC returns increasingly worse solutions compared to the graph cut technique when the number of input data grows for fixed computation time. As expected, graph cut shows an approximately linear increase in energy with linearly increasing system size.

To conclude, the plain MCMC algorithm used here is conceptually simpler than graph cut but computationally more expansive and also less suited to cluster the denoised steps optimally according to an energy functional. This finding in one dimension is not surprising, since similar observations had been made in 2D image analysis [28].

8. Noisy step simulations

Single base pair steps are typically exceeded by noise fluctuations and most of the time it is not possible to judge by eye whether an algorithm correctly positioned steps. Therefore, simulated data are necessary to show and compare the performance of step-detection algorithms. We generate noisy steps in two stages, as outlined in Fig. 16. First, we generate a piecewise constant signals according to a simplified version of the linear ratchet model of Pol II [59]. This model contains elongation and backtracked states and reproduces the ability to pause [42,43], but does not accurately reflect the temporal order of translocation and other enzymatic processes. During elongation, 1-bp forward steps are generated with an effective rate of k_{elong}. This effective rate includes the process of translocation, nucleoside triphosphate (NTP) insertion, and pyrophosphate release. In our model catalysis, bond formation and PP_i release are summarized by a rate k_3. Furthermore, the NTP-binding net rate is $k_2 = c_{NTP} k_3 / K_D$ and the translocation net rate $k_1 = k_+ k_2/(k_- + k_2)$. c_{NTP} is the NTP concentration, $K_D = 9.2\ \mu$M the dissociation constant, $k_+ = 88$ Hz the forward translocation rate of Pol II, and $k_- = 680$ Hz the backward translocation. The values of these constants are known from experiments [59]. The elongation rate is then determined by $k_{\text{elong}} = (1/k_1 + 1/k_2 + 1/k_3)^{-1}$.

With a rate of $k_{b1} = 5$ Hz the motor makes a backward step of identical size as the forward step and thus enters the backtracked state. The enzyme can further backtrack by a rate $k_b = 1.3$ Hz or return to the original state with a rate $k_f = 1.3$ Hz (Fig. 17).

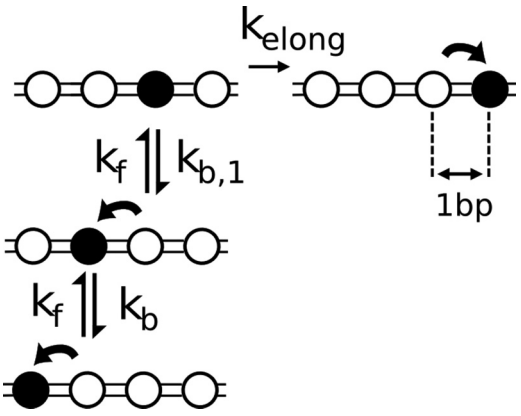


FIG. 17. Simplified stepping model of pol II with an elongation and backtracked states.

The rates corresponding to a forward step (k_+, k_f) or backward step (k_{b1}, k_b, k_-) are modified under external forces according to $k(F) = k(0)\exp[\pm F 0.17\ \text{nm}/(k_bT)]$, where $k_bT = 4.11$ pN nm and the plus sign in the exponent applies to rates of forward steps. Simulations were computed for an assisting force of 6.5 pN. At this force forward and backward diffusion rates are $k_b = 3.8$ Hz, $k_b = 1.0$ Hz, and $k_f = 1.7$ Hz, in accordance with the kinetic model. For numerical simulation purposes the rates above are divided by the simulation's time increment to yield dimensionless quantities.

The transitions between elongation and backtracked states are generated using the Gillespie stochastic simulation scheme [60] for a single enzyme. Dwell times are sampled from an exponential distribution according to the respective rates.

In a second step, we simulated experimental noise including effects of confined Brownian motion of trapped microspheres. To accurately reflect the experiment, we take into account changes in the tether length and in the tether stiffness due to motion of the enzyme. We apply a harmonic description of the trapping potentials and assume that the DNA linker can be described by a spring constant k_{DNA} determined by the wormlike chain model [61].

To formulate the equation of motion of two trapped microspheres tethered by DNA, we choose the coordinate system such that the enzyme moves in the x direction. Furthermore, we assure that drag coefficients γ_i and the trapping stiffness $k_{\text{trap},i}$ are identical in both traps. With this the effective DNA length x can be described by the equation

$$\gamma\dot{x} = -kx + F_T(t), \tag{A31}$$

where $k = k_{\text{trap}} + 2k_{DNA}$, k_{DNA} is the DNA stiffness, and γ is the drag coefficient. $F_T(t)$ is the thermal force which is treated as Gaussian white noise: $\langle F_T(t)\rangle = 0$ and $\langle F_T(t)F_T(t')\rangle = 2k_BT\gamma\delta(t-t')$. Equation (A31) describes a so-called Ornstein-Uhlenbeck process and can be solved and simulated by standard techniques of stochastic differential equations [62], which is shown in the next section. Equation (A31) was derived for the static situation without positional changes. However, a molecular motor which is attached between microspheres by a DNA tether will change the tether length during its activity. Thus, k_{DNA} is also changing and can be computed using the wormlike chain model [61].

In the simulations we use a trap stiffness of $k_{\text{trap}} = 0.25\,\text{pN/nm}$, a drag coefficient of $\gamma = 0.8 \times 10^{-5}$ pN s/nm corresponding to beads with 850 nm diameter, and an initial length of $L = 3$ kbp for the DNA tether.

We simulated a slow, an intermediate, and a fast scenario, which differ by stepping speed, sampling frequency, number of data points, and noise amplitudes. Sampling frequencies and number of data points of the slow scenario are $f = 5$ kHz and $N = 2.5 \times 10^5$ points; for the intermediate scenario, $f = 2$ kHz and $N = 10^5$ points; and for the fast scenario, $f = 1$ kHz and $N = 5 \times 10^4$. The elongation rate k_{elong} of the slow scenario $k_{\text{elong}} = 4.1$ Hz can be expected at a NTP concentration of $c_{NTP} = 7$ mM. Since backtracking becomes more likely at these NTP concentrations, we limited analysis to simulated data that shows a net forward translocation. This excludes analysis of simulated data which exhibit only backtracked states. Elongation rates of the intermediate ($k_{\text{elong}} = 9.1$ Hz) and fast scenario ($k_{\text{elong}} = 25.8$ Hz) are expected at $c_{NTP} = 20$ mM and $c_{NTP} = 1000$ mM, respectively. The standard deviations of noise amplitudes are directly computed from the noisy input data. This is done by subtracting the simulated step signal from the noisy steps and computing the standard deviation of the remaining signal. In both scenarios, slow and intermediate, the computed standard deviation is 5.5 bp at the given sampling frequency. For the fast scenario we choose $N = 5 \times 10^4$ data points and 1-kHz sampling rate. Moreover, in the fast scenario we use higher noise amplitudes with a computed standard deviation of 10.0 bp at the 1-kHz sampling frequency.

Finally, for all three scenarios 25 data sets were simulated and analyzed. Table I gives an overview over the simulation parameters.

9. Simulating beads in a harmonic optical trap

As described above we account for confined Brownian motion of trapped beads in a dual trap optical tweezers. A harmonic description of trapping potentials is applied and we assume the DNA linker can be described by a worm like chain (WLC) model with a spring constant k_{DNA}. In the following we briefly show the derivation of Eq. (A31) and its solution.

TABLE I. Overview of simulation parameters. Shown is elongation rate k_{elong}, corresponding NTP concentration c_{NTP} and rate constants of the backtracking state. Moreover, the standard deviation of noise σ_n, sampling frequency f, and number of data points N is given.

Scenario:	k_{elong}/Hz	c_{NTP}/mM	k_{b1}/Hz	k_b/Hz	k_f/Hz	σ_n/bp	f/kHz	N
Slow	4.1	7	3.8	1.0	1.7	~6	5	2.5×10^5
Intermediate	9.1	20	2.3	1.0	1.7	~6	2	1×10^5
Fast	25.8	1000	2.3	1.0	1.7	~10	1	5×10^4

We focus on the x coordinates of two beads trapped in different optical traps and tethered by DNA. The equation of motion of such a system of reads [63]

$$\boldsymbol{\gamma}\dot{\mathbf{x}} = -\boldsymbol{\kappa}\mathbf{x} + \boldsymbol{F}_T(t), \tag{A32}$$

where $\mathbf{x} = (x_1, x_2)$ is the x coordinate of first and second bead. Furthermore, drag coefficient, stiffness, and thermal force are

$$\boldsymbol{\gamma} = \begin{pmatrix} \gamma_1 & 0 \\ 0 & \gamma_2 \end{pmatrix},$$

$$\boldsymbol{\kappa} = \begin{pmatrix} k_{\text{trap},1} + k_{DNA} & -k_{DNA} \\ -k_{DNA} & k_{\text{trap},2} + k_{DNA} \end{pmatrix},$$

$$\boldsymbol{F}_T(t) = \begin{pmatrix} F_{T,1}(t) \\ F_{T,2}(t) \end{pmatrix}.$$

The thermal force fulfills the Gaussian white noise properties: $\langle \xi_i(t) \rangle = 0$ and $\langle F_{T,i}(t) F_{T,j}(t') \rangle = 2k_B T \gamma \delta_{ij} \delta(t - t')$. The relative coordinate $\tilde{x} = x_2 - x_1$, which is called x in the following, can be simplified by assuming that $\gamma_1 = \gamma_2 = \gamma$ and $k_{\text{trap},1} = k_{\text{trap},2} = k_{\text{trap}}$ to

$$\dot{x} = -2\pi f_c x + \frac{1}{\gamma} F_T, \tag{A33}$$

where $f_c = (k_{\text{trap}} + 2k_{DNA})/2\pi\gamma$ is the corner frequency of the system. $k_{DNA} = k_{DNA}(F, L)$ depends on force and length of the DNA tether and is calculated from the wormlike chain model [64]. During enzyme stepping k_{DNA} has to be updated repeatedly with respect to the external parameters force F and length L. Equation (A33) describes a so-called Ornstein-Uhlenbeck (OU) process and can be solved and simulated by standard techniques of stochastic differential equations [62]. From Eq. (A33) it can be seen that for time scales slower than the corner frequency f_c noise behaves essentially as white noise. For time scales faster than f_c noise rather has characteristics of Brownian motion. In the following, the simulation of an Ornstein-Uhlenbeck process is described. We rewrite Eq. (A33) in Ito form,

$$dx_t = -kx dt + \sqrt{2D} dW_t, \tag{A34}$$

where $k = (\kappa + 2k_{DNA})/\gamma$, $D = k_B T/\gamma$ is the diffusion constant, and dW_t infinitesimally describes Brownian motion. For a finite time interval, $\Delta W_t = \int_{t-h}^{t} dW_{t'}$ describes a standard normal distributed random variable $\mathcal{N}(0,h)$, with standard deviation $\sigma = \sqrt{h}$.

Equation (A34) just describes a Gaussian random variable with mean μ and variance σ^2 [65],

$$x_t \in \mathcal{N}(\mu, \sigma^2) = \mathcal{N}\left[x_{t-1} e^{-kt}, \frac{D}{k}(1 - e^{-2kt}) \right], \tag{A35}$$

and a random path can be straightforwardly simulated starting from an initial position x_0.

10. Details of algorithm comparison

To achieve best results for the three simulation scenarios, we need to tune the external parameters of the t test, the HMM and the EBS algorithm. While the latter is described in the main text (only for the slow scenario we used $\rho_S = 1.5$ instead of $\rho_S = 2$), we briefly explain how to adapt the other two algorithms to yield as many correct steps as possible but also to have a large fraction of correct steps among the found steps.

For the t test a minimum step size of 0.3 nm and a shortest dwell of 10 ms was used. Moreover, the t-test threshold was 0.01, the binomial threshold was 0.005, and the maximal number of iterations was 100.

The HMM analysis was conducted with maximally 100 iterations for maximum likelihood estimation of transition probabilities. More iterations did not give better results and fewer iterations ($\leqslant$10) could not optimize the log-likelihood properly (data not shown). For the slow scenario 85 states were used and for the intermediate and fast scenario we used 140. To prevent memory overflow in the intermediate scenario, we performed boxcar averaging to reduce the number of data points by a factor of two. Furthermore, a grid spacing of 1/2 bp was used which proved to be better than a 1-bp spacing. Since the HMM level grid has to be aligned by using noisy data as an input, a two times smaller grid spacing showed better

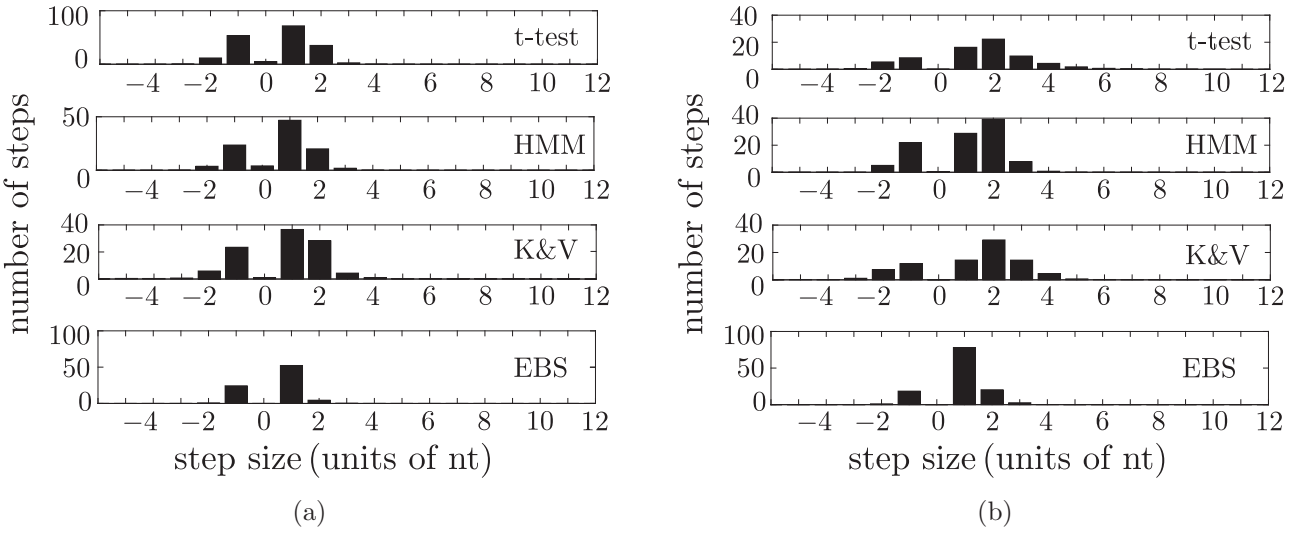


FIG. 18. Histogram of step sizes of the slow scenario (a) and the intermediate scenario (b) for t test, HMM, K & V, and EBS (from upper to lower histogram).

TABLE II. Comparison of computation efficiency of the different step-finding algorithms. ∼900 simulated steps on commodity hardware (i7-2600, 3.6-GHz CPU Ubuntu System, 4-GB memory). Corresponding run times were recorded in MATLAB for the signal of the given size. Peak memory usage, i.e., resident set size (RSS) was measured with Linux's proc information system.

	t test	HMM	K & V	EBS
Data points/10^5	2.5	2.5	2.5	2.5
Run time/s	708	4858	2555	5
Peak RSS/GB	0.17	2.38	0.14	0.15

results. In contrast the situation is advantageous in the case of combinatorial clustering, which constructs the level grid on already denoised data.

The algorithm from Kalafut and Visscher has the great advantage that it works completely without parameters that have to be adjusted by the user.

To complete the performance results of the algorithm comparison (Fig. 8) the step size histograms of the step-detection result for easy and intermediate scenario are given (Fig. 18). For slow stepping rates the step size histograms resemble the simulated step size ±1 bp quite well [Fig. 18(a)]. However, a deviation from the 1-bp steps can be already seen in the intermediate scenario for the *t* test and HMM, Fig. 18(b). For EBS the majority of the detected steps are 1 bp in size in both scenarios. The more difficult situation in the intermediate scenario is reflected by the larger fraction of 2-bp steps [Fig. 18(b), red].

Table II summarizes computational speed and memory consumption of the algorithms for test runs with 100 s of temporal length and rate constants according to the intermediate scenario. We simulated data containing ∼900 simulated steps and successively increased the number of data points by increasing the sampling frequency. For each sampling frequency the standard deviation of noise was constant. EBS processed 2×10^7 data points, simulated with 200-kHz sampling frequency in 4 min. The other algorithms had comparably long run times and we restricted computation times to ∼50 min and memory consumption to a limit of 2.38 GB. EBS can process many more data points at comparably short run time and is essentially limited only by the size of available memory. The high bandwidth signals can be compressed very well by TVDN to a few thousand plateaus, which yields shorter run times for the subsequent CC. In contrast, the *t* test becomes slower at $>10^5$ data points since it has more possibilities to adapt window sizes. A limiting factor in case of a lot of data points for the HMM beside processing speed is memory consumption of the Viterbi reconstruction. Taken together the more efficient computation of EBS compared to the other algorithms allows for the analysis of high-bandwidth data. This in turn can increase the performance of step finding.

To show how much the performance of step detection improves, we use the intermediate scenario but increase the bandwidth (i.e., number of data points) from 2 to 200 kHz. In order to have the same standard deviation of noise at 2-kHz sampling rate, the noise amplitude of the high-bandwidth signal is increased accordingly. This increases precision and recall from (∼30% and ∼60%) at the lower bandwidth to (∼40% and ∼60%) at the higher bandwidth. For CC the same set of parameters is used for low- and high-bandwidth signal (see the Methods section). Due to the very fast denoising stage and the efficient compression to tuples, run times are still below 3 min for 10^7 data points and 200–300 steps.

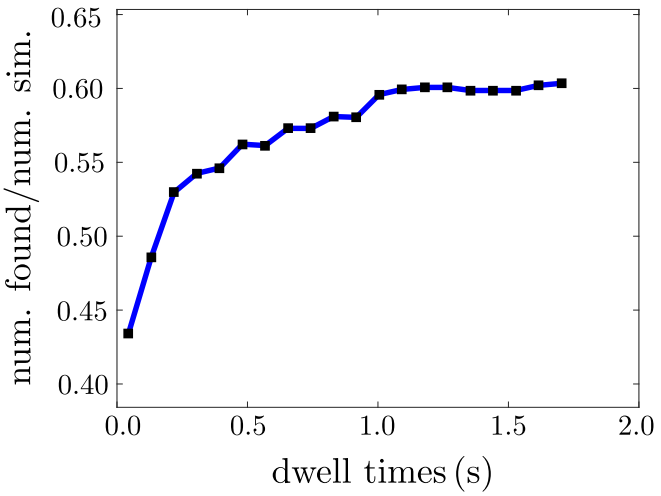


FIG. 19. Cumulative fraction of found steps for the EBS in the intermediate scenario. Plotted is the cumulative number of found steps/simulated steps of the signal plotted in Fig. 4 for different dwell times. The number of detected steps compared to simulated ones is smaller for short dwell time steps. Dwell time histograms with a binning of 87.3 ms were determined for the detected and simulated steps, respectively. The number of detected steps for each dwell time was divided by the corresponding number of simulated steps and cumulatively summed up. In total, 60.5% of the number of simulated steps were found.

11. Remarks on example of TVDN and combinatorial clustering

In order to get temporal information of the missing steps in the example given in the Results and Discussion section (Fig. 4), we compare the dwell time histograms of the simulated and detected steps. The cumulative fraction of found steps for a certain dwell time shows that steps with short dwell times are omitted with higher probability (Fig. 19).

12. Effect of prior information in combinatorial clustering

To analyze the impact of the prior terms and level grid spacing on step-detection quality, we performed CC with different prior potential strength and level grid spacing [Fig. 20(a)]. We varied the prior regularization parameters ρ_S and ρ_P starting from $\rho_S = 0$ to a maximum of $\rho_S = 6$, while the jump height prior parameter was varied simultaneously such that $\rho_P/\rho_S = 12.5$ remained constant. By increasing the prior regularization parameters the precision of step detection can be increased [triangles, Fig. 20(a)]. Furthermore, precision can be improved by choosing a level grid with a spacing of the simulated step size of 1 bp [squares, Fig. 20(a)].

The best result regarding the absolute number of correct steps and a small number of falsely detected steps which lie outside a certain time window around an actual step (see the Methods section) was found for $\rho_S = 4$, $\rho_P = 50$ [Fig. 20(a)].

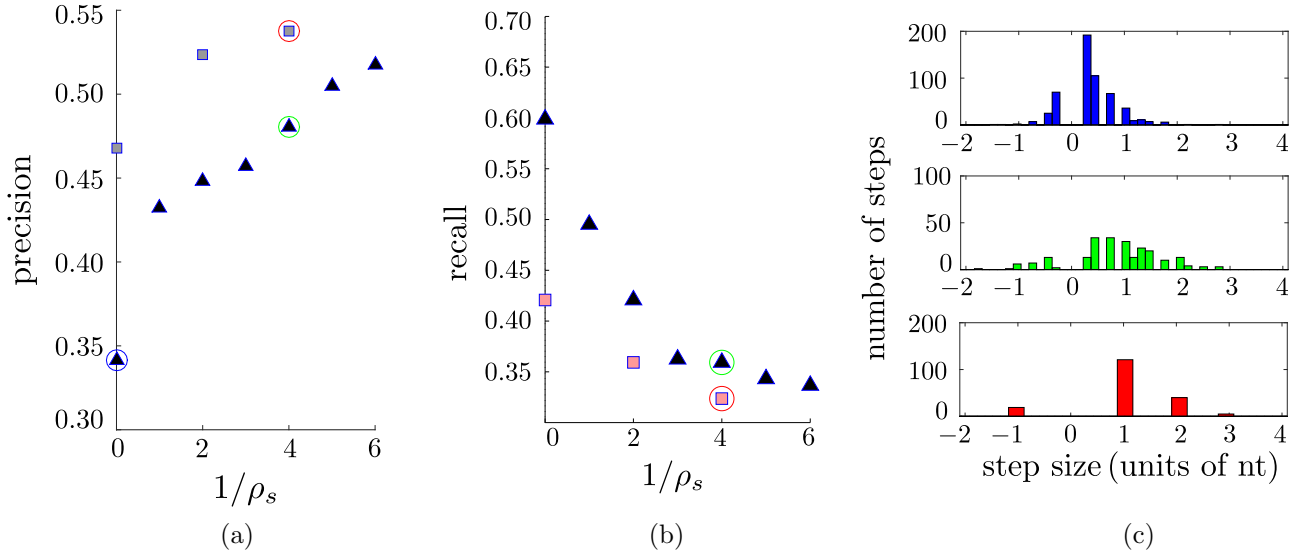


FIG. 20. Prior terms and level grid regularize combinatorial clustering. (a) Relative frequencies of correct steps among the number of detected steps (precision (a), recall (b)) as a function of prior potential strength $1/\rho_S$ ($\rho_S/\rho_P = 0.08$ is kept constant) for simulated data using the intermediate scenario. Shown is the precision for clustering with a level grid of 1/4 bp spacing (black triangles) and with a spacing of 1 bp (gray squares). (c) Step size histograms of detected steps with a label grid of 1/4 bp without prior terms (blue), with a spacing of 1/4 bp and prior terms (green), and with a spacing of 1 bp and prior terms of the same strength (red). The computed precision (a) and recall (b) corresponding to the three histograms is encircled with the respective colors (a).

This increases the number of correct steps from 34% of the steps found at vanishing prior potential to 48% for the 1/4 bp level grid analysis. By increasing the spacing to 1 bp, the detection precision can be improved even further to a fraction of 54% correct steps of the found steps.

The prior potential strength of the smoothing term ρ_S cannot be arbitrarily large since it would remove steps in favor of fewer large steps and thus reduce the number of correctly recovered simulated steps. The variation of the prior term also effects step size histograms [Fig. 20(c)]. When no prior terms are present ($\rho_S = \rho_P = 0$) the detected step size is oftentimes smaller than the simulated step height [Fig. 20(c), top panel]. Optimization of the regularization parameters as well as an increase in level spacing improves the precision of the EBS algorithm, as shown in the histograms of detected step sizes [Fig. 20(c), middle and lower panels].

13. EBS application to experimental data of φ29 bacteriophage

The noisy 2.5-kHz recording of a φ29 bacteriophage (Fig. 21) has the characteristic dependency of the number of denoised steps on the λ regularization parameter of TVDN [Fig. 21(b)] and Algorithm 1 finds the regularization parameter for optimal denoising, λ_h. Based on the TVDN result [red signal, Fig. 21(a)] and the prior information, that the φ29 bacteriophage performs substeps of 2.5 bp [12], a level grid is formed [black lines (a)]. After combinatorial clustering the detected step signal is obtained [blue signal 21(a)].

14. Application of EBS to find pauses in experimental transcription data

In the following, we discuss the determination of pauses in experimental Pol II data as an example of further postprocessing of the detected steps and compare EBS-based pause finding and SGVT on simulated data.

For the simulated Pol II steps dwell times are assigned to a pause when they lead to a backward step. The corresponding pause ends when a forward step brings Pol II back to the elongation state. For the detected steps this criterion also applies; however, unlikely long dwells are also considered as pauses, since the algorithm will not perfectly find all steps present. Given the limited bandwidth (1 kHz), high speed (saturating NTP concentration) of the enzyme and noise (standard deviation ~10 bp) in the traces step-detection performance should be similar to the fast scenario in our algorithm comparison. One can expect that mostly very fast steps are lost (Fig. 19), i.e., fused to large steps. On the other hand, that means that also short backtracks are likely to be skipped and instead a longer dwell time between two forward steps is returned by the algorithm. Nevertheless, these longer dwells can be identified based on statistical hypothesis testing. Assuming that dwell times of forward stepping $\langle\tau_{\text{forward}}\rangle$ follow an exponential waiting time distribution, we calculate the mean dwell time of forward steps to estimate the probability distribution,

$$\langle\tau_{\text{elongation}}\rangle \sim \langle\tau_{\text{forward}}\rangle = \frac{1}{N}\sum_{i=1}^{N}\tau_{\text{forward}}. \tag{A36}$$

Since not all backtracked pauses are discovered, this estimate of $\langle\tau_{\text{elongation}}\rangle$ also contains longer dwell times at a skipped pause. Thus, $\langle\tau_{\text{forward}}\rangle$ can be larger than $\langle\tau_{\text{elongation}}\rangle$ and should be taken as an upper bound for the actual mean waiting time.

Furthermore, we assume that forward steps obey an exponential distribution of the following form:

$$p(\tau) = \frac{1}{\langle\tau\rangle}\exp(-\tau/\langle\tau\rangle). \tag{A37}$$

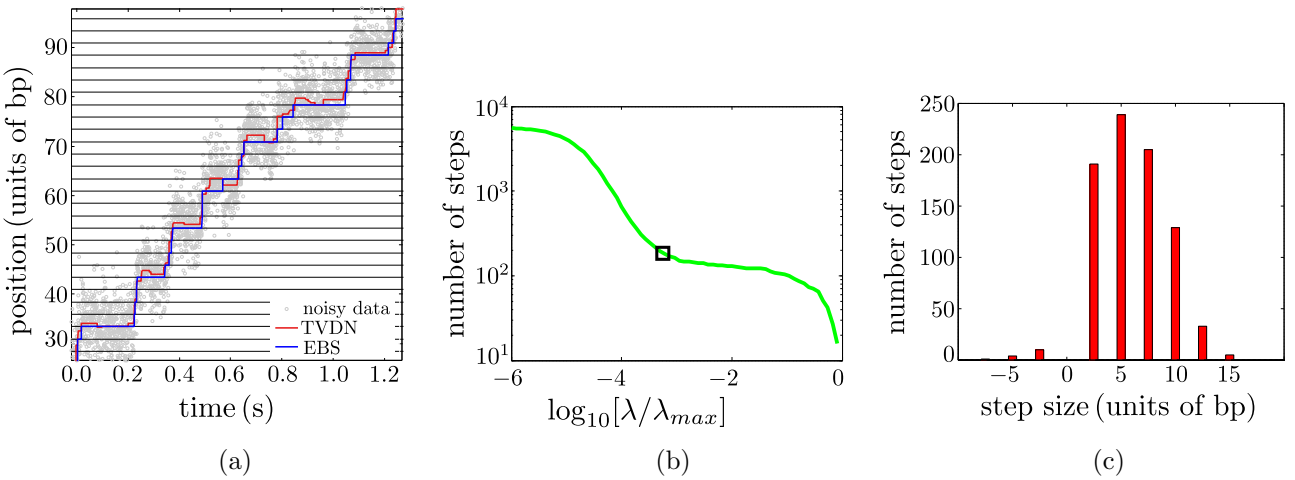


FIG. 21. Detected steps and intermediate results of EBS application to the φ29 example (Results and Discussion). Shown is the noisy data [gray circles (a)], the corresponding λ-characteristic curve [green curve (b)], TVDN with respect to optimal λ_h [black box (b) and red signal (a)], level grid for CC [black horizontal lines (a)], and steps detected by EBS [blue signal (a)]. Panel (c) shows the sum of detected step sizes of a set of 40 φ29 measurements.

Under these assumptions we can define a confidence level to discriminate between normal dwell times of elongation and unlikely long dwell times which are caused by undetected backtracks.

The confidence level can be adjusted by comparing recovered pauses to simulated backtracked pauses. A good compromise is found when most of the pauses are recovered and none or only very few of them are wrongly found.

To this end we simulated 10 data sets with stepping rates and sampling frequency of the fast scenario and a computed noise amplitude of ~6 bp. The simulated data is processed by EBS and the paused regions are identified according to the criterion described above. We also identify paused regions by SGVT with a threshold of two standard deviations of the pause peak, as described in the Methods section. SGVT sometimes returns very short pauses which are not related to simulated ones and are presumably caused by high noise affecting the filtered signal. Thus, we exclude pauses smaller than 10 ms in the SGVT analysis. Pauses found by EBS were always larger than 10 ms and thus there was no need for such an additional postprocessing step. For each detected pause we identify if it is a correctly found one by checking whether it coincides with a simulated pause. We also take into account that either two detected pauses which are close but separated could overlap with a simulated pause or a single detected pause could cover two very close but separated simulated ones. Having identified how many pauses are correct, we can compute the recall (i.e., the number of correctly found pauses divided by the number of simulated pauses), the precision (i.e., the number of correctly found pauses divided by the number of found pauses), and the false discovery rate (FDR, i.e., number of wrongly found pauses divided by the number of found pauses). Moreover, we

TABLE III. Detection of short and long backtracks in simulated data by EBS and the Savitzky-Golay (SG) filter. Shown is the number of correctly detected backtracks divided by the number of simulated backtracks (recall), the number of correctly detected backtracks divided by the number of found backtracked regions (precision), and the false discovery rate (FDR, number of false positives divided by number of found backtracked regions). Moreover, the total length of detected backtracks divided by the total length of simulated backtracks is given. Backtracks with a detected duration <10 ms were excluded. The uncertainties for recall, precision, and FDR are SEM.

	Recall/%	Precision/%	FDR/%	Total length/%
Short pauses				
SG filter	38 ± 7	57 ± 7	43 ± 8	70
EBS	61 ± 4	98 ± 2	8 ± 2	91
Long pauses				
SG filter	98 ± 1	100	0	94
EBS	100	100	0	113

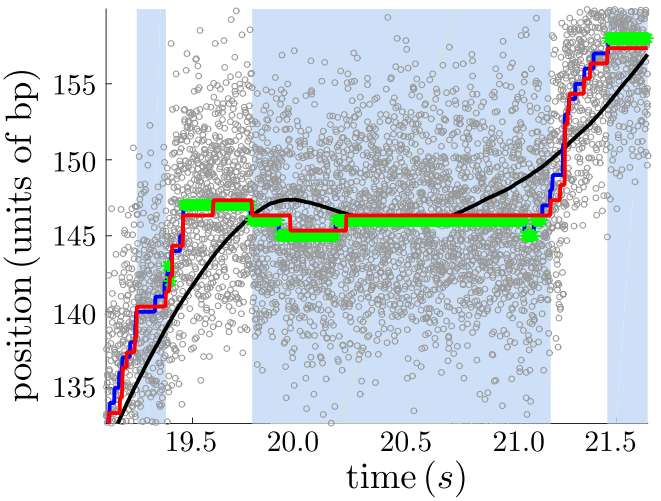


FIG. 22. Backtracked pause detection in simulated data. Shown are the noisy input signal (circles), the simulated step signal (blue), the Savitzky-Golay filtered signal (black), and the detected step signal from EBS (red). Pauses in simulated data are highlighted in green and paused regions in step detected data are indicated by the blue shaded areas.

compare the total cumulated length of all detected pauses to the total cumulated length of simulated ones. This value is relevant since a correct determination of the total length of pauses is important for determining pause-free velocities which are computed by excluding the paused intervals from the measured data. Table III shows mean and standard deviation of recall, precision, FDR, and total length for long ($t > t_p$) and short pauses ($t < t_p$), where the threshold for determining a long pause is $t_p = 0.8$ s (see the Results and Discussion section). Although in the fast scenario step-detection performance is inappropriate for further dwell time analysis, finding pauses still works well (Fig. 22 and Table III).

[1] A. D. Mehta, R. S. Rock, M. Rief, J. A. Spudich, M. S. Mooseker, and R. E. Cheney, Myosin-V is a processive actin-based motor, Nature (London) **400**, 590 (1999).

[2] N. J. Carter and R. A. Cross, Mechanics of the kinesin step, Nature (London) **435**, 308 (2005).

[3] E. A. Abbondanzieri, W. J. Greenleaf, J. W. Shaevitz, R. Landick, and S. M. Block, Direct observation of base-pair stepping by RNA polymerase, Nature (London) **438**, 460 (2005).

[4] E. A. Galburt, S. W. Grill, A. Wiedmann, L. Lubkowska, J. Choy, E. Nogales, and C. Bustamante, Backtracking determines the force sensitivity of RNAP II in a factor-dependent manner, Nature (London) **446**, 820 (2007).

[5] J. Michaelis and B. Treutlein, Single-molecule studies of RNA polymerses, Chem. Rev. **113**, 8377 (2013).

[6] J. Michaelis, A. Muschielok, J. Andrecka, W. Kgel, and J. R. Moffitt, DNA based molecular motors, Phys. Life Rev. **6**, 250 (2009).

[7] A. Yildiz and P. R. Selvin, Fluorescence imaging with one nanometer accuracy: Application to molecular motors, Acc. Chem. Res. **38**, 574 (2005).

[8] K. C. Neuman and A. Nagy, Single-molecule force spectroscopy: optical tweezers, magnetic tweezers and atomic force microscopy, Nat. Methods **5**, 491 (2008).

[9] A. B. Kolomeisky and M. E. Fisher, Molecular motors: A theorists perspective, Annu. Rev. Phys. Chem. **58**, 675 (2007).

[10] I. Heller, T. P. Hoekstra, G. A. King, E. J. Peterman, and G. J. Wuite, Optical tweezers analysis of DNA–protein Complexes, Chem. Rev. **114**, 3087 (2014).

[11] Y. R. Chemla, K. Aathavan, J. Michaelis, S. Grimes, P. J. Jardine, D. L. Anderson, and C. Bustamante, Mechanism of force generation of a viral DNA packaging motor, Cell **122**, 683 (2005).

[12] J. R. Moffitt, Y. R. Chemla, K. Aathavan, S. Grimes, P. J. Jardine, D. L. Anderson, and C. Bustamante, Intersubunit coordination in a homomeric ring ATPase, Nature (London) **457**, 446 (2009).

[13] G. Chistol, S. Liu, C. L. Hetherington, J. R. Moffitt, S. Grimes, P. J. Jardine, and C. Bustamante, High degree of coordination and division of labor among subunits in a homomeric ring ATPase, Cell **151**, 1017 (2012).

[14] S. A. McKinney, C. Joo, and T. Ha, Analysis of single-molecule FRET trajectories using hidden markov modeling, Biophys. J. **91**, 1941 (2006).

[15] J. Opfer and K. E. Gottschalk, Identifying discrete states of a biological system using a novel step detection algorithm, PLoS One **7**, e45896 (2012).

[16] L. Venkataramanan and F. J. Sigworth, Applying hidden Markov models to the analysis of single ion channel activity, Biophys. J. **82**, 1930 (2002).

[17] L. S. Milescu, A. Yildiz, P. R. Selvin, and F. Sachs, Extracting dwell time sequences from processive molecular motor data, Biophys. J. **91**, 3135 (2006).

[18] B. C. Carter, M. Vershinin, and S. P. Gross, A comparison of step-detection methods: How well can you do? Biophys. J. **94**, 306 (2008).

[19] M. A. Little and N. S. Jones, Generalized methods and solvers for noise removal from piecewise constant signals: Part I Background theory, Proc. R. Soc. London, Ser. A **467**, 3088 (2011).

[20] M. A. Little, B. C. Steel, F. Bai, Y. Sowa, T. Bilyard, D. M. Mueller, and N. S. Jones, Steps and bumps: Precision extraction of discrete states of molecular machines, Biophys. J. **101**, 477 (2011).

[21] L. S. Milescu, A. Yildiz, P. R. Selvin, and F. Sach, Maximum likelihood estimation of molecular motor kinetics from staircase dwell-time sequences, Biophys. J. **91**, 1156 (2006).

[22] F. E. Mllner, S. Syed, P. R. Selvin, and F. J. Sigworth, Improved hidden Markov models for molecular motors. 1. Basic theory, Biophys. J. **99**, 3684 (2010).

[23] G. D. Forney, The viterbi algorithm, Proc. IEEE **61**, 268 (1973).

[24] B. Kalafut and K. Visscher, An objective, model-independent method for detection of non-uniform steps in noisy signals, Comput. Phys. Commun. **179**, 716 (2008).

[25] J. W. Kerssemakers, E. L. Munteanu, L. Laan, T. L. Noetzel, M. E. Janson, and M. Dogterom, Assembly dynamics of microtubules at molecular resolution, Nature (London) **442**, 709 (2006).

[26] N. R. Zhang and D. O. Siegmund, A modified Bayes information criterion with applications to the analysis of comparative genomic hybridization data, Biometrics **63**, 22 (2007).

[27] M. A. Little and N. S. Jones, Sparse Bayesian step-filtering for high-throughput analysis of molecular machine dynamics, arXiv:1003.5535 (2010).

[28] Y. Boykov, O. Veksler, and R. Zabih, Fast approximate energy minimization via graph cuts, IEEE Trans. Pattern Anal. Mach. Intell. **23**, 1222 (2001).

[29] L. I. Rudin, S. Osher, and E. Fatemi, Nonlinear total variation based noise removal algorithms, Physica D (Amsterdam, Neth.) **60**, 259 (1992).

[30] S. P. Boyd and L. Vandenberghe, *Convex Optimization* (Cambridge University Press, Cambridge, U.K., New York, 2004).

[31] L. Condat, A direct algorithm for 1-D total variation denoising, IEEE Signal Process. Lett. **20**, 1054 (2013).

[32] R. T. Rockafellar, *Convex Analysis*, Princeton Landmarks in Mathematics (Princeton University Press, Princeton, NJ, 1992).

[33] C. R. Vogel and M. E. Oman, Iterative methods for total variation denoising, SIAM J. Sci. Comput. **17**, 227 (1996).

[34] A. Beck and M. Teboulle, Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems, IEEE Trans. Image Process. **18**, 2419 (2009).

[35] S. Z. Li, *Markov Random Field Modeling in Image Analysis* (Springer Science & Business Media, New York, 2009).

[36] V. Kolmogorov, Graph based algorithms for scene reconstruction from two or more views, PhD. thesis, Cornell University, 2003.

[37] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity* (Courier Corporation, North Chelmsford, MA, 1998).

[38] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov, Fast approximate energy minimization with label costs, Int. J. Comput. Vision **96**, 1 (2010).

[39] V. Kolmogorov and C. Rother, Minimizing nonsubmodular functions with graph cuts-a review, IEEE Trans. Pattern Anal. Mach. Intell. **29**, 1274 (2007).

[40] R. B. Potts, Some generalized order-disorder transformations, Math. Proc. Cambridge Philos. Soc. **48**, 106 (1952).

[41] C. Rother, S. Kumar, V. Kolmogorov, and A. Blake, Digital tapestry [automatic image synthesis], *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (IEEE, Piscataway, NJ, 2005), Vol. 1, pp. 589–596.

[42] N. Komissarova and M. Kashlev, RNA polymerase switches between inactivated and activated states by translocating back and forth along the DNA and the RNA, J. Biol. Chem. **272**, 15329 (1997).

[43] J. W. Shaevitz, E. A. Abbondanzieri, R. Landick, and S. M. Block, Backtracking by single RNApolymerase molecules observedat near-base-pair resolution, Nature (London) **426**, 684 (2003).

[44] K. C. Neuman, E. A. Abbondanzieri, R. Landick, J. Gelles, and S. M. Block, Ubiquitous transcriptional pausing is independent of RNA polymerase backtracking, Cell **115**, 437 (2003).

[45] A. Lisica, C. Engel, M. Jahnel, É. Roldán, E. A. Galburt, P. Cramer, and S. W. Grill, Mechanisms of backtrack recovery by RNA polymerases I and II, PNAS **113**, 2946 (2016).

[46] M. Depken, E. A. Galburt, and S. W. Grill, The origin of short transcriptional pauses, Biophys. J. **96**, 2189 (2003).

[47] L. Bai, R. M. Fulbright, and M. D. Wang, Mechanochemical Kinetics of Transcription Elongation, Phys. Rev. Lett. **98**, 068103 (2007).

[48] B. Treutlein, A. Muschielok, J. Andrecka, A. Jawhari, C. Buchen, D. Kostrewa, and J. Michaelis, Dynamic architecture of a minimal RNA polymerase II open promoter complex, Mol. Cell **46**, 136 (2012).

[49] H. Isack and Y. Boykov, Energy-based geometric multi-model fitting, Int. J. Comput. Vision **97**, 123 (2012).

[50] S. G. Arunajadai and W. Cheng, Step detection in single-molecule real time trajectories embedded in correlated noise, PLoS One **8**, e59279 (2013).

[51] D. Donoho and J. Tanner, Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing, Philos. Trans. R. Soc. London A **367**, 4273 (2009).

[52] D. Donoho, A. Maleki, and A. Montanari, The noise-sensitivity phase transition in compressed sensing, IEEE Trans. Inf. Theory **57**, 6920 (2011).

[53] E. Pfitzner, C. Wachauf, F. Kilchherr, B. Pelz, W. M. Shih, M. Rief, and H. Dietz, Rigid DNA beams for high-resolution single-molecule mechanics, Angew. Chem. **52**, 7766 (2013).

[54] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces* (Springer, Heidelberg, 2011).

[55] D. J. Rose, An algorithm for solving a special class of tridiagonal systems of linear equations, Commun. ACM **12**, 234 (1969).

[56] V. Kolmogorov and R. Zabin, What energy functions can be minimized via graph cuts? IEEE Trans. Pattern Anal. Mach. Intell. **26**, 147 (2004).

[57] Y. Boykov and V. Kolmogorov, An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision, IEEE Trans. Pattern Anal. Mach. Intell. **26**, 1124 (2004).

[58] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, Optimization by simulated annealing, Science **220**, 671 (1983).

[59] M. Dangkulwanich, T. Ishibashi, S. Liu, M. L. Kireeva, L. Lubkowska, M. Kashlev, and C. J. Bustamante, Complete dissection of transcription elongation reveals slow translocation of RNA polymerase II in a linear ratchet mechanism, eLife **2**, e00971 (2013).

[60] D. T. Gillespie, Exact Stochastic Simulation of Coupled Chemical Reactions, J. Phys. Chem. **8**, 1, 25 (1977).

[61] E. Siggia, S. Smith, C. Bustanmente, and J. Marko, Entropic elasticity of λ-phage DNA, Science **265**, 1599 (1994).

[62] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations* (Springer Science & Business Media, Berlin Heidelberg, 1992).

[63] J. R. Moffitt, Y. R. Chemla, D. Izhaky, and C. Bustamante, Differential detection of dual traps improves the spatial resolution of optical tweezers, Proc. Natl. Acad. Sci. USA **103**, 9006 (2006).

[64] J. F. Marko, and E. D. Siggia, Stretching DNA, Macromolecules **28**, 8759 (1995).

[65] D. L. Ermak and J. A. McCammon, Brownian dynamics with hydrodynamic interactions, J. Chem. Phys. **69**, 1352 (1978).