Detection of events in single molecule data

Tanuj Aggarwal, Donatello Materassi, Thomas Hays & Murti Salapaka

Supplementary Note

Modeling probe dynamics and noise process

The objective of any measurement system is to sense physical quantities (input signal) of interest and give a corresponding output signal. Often, output signal is assumed to be a direct representation of the input, that is, input signal is estimated by linear scaling of the output signal. However, this is not true in general and therefore the transformation from physical units to measurement units is described by a dynamical model. A generic discrete time dynamical model, relating input and output is described by the following

$$p_k + a_1 p_{k-1} + \cdots + a_l p_{k-1} = b_0 q_k + b_1 q_{k-1} + \cdots + b_m q_{k-m} \tag{1}$$

where $p = (p_0, p_1, \ldots)$ represents the output signal of the measurement system due to an input signal $q = (q_0, q_1, \cdots)$. k is the time/sample index. Thus, the measurement system processes or "filters" the input signal q and provides the output signal p. **Supplementary Fig. 1** shows a block diagram depicting various input sources and their filtering by the system dynamics. Represent the time shift operator by D, with $(D^r x)_k = x_{k-r}$, then the above dynamical model can be

symbolically represented as

$$p = H(D)q, \text{ where } H(D) = \frac{b_0 + b_1 D + \cdots + b_m D^m}{1 + a_1 D + \cdots + a_l D^l}, \tag{2}$$

which is a representation analogous to the $\mathcal{Z}$-transform. The above description includes all systems that are linear, time invariant, causal and finite dimensional which is a large class of systems that include, for example, optical tweezers and AFMs. As an example, a model for optical tweezers in the above form is developed here, where, the measurement system dynamics in continuous-time is described by

$$\gamma \dot{x_b} + k\left(x_T - x_b\right) = f$$

where γ is the drag coefficient of the trapped bead, k is the trap stiffness, x_b is the bead position, x_T is the trap position and f is the external force on the bead. In studies of motor proteins using optical tweezers, external force, f has two components. One is the force applied by the motor, f_m and the other is random force, n' due to the thermal bath. f_m is generated when the motor is stretched and is assumed to admit the following description:

$$f_m = k_m(x_m - x_b)$$

where k_m is the motor stiffness and x_m is the position of the motor beyond its rest length. In controlled force studies, f_m is regulated at a constant value, f_0, by modulating x_T. The popular choice of control is to have $x_T = x_b - \frac{f_0}{k}$. This leads to the continuous-time dynamics

$$\gamma \dot{x}_b + k_m x_b = k_m x_m + f_0 + n'.$$

As f_0 is a constant, it only affects the steady state solution of x_b. Therefore, without loss of generality, the dynamics, once discretized in time, leads to the difference equation

$$\underbrace{x_{b,k}}_{p_k} = \underbrace{\frac{k_m T_s}{(\gamma + k_m T_s) - \gamma D}}_{H(D)} \underbrace{\left(x_{m,k} + \frac{n'}{k_m} \right)}_{q_k} \tag{3}$$

where T_s is the sampling time of the measurement system and discretization method used is described by the transformation $\dot{x} \rightarrow \frac{x_k - x_{k-1}}{T_s} = \frac{1-D}{T_s} x_k$. Thermal noise n' is a white process with a power spectral density $4K_B T\gamma$ where K_B is the Boltzmann's constant, and T is the absolute temperature. Note that **Eq. 3** has the same form as **Eq. 2**. Note that q in **Eq. 3** is the sum of two signals: signal of interest, $x := x_{m,k}$ and noise, $n := \frac{n'}{k_m}$. Using the linear dynamics of the system, p can be decomposed into two signals as

$$p_k = z_k + w_k \text{ where, } z = Hx \text{ and } w = Hn, \tag{4}$$

where z is the output of the measurement system when input is only the signal x, whereas w is the output of the measurement system when input is the noise, n. In addition, typically p is corrupted during measurements and thus the measured data is given by

$$y_k = p_k + \nu_k \tag{5}$$

where ν_k is also often modeled as a white noise process with known power that can be experimentally determined. As an application to step detection, additional modeling assumptions are made about the input signal, x, that it is a staircase function generated by a sequence of steps. Thus

$$x_{k+1} = x_k + u_k. \tag{6}$$

Note that without any further assumptions on u_k, **Eq. 6** is merely stating that $x_{k+1} - x_k := u_k$ and thus poses no further constraints on the applicability of the model. The complete model of

the measurement system is then described by **Eq. 2**, **4**, **5** and **6** that is depicted in **Supplementary Fig. 1**.

Cost function derivation for a general dynamical model

Step detection methodology, as outlined int the main article, needs a cost function formulation for a candidate step function. The step function that minimizes the chosen cost function is the optimal estimate for a particle iteration of the algorithm. The formulation of cost function here is for a more general setting than what is presented in the main article. For the remainder of the article, stochastic variables will be represented by bold characters and the normal character will represent a particular realization of the corresponding stochastic variable. Summarizing the model description (**Eq. 4-6**), it follows that

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \boldsymbol{u}_k \\ \boldsymbol{z}_k &= \sum_{i=0}^{m} b_i \boldsymbol{x}_{k-i} - \sum_{j=1}^{l} a_j \boldsymbol{z}_{k-j} \ , \quad \boldsymbol{w}_k = \sum_{i=0}^{m} b_i \boldsymbol{n}_{k-i} - \sum_{j=1}^{l} a_j \boldsymbol{w}_{k-j} \\ \boldsymbol{y}_k &= \boldsymbol{z}_k + \boldsymbol{w}_k + \boldsymbol{\nu}_k. \end{aligned} \tag{7}$$

with the assumption that the noise process, $\boldsymbol{n}$ and $\boldsymbol{\nu}$ are zero mean Gaussian white noise with variance σ_n^2 and σ_ν^2 respectively. An intuitive strategy to identify the location and size of steps that

occur in x, by analyzing measured data y, involves solving the following optimization problem

$$\begin{aligned} &\min_{(u_0,u_1,\ldots,u_{N-1})} \sum_{k=0}^{N-1} \underbrace{(y_{k+1}-z_{k+1})^2}_{\text{Quadratic Error}} + \underbrace{W(u_k)}_{\text{Penalty}} \\ &\text{subject to :} \\ &x_{k+1} = x_k + u_k \ ; \ x_0 = 0 \\ &z_k = \sum_{i=0}^{m} b_i x_{k-i} - \sum_{j=1}^{l} a_j z_{k-j} \ ; \ z_0 = 0 \end{aligned} \tag{8}$$

where the data record $(y_1, \ldots, y_N)$ is available and $W(u_k)$ is a penalty term, that penalizes every nonzero choice of u_k. Evidently, the quadratic term strives to choose u such that the data y is interpolated well while the penalty term provides a means to isolate the effect of noise terms w and ν from the eventual fit, $(u_0, \cdots, u_{N-1})$ that also determines $(x_1, \ldots, x_N)$. $W(u_k)$ also provides a means to incorporate a relative preference of one size of the step over another. Taking as an example, the detection of steps in kinesin motor data, $W(u_k)$ can be shaped such that it is small when u_k is close to 8 nm and thus, these steps will be favored over other step sizes. In the algorithm developed here, such shaping is tuned automatically as an integral part of the detection methodology.

It is shown in this article that the heuristic approach above is similar to maximum likelihood sequence estimation, $\hat{x}_1^N$ of sequence x_1^N from given measurement sequence y_1^N, that is obtained by determining

$$\hat{x}_1^N = \arg\max_{x_1^N} p_{\boldsymbol{x}_1^N|\boldsymbol{y}_1^N}(x_1^N|y_1^N) \tag{9}$$

where, $x_1^N := \{x_1, x_2, ..., x_N\}$, $p_{\boldsymbol{x}}(x)$ denotes the probability density function (p.d.f.) of random variable $\boldsymbol{x}$, $p_{\boldsymbol{x}|\boldsymbol{y}}(x|y)$ denotes the p.d.f. of $\boldsymbol{x}$ given $\boldsymbol{y} = y$. Thus in **Eq. 9**, the optimal choice

$\hat{x}_1^N$ is sought that is the most probable input sequence given that the measured sequence is $y_1^N = (y_1, \ldots, y_N)$. Note that by applying Bayes' rule [1] it follows that

$$\hat{x}_1^N = \arg\max_{x_1^N} \frac{p_{\boldsymbol{x}_1^N, \boldsymbol{y}_1^N}(x_1^N, y_1^N)}{p_{\boldsymbol{y}_1^N}(y_1^N)} = \arg\max_{x_1^N} p_{\boldsymbol{y}_1^N|\boldsymbol{x}_1^N}(y_1^N|x_1^N) p_{\boldsymbol{x}_1^N}(x_1^N). \tag{10}$$

By further application of Bayes' rule, it can be shown that

$$\hat{x}_1^N = \arg\max_{x_1^N} \left\{ \prod_{k=1}^{N} p_{\boldsymbol{y}_k|\boldsymbol{y}_1^{k-1}, \boldsymbol{x}_1^N}(y_k|y_1^{k-1}, x_1^N) p_{\boldsymbol{x}_k|\boldsymbol{x}_1^{k-1}}(x_k|x_1^{k-1}) \right\} \tag{11}$$

where

$$p_{\boldsymbol{y}_1|\boldsymbol{y}_1^0, \boldsymbol{x}_1^N}(y_1|y_1^0, x_1^N) := p_{\boldsymbol{y}_1|\boldsymbol{x}_1^N}(y_1|x_1^N)$$

$$p_{\boldsymbol{x}_1|\boldsymbol{x}_1^0}(x_1|x_1^0) := p_{\boldsymbol{x}_1}(x_1).$$

From the input-output model in **Eq. 7** it follows that,

$$\boldsymbol{y}_k = \boldsymbol{z}_k + \boldsymbol{w}_k + \boldsymbol{\nu}_k$$

$$= \underbrace{\sum_{i=0}^{m} b_i \boldsymbol{x}_{k-i}}_{\bar{\boldsymbol{x}}_k} + \underbrace{\sum_{i=0}^{m} b_i \boldsymbol{n}_{k-i}}_{\bar{\boldsymbol{n}}_k} - \underbrace{\sum_{j=1}^{l} a_j \boldsymbol{y}_{k-j}}_{\bar{\boldsymbol{y}}_k} + \underbrace{\sum_{j=1}^{l} a_j \boldsymbol{\nu}_{k-j}}_{\bar{\boldsymbol{\nu}}_k} + \boldsymbol{\nu}_k.$$

From the last equation, it can be seen that if $\boldsymbol{y}_1^{k-1} = y_1^{k-1}$ and $\boldsymbol{x}_1^{k-1} = x_1^{k-1}$ then $\boldsymbol{y}_k = \bar{x}_k + \bar{\boldsymbol{n}}_k - \bar{y}_k + \bar{\boldsymbol{\nu}}_k + \boldsymbol{\nu}_k$. Therefore, the distribution of $\boldsymbol{y}_k$ is dictated by the random variable, $\bar{\boldsymbol{n}}_k + \bar{\boldsymbol{\nu}}_k + \boldsymbol{\nu}_k$. When $\boldsymbol{y}_k = y_k$, then $\bar{\boldsymbol{n}}_k + \bar{\boldsymbol{\nu}}_k + \boldsymbol{\nu}_k = y_k - \bar{x}_k + \bar{y}_k$. It follows that,

$$p_{\boldsymbol{y}_k|\boldsymbol{y}_1^{k-1}, \boldsymbol{x}_1^N}(y_k|y_1^{k-1}, x_1^N) = p_{\bar{\boldsymbol{n}}_k + \bar{\boldsymbol{\nu}}_k + \boldsymbol{\nu}_k}(y_k - \bar{x}_k + \bar{y}_k) = \frac{1}{\sqrt{2\pi\bar{\sigma}_n^2}} \exp\left(-\frac{(y_k - \bar{x}_k + \bar{y}_k)^2}{2\bar{\sigma}_n^2} \right), \text{ where}$$

$$\bar{\sigma}_n^2 := \sum_{i=0}^{m} b_i^2 \sigma_n^2 + \sum_{j=1}^{l} a_j^2 \sigma_\nu^2$$

which follows from the fact that $\boldsymbol{n}_i$, $\boldsymbol{\nu}_j$ are Gaussian, i.i.d. (independent and identically distributed) noise sources. Note that in **Eq. 11**, $p_{\boldsymbol{x}_k|\boldsymbol{x}_1^{k-1}}(x_k|x_1^{k-1}) = p_{\boldsymbol{x}_k|\boldsymbol{x}_{k-1}}(x_k|x_{k-1})$ and as $\boldsymbol{u}_{k-1} = \boldsymbol{x}_k - \boldsymbol{x}_{k-1}$, it follows that $p_{\boldsymbol{x}_k|\boldsymbol{x}_{k-1}}(x_k|x_{k-1}) = p_{\boldsymbol{u}_{k-1}}(x_k - x_{k-1})$. In absence of a-priori knowledge and for the purpose of initializing the algorithm, a uniform distribution for $p_{\boldsymbol{x}_1}(x_1)$ is assumed. Likewise, it is assumed that $x_0 := x_1$ and $p_{\boldsymbol{u}_0}(x_1 - x_0) := 1$, from which it follows that,

$$
\begin{aligned}
\hat{x}_1^N &= \arg\max_{x_1^N} \left\{ \prod_{k=1}^{N} p_{\boldsymbol{y}_k|\boldsymbol{y}_1^{k-1},\boldsymbol{x}_1^N}(y_k|y_1^{k-1}, x_1^N) p_{\boldsymbol{x}_k|\boldsymbol{x}_1^{k-1}}(x_k|x_1^{k-1}) \right\} \\
&= \arg \max_{\substack{x_1^N \\ u_k = x_{k+1} - x_k}} \prod_{k=1}^{N} \frac{\exp\left(-(y_k - \bar{x}_k + \bar{y}_k)^2 / 2\bar{\sigma}_n^2\right)}{\sqrt{2\pi\bar{\sigma}_n^2}} p_{\boldsymbol{u}_{k-1}}(u_{k-1}) \qquad (12) \\
&= \arg \min_{\substack{x_1^N \\ u_k = x_{k+1} - x_k}} \sum_{k=1}^{N} \frac{(y_k - \bar{x}_k + \bar{y}_k)^2}{2\bar{\sigma}_n^2} - \log p_{\boldsymbol{u}_{k-1}}(u_{k-1}) \qquad (13) \\
&= \arg \min_{\substack{x_1^N \\ u_k = x_{k+1} - x_k}} \sum_{k=1}^{N} (y_k - \bar{x}_k + \bar{y}_k)^2 - 2\bar{\sigma}_n^2 \log p_{\boldsymbol{u}_{k-1}}(u_{k-1}) Viterbialgorithm \\
&= \arg \min_{\substack{x_1^N \\ u_k = x_{k+1} - x_k}} \sum_{k=0}^{N-1} (y_{k+1} - \bar{x}_{k+1} + \bar{y}_{k+1})^2 + W(u_k) \\
&= \arg \min_{\substack{x_1^N \\ u_k = x_{k+1} - x_k}} \sum_{k=0}^{N} h_k(x_{k-m+1}^k, u_k) + W(u_k) \qquad (14)
\end{aligned}
$$

where

$$
W(u_k) := -2\bar{\sigma}_n^2 \log p(u_k), \;\; h_k := (y_{k+1} - \bar{x}_{k+1} + \bar{y}_{k+1})^2 .
$$

h_k does not have explicit dependence on x_{k+1} because x_{k+1} is expressed in terms of x_k and u_k. The structure of cost function in Eq. 14 is similar to one in **Eq. 8** for appropriate choice of weighting function W. Thus, the intuition driven optimization problem where the compromise between the

accuracy of fitting data and fitting noise as posed in **Eq. 8** is the same as the probabilistic setup of problem in **Eq. 14**.

The objective now is to find a staircase function that minimizes the cost function in **Eq. 14**. A straightforward approach is to compare the cost of all possible staircase functions. Computationally this is not feasible. Consider a measured signal that has N samples with each sample allowed to take any of the possible M values. Evidently, the total number of candidate sequences is M^N. As N can be a large number (for example $\sim 10^4$ in this article), even if $M = 2$, the total number of possible comparisons is intractably large (10^{3010}). Fortunately, the structure of the cost function lends itself to a form where a solution approach, based on the dynamic programming, can be applied which greatly reduces the number of computations (less than 10^8). As shown later, computational complexity of this approach is NM^{m+1}, where m is the same as in **Eq. 7**. This is a significant reduction in the number of computations. Subsequently, other techniques are developed to further reduce the computational complexity that render the step detection method applicable to single molecule studies based on optical tweezers or AFM.

Dynamic programming approach to minimization

The minimization problem is made feasible by discretizing the space of step functions. That is, a step function is allowed to takes values from a finite but large set of numbers. The difference between two successive values is kept small and is referred to as the resolution parameter of the algorithm. Define, $s_k := [x_k, x_{k-1}, ..., x_{k-m+1}]^{'}$ that will be referred to as a state which keeps

track of the last m values of x_k. Assuming that x_k can have M possible values, then s_k can have $M_s := M^m$ possible values, $\{\alpha_1, ..., \alpha_{M_s}\}$. A staircase function, $\{x_1, x_2, \cdots, x_N\}$ can be uniquely represented by the corresponding sequence $\{s_1, s_2, \cdots, s_N\}$ such that s_{k+1} depends on s_k and another independent variable, $u_k := x_{k+1} - x_k$ through a function f, i.e., $s_{k+1} = f(s_k, u_k)$. u_k is a step that is said to transit s_k to s_{k+1} through a function f. Now define, $g_k := h_k(x_{k-m+1}^k, u_k) + W(u_k)$. Clearly, g_k is a function of (s_k, u_k) and represents the cost associated with the transition $s_k \rightarrow s_{k+1}$. Thus, the problem to be solved takes the form,

$$\hat{s}_1^N = \arg\min_{s_k} \sum_{k=0}^{N-1} g_k(s_k, u_k) \tag{15}$$

such that

$$s_{k+1} = f(s_k, u_k).$$

In other words, the problem in **Eq. 14** has been converted into the problem in **Eq. 15** where the optimal sequence $\{\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_N\}$ is to be found. A strategy to obtain the optimal sequence s_k is provided by the dynamic programming algorithm. The key step of this algorithm is described in **Supplementary Fig 2**. In the figure, horizontal axis represents the time/sample index k and the vertical axis represents the various values that the state s_k can take at any time index k. In the figure, the state s_k can take any of the values $\alpha_1, \ldots, \alpha_4$ that is $s_k \in \{\alpha_1, \ldots, \alpha_4\}$. The links in the graph of **Supplementary Fig 2** represent the cost of transitioning; for example, a particular link connection from an element of the column at time instant k to an element of the column at time instant $k+1$ represent the cost to transition from a particular value of the state at time k to a value of the state at $k+1$. For example, the cost to transition from a value α_j at time instant k to α_i at time instant $k+1$ is given by the link with weight $g_k(\alpha_j, u_k)$, where u_k is such that $f(\alpha_j, u_k) = \alpha_i$.

In the figure, the cost for transitioning from various values $\alpha_j,\ j = 1, \ldots, 4$ at time instant $k = 3$ to α_1 at time instant $k = 4$ is shown by the weights on the links. Also, at every instant k, for every possible value of the state s_k, the optimal cost to reach that state from the state at time zero is kept track of in $J_k(x_j)$. Thus, for time step $k = 3$, the optimal cost to reach α_2 is given by $J_3(\alpha_2)$. This cost is denoted in the figure at the $(3, 2)$ location of the graph. Also, the sequence of states that lead to the state s_j at time step k is determined, and is denoted by $S_1^k(\alpha_j)$. In the figure, for example, the optimal sequence that leads to the optimal cost to reach α_2 at time step 3, $J_3(\alpha_2)$ is given by the sequence $S_1^3(\alpha_2) = \{\alpha_1, \alpha_2, \alpha_2\}$. Furthermore $J_3(\alpha_2) = g_1(\alpha_1, u_1) + g_2(\alpha_2, u_2)$ where $\alpha_2 = f(\alpha_1, u_1)$ and $\alpha_2 = f(\alpha_2, u_2)$.

The iterative procedure to obtain the optimal cost and the optimal sequence to reach a particular state α_j at time instant $k + 1$ is obtained as follows. Assume that optimal cost $J_k(\alpha_j)$ and the optimal sequence $S_1^k(\alpha_j)$ to reach state α_j at time instant k is known for all $j = 1, \ldots, 4$. For time instant $k + 1$, consider the task of finding the optimal cost to reach α_i and the corresponding optimal sequence. This is achieved by comparing the optimal cost of reaching α_i at time instant $k+1$ from every possible state α_j at time instant k. This optimal cost of reaching α_i at instant $k+1$ *via* α_j at instant k is $J_k(\alpha_j) + g_k(\alpha_j, u_k)$ where u_k is such that $f(\alpha_j, u_k) = \alpha_i$. The optimal cost to reach α_i at time instant $k + 1$ is determined by the index $\hat{j}$ that minimizes $J_k(\alpha_j) + g_k(\alpha_j, u_k)$. The optimal cost $J_{k+1}(\alpha_i)$ to reach α_i at time instant $k + 1$ is then $J_k(\alpha_{\hat{j}}) + g_k(\alpha_{\hat{j}}, \hat{u}_k)$ where $g_k(\alpha_{\hat{j}}, \hat{u}_k) = \alpha_i$. The optimal sequence to reach α_i at instant $k + 1$ is $S_1^{k+1}(\alpha_i) = [S_1^k(\alpha_{\hat{j}}), \alpha_i]$.

This procedure can be carried to the last time instant and the state at the last time step with

the smallest cost denotes the optimal global cost achievable and the corresponding sequence to reach this state determines the optimal sequence for the problem. The proof of the last assertion is standard in the dynamic programming literature and can be found in [2].

Note that at any time step, arbitrary state transitions are not allowed. Only those state transitions that correspond to a valid staircase function are allowed. For example, if s_k is fixed to$\{\alpha_1, \alpha_2, \alpha_3\}$ then next possible states s_{k+1} will be of the form $\{\alpha_2, \alpha_3, \alpha_j\}\,;\ j = 1, ..., M$, assuming that x_k can take M possible values. Therefore for a given state, s_k, M transitions will be compared. As there are $M_s = M^m$ number of states s_k, the number of comparisons made at each time step is $MM_s = M^{m+1}$. Therefore, the number of comparisons made for N time steps is NM^{m+1} which is much less compared to the brute force method (M^N). However, if $m > 1$ then computational burden is still large especially if M is large. In order to limit computational burden when $m > 1$, the system can be modeled in an alternative fashion so that the dynamics as written in **Eq. 1** includes terms due to past outputs and just the immediate past input. Therefore, the numerator of **Eq. 2** will be of the form $b_0 + b_1 D$ so that $m = 1$. This is possible is most cases. In cases where this is not possible, an alternative formulation to cost function is suggested below that does not include higher order (> 1) input delay terms. Note that in the formulation for first order systems as in **Eq. 3**, which is the case for optical tweezers, no modification is required.

$$\tilde{z}(x_{k+1}) \triangleq b_0 x_{k+1} + \sum_{i=1}^{m} b_i S^k_{k+1-i}(x_k) - \sum_{j=1}^{l} a_i y_{k+1-j}$$

$$g(x_k, u_k) = \{y_{k+1} - \tilde{z}(x_{k+1})\}^2 + W(u_k)$$

$$\hat{u}_k = \arg\min_{u_k} [g(x_k, u_k) + J(x_k)]$$

where $\tilde{z}(x_{k+1})$ denotes a pseudo output of the filtering system when input is the candidate staircase function. $W(u_k)$ takes the same expression as in **Eq.** (8). Although this method is only sub-optimal, simulation results show good performance.

Optimizing the computation

Generally, dynamic programming algorithm requires large number of computations even if the states do not have memory. Consider a stair-case like stepping signal that ranges from 0 to L including noise. Then the possible staircase levels lie in the interval $[0, L]$. To apply dynamic programming, the set is partitioned as $\{0, \Delta, 2\Delta, ..., M\Delta\}$ such that $M\Delta \geq L$. As mentioned earlier, with M levels and N time points in the data, the total number of computations required scales as NM^2. Note that $M \propto L \propto N\Delta^{-1}$. Note that most of the computations are for transitions that are unlikely. Step sizes that take the states beyond 3σ of the observations may not be expected, therefore such state transitions can be removed from the computation. This is done by finding upper and lower bounds that envelope the observed data for each time point. Specifically, for any time index, the maximum and minimum values of the observed data in a window around the sample is computed and set as the bounds. Hence only the state transitions that involve states within the bounds of current and next time points are computed. This results in significant reduction in computational costs without affecting the quality of the results. In the implementation the window size is adapted based on the system model. If the system bandwidth is low compared to the sampling frequency then larger window size is required. Another technique to reduce the computation cost is to store optimal state transitions rather than the survivor vector. Updating survivor vector at each time step

leads to unnecessary read-write operations of large blocks of data that increases with time index. Storing state transitions instead involves bounded read-write operations of the order of M and the global survivor vector can be reconstructed at the end using stored optimal transitions.

Evidence for Single Molecule Force Experiments with Kinesin

Stall Force experiments were performed with the same kinesin sample, that data for which is presented in the main article. **Supplementary Fig. 3** shows representative traces for an optical trapped bead being pulled by kinesin. Bead movement stops when displaced by about 140 nm from the center of optical trap. Optical trap stiffness was set to about 0.035 pN/nm. Therefore, kinesin stalled at a load of about 5 pN. Multiple motor based bead movement in similar setups are shown to sustain forces in excess of 5 pN[3]. Another empirical indicator of single molecule experiment is the percentage of beads that show motility when brought in the vicinity of the microtubule. With a success rate of less than 30%, a negligible percentage of beads would have two or more motors attached to the same bead.

1. Rowe, D. *Multivariate Bayesian Statistics* (Chapman & Hall/CRC, 2002).

2. Sniedovich, M. *Dynamic programming* (CRC, 2009).

3. Vershinin, M., Carter, B., Razafsky, D., King, S. & Gross, S. Multiple-motor based transport and its regulation by Tau. *Proceedings of the National Academy of Sciences* **104**, 87 (2007).

USER INPUT: $y, \sigma_n^2, \sigma_v^2, b, a,$ resolution

$\alpha =$min(y):resolution:max(y)

$S_1^1(x_1) = x_1 \ \forall x_1 \in \alpha$

$\tilde{z}(x_1) = x_1$

$J_1(x_1) = \{y_1 - \tilde{z}(x_1)\}^2,$

$\bar{\sigma} = \sum_{j=0}^{m} b_j^2 \sigma_n^2 + \sum_{j=1}^{l} a_j^2 \sigma_\nu^2$

$p_0(u) = \exp\left(\frac{-4.5}{\bar{\sigma}} \delta(u_k \neq 0)\right)$

$\epsilon = \exp(-50)$

DO

$p(u) = p_0(u)$

FOR $k = 1 : N-1$

FOR x_{k+1} in α

$u_k = x_{k+1} - \alpha$

$W(u_k) = -2\bar{\sigma} \log p(u_k)$

$\tilde{z}(x_{k+1}) = b_0 x_{k+1} + \sum_{i=1}^{m} b_i S_{k+1-i}^k(\alpha) - \sum_{j=1}^{l} a_i y_{k+1-j}$

$g_x(\alpha, u_k) = \{y_{k+1} - \tilde{z}(x_{k+1})\}^2 + W(u_k)$

$\hat{u}_k = \arg\min_u [J(x_k) + g_k(\alpha, u_k)]$

$\tilde{x}_k = x_{k+1} - \hat{u}_k$

$J_{k+1}(x_{k+1}) = J_k(\tilde{x}_k) + g_k(\tilde{x}_k, \hat{u}_k)$

$S_1^{k+1}(x_{k+1}) = [S_1^k(\tilde{x}_k) \ x_{k+1}]$

END FOR

END FOR

$\hat{S} = \arg\min_{x_N} S(x_N)$

$p_0(u) =$ Normalized Histogram($\hat{S}$) + ϵ

WHILE $(p(u) \neq p_0(u))$

RETURN $\hat{S} = S(\tilde{x})$

Supplementary Table 1. Step detection Algorithm

Supplementary Figure 1 Discrete time input output model.

Supplementary Figure 2 Forward dynamic programming algorithm

Supplementary Figure 3 Stall force experiment. A kinesin coated optically trapped bead is pulled by the kinesin over a microtubule. The graph shows position of the bead (*green* axes) and force (*blue* axes), obtained by multiplying position values by optical trap stiffness constant (0.035 pN/nm). As kinesin pulls the bead away from the center of the trap, kinesin experiences increasing amount of force. At about 5 pN restoring force, the kinesin stalls, unable to pull the bead further away. Fast downward transitions are due the kinesin molecule getting detached from the microtubule. When kinesin detaches, the bead is pulled back to the center of the optical trap.