AIP | The Journal of Chemical Physics

Marcus canonical integral for non-Gaussian processes and its computation: Pathwise simulation and tau-leaping algorithm

Tiejun Li, Bin Min, and Zhiming Wang

Citation: J. Chem. Phys. **138**, 104118 (2013); doi: 10.1063/1.4794780
View online: http://dx.doi.org/10.1063/1.4794780
View Table of Contents: http://jcp.aip.org/resource/1/JCPSA6/v138/i10
Published by the American Institute of Physics.

Marcus canonical integral for non-Gaussian processes and its computation: Pathwise simulation and tau-leaping algorithm

Tiejun Li,[1,2,a)] Bin Min,[1,b)] and Zhiming Wang[1,c)]
[1]*Laboratory of Mathematics and Applied Mathematics and School of Mathematical Sciences, Peking University, Beijing 100871, People's Republic of China*
[2]*Beijing International Center for Mathematical Research, Beijing 100871, People's Republic of China*

(Received 13 December 2012; accepted 22 February 2013; published online 14 March 2013)

The stochastic integral ensuring the Newton-Leibnitz chain rule is essential in stochastic energetics. Marcus canonical integral has this property and can be understood as the Wong-Zakai type smoothing limit when the driving process is non-Gaussian. However, this important concept seems not well-known for physicists. In this paper, we discuss Marcus integral for non-Gaussian processes and its computation in the context of stochastic energetics. We give a comprehensive introduction to Marcus integral and compare three equivalent definitions in the literature. We introduce the exact pathwise simulation algorithm and give the error analysis. We show how to compute the thermodynamic quantities based on the pathwise simulation algorithm. We highlight the information hidden in the Marcus mapping, which plays the key role in determining thermodynamic quantities. We further propose the tau-leaping algorithm, which advance the process with deterministic time steps when tau-leaping condition is satisfied. The numerical experiments and its efficiency analysis show that it is very promising. © *2013 American Institute of Physics*. [http://dx.doi.org/10.1063/1.4794780]

I. INTRODUCTION

In various branches in natural and social sciences, the stochastic processes driven by non-Gaussian noise are very common and can be modeled with the stochastic differential equations (SDEs) driven by different types of non-Gaussian noise. This includes many examples such as the analysis of the shot noise in electrical circuits,[1] stochastic energetics in statistical mechanics,[2] the stock price modeling in option pricing,[3] and even the study of quantum groups in quantum mechanics.[4] In mathematics community, the theory of Lévy processes and its related SDEs have been extensively studied in the literatures and monographs.[5,6]

Recently, there are growing interests in understanding the small thermodynamic systems.[7] In the study of such systems, the thermodynamic quantities such as energy, work, and heat become stochastic due to the environmental fluctuations, and the thermodynamic energy balance can be established in a pathwise way. A theoretical framework called stochastic energetics is now in shape and has wide applications in theories and experiments (see Refs. 8–13 and the references therein).

For the SDEs driven by Gaussian white noise, the Stratonovich integral has been shown as an appropriate definition to be consistent with the stochastic energetics[8,9] since it satisfies the traditional Newton-Leibnitz chain rule. Indeed this approach has been adopted by many researchers.[10–13] The Stratonovich integral is somehow more preferred by physicists than Ito integral since it admits an interpretation as the limit of the solution of SDEs driven by Gaussian colored noise. This is the well-known Wong-Zakai's theorem.[14,15] For the SDEs driven by non-Gaussian noises which often appear in small thermodynamic systems,[9,16–20] the Stratonovich type integral in the sense of Wong-Zakai's smoothing limit has also been defined.[21] It is usually named as *Marcus canonical integral* for the pioneering work of Marcus in this topic[22,23] (for simplicity we will abbreviate it as *Marcus integral* in the rest of this paper). This type of integral is different from the straightforwardly generalized midpoint integral like the Stratonovich type integral for SDEs driven by Gaussian white noise, and of course is different from the leftmost endpoint integral as Ito's definition. It satisfies the Newton-Leibnitz chain rule, which is usually needed in stochastic energetics.[8] Some further properties of this integral may be referred to Ref. 5. Interestingly, it seems that this integral is not well-known in physics community and was recently rediscovered in Ref. 2 as * integral for understanding the stochastic energetics.

The purpose of this paper is to give a comprehensive review of Marcus integral and discuss its numerical methods, which will be necessary to further advance the study of stochastic energetics for non-Gaussian processes. We will discuss the pathwise simulation of the trajectory, its convergence estimate, the computation of thermodynamic quantities, and the forward Euler time stepping for acceleration, which we also call tau-leaping scheme due to similar ideas in simulating chemically reacting systems.[24,25] We will restrict us to SDEs driven by compound Poisson noise in the current paper, which is the common setup in practice.

The paper has two main parts. The first part is the theoretical part, which mainly states the equivalence between different definitions of the solutions. In Sec. II, we give a review of the development of Marcus integral. We will show how to derive it both physically and mathematically. In particular, we show the equivalence between the Di Paola-Falsone

[a)]Electronic mail: tieli@pku.edu.cn.
[b)]Electronic mail: minbmath@gmail.com.
[c)]Electronic mail: wangzmpku@gmail.com.

formulation and * integral in Sec. II A, Di Paola-Falsone formulation and Marcus integral in Sec. II B, and the * integral and Marcus integral in Sec. II C. The second part is the computational part, which introduces the pathwise simulation algorithm, its analysis and the tau-leaping algorithm. In Sec. III, we offer the pathwise simulation algorithm to the SDEs defined through Marcus integral, which is also proposed in Ref. 26. We give the convergence analysis to the algorithm and present some numerical examples for the verification of our analysis. In Sec. IV, we show how to compute thermodynamic quantities and numerically confirm the first law of thermodynamics in the framework of stochastic energetics. In Sec. V, we propose the forward Euler time stepping, i.e., tau-leaping scheme, to speed up the simulation without losing much accuracy. We give the condition when to use this algorithm and its efficiency analysis. In Sec. VI, some numerical examples are exhibited to show the efficiency and accuracy of the proposed tau-leaping algorithm. Finally we make conclusion in Sec. VII.

II. MARCUS INTEGRAL

In stochastic energetics with non-Gaussian noise perturbation, one usually faces the SDEs like

$$\frac{\mathrm{d}X(t)}{\mathrm{d}t} = f(X(t), t) + g(X(t), t)\xi(t), \tag{1}$$

where $\xi(t)$ is a Poisson white noise with realizations

$$\xi(t) = \frac{\mathrm{d}L(t)}{\mathrm{d}t} = \sum_{k=1}^{N(t)} R_k \delta(t - \tau_k).$$

Here τ_k is the jump time with rate λ, R_k is the jump size with distribution μ, and $N(t)$ is the number of jumps until time t. $L(t)$ is the underlying compound Poisson process and has the corresponding realization

$$L(t) = \sum_{k=1}^{N(t)} R_k H(t - \tau_k), \tag{2}$$

where $H(t)$ is the Heaviside function with unit jump at time zero. Equation (1) is also denoted as

$$\mathrm{d}X(t) = f(X(t), t)\mathrm{d}t + g(X(t), t)\mathrm{d}L(t) \tag{3}$$

in mathematics community. Both notations will be taken without distinction in this paper. We will only consider the scalar SDE case in this section for simplicity of clarification. But all of the results can be generalized to the vectorial case with straightforward modifications.

The issue comes from the definition of stochastic integral $\int_0^t g(X(s), s)\xi(s)\mathrm{d}s$ or equivalently $\int_0^t g(X(s), s)\mathrm{d}L(s)$ in explaining SDE (1). If $\xi(t) = \dot{W}(t)$ is the temporal Gaussian white noise with $\mathbb{E}\xi(t) = 0$ and $\mathbb{E}\xi(t)\xi(s) = \delta(t - s)$ where $\mathbb{E}$ means the probabilistic expectation and $W(t)$ is the standard Brownian motion, we have the famous Ito integral and Stratonovich integral defined as

(i) Ito integral

$$\int_0^t g(X(s), s) \cdot \mathrm{d}W(s) = \lim_{\Delta t \to 0} \sum_{i=1}^{n} g(X(t_i), t_i)\Delta W_i, \tag{4}$$

(ii) Stratonovich integral

$$\int_0^t g(X(s), s) \circ \mathrm{d}W(s) = \lim_{\Delta t \to 0} \sum_{i=1}^{n} \frac{1}{2}\left[g(X(t_i), t_i) + g(X(t_{i+1}), t_{i+1})\right]\Delta W_i, \tag{5}$$

where $\{t_i\}$ is a subdivision of time interval $[0, t]$ and $\Delta W_i = W(t_{i+1}) - W(t_i)$. The notation $\cdot$ and $\circ$ denote the Ito's definition and Stratonovich's definition, respectively. It is well-known that the Stratonovich integral satisfies the Newton-Leibnitz's chain rule, while the Ito integral satisfies the famous Ito's rule.[27] In quantitative studies in finance, Ito calculus is preferred due to the property that the expected income given the current state is kept invariant under the fluctuations, which is the result of choosing the leftmost endpoint in the definition (4). However, Stratonovich calculus is often prefereed in physics because it admits an interpretation as the limit of the solution of SDEs with Gaussian colored noise, which is suggested by Wong-Zakai's theorem.[14] Since most physical noise are colored noise, the adoption of Stratonovich integral will be more consistent with reality.

In case of Poisson white noise, since the trajectories of X and L are defined to be right-continuous with left-hand limit mathematically due to discontinuity, one should define the Ito integral as in (4)

$$\int_0^t g(X(s-), s)\mathrm{d}L(s) = \lim_{\Delta t \to 0} \sum_{i=1}^{n} g(X(t_i-), t_i)\Delta L_i, \tag{6}$$

where $X(s-)$ is the left-hand limit at s, $\Delta L_i = L(t_{i+1}) - L(t_i)$ (we will omit the symbol $\cdot$ in Ito integrals for ease of notation in later texts). The Ito's rule for $h(X(t))$ with this definition is proved to be[5]

$$\begin{aligned} h(X(t)) - h(X(0)) &= \int_0^t h'(X(s-))\mathrm{d}X(s) \\ &\quad + \sum_{0 \le s \le t} [h(X(s)) - h(X(s-)) - \Delta X(s)h'(X(s-))], \end{aligned} \tag{7}$$

where $\Delta X(s) = X(s) - X(s-)$ is the jump of X at s. The first term can be further reduced to

$$\int_0^t h'(X(s-))\mathrm{d}X(s) = \int_0^t h'(X(s))f(X(s), s)\mathrm{d}s + \int_0^t h'(X(s-))g(X(s-), s)\mathrm{d}L(s).$$

The second term on the right-hand side (rhs) of (7) shows the difference between the Ito's rule and the Newton-Leibnitz chain rule. The necessity to include this additional term can

be heuristically validated from the following simple example. Suppose $f=0$, $g=1$ in (1). We have $X(t)=L(t)$ indeed. Consider a specific realization $L(t)=H(t-1/2)$ and let us verify the Ito's rule (7) when taking $t=1$. The left hand side gives $h(1)-h(0)$. The first term on the rhs gives

$$\int_0^t h'(X(s-))\mathrm{d}X(s)=h'(0)$$

by the definition of Ito integral (6) since the only contribution term in the summation arises in the interval $[t_i, t_{i+1})$ containing 1/2. The second term on the rhs gives $h(1)-h(0)-h'(0)$. It is this additional term that compensates the difference between $\int_0^t \mathrm{d}h(X(s))$ and $\int_0^t h'(X(s-))\mathrm{d}X(s)$.

Turning to the Stratonovich integral for Poisson noise, we will find that the straightforward extension of (5) does not satisfy Newton-Leibnitz chain rule. This can be heuristically checked from the same example considered above. We have

$$\int_0^t h'(X(s))\circ \mathrm{d}X(s)=\frac{1}{2}[h'(0)+h'(1)]$$

from the definition (5). The other choices like the midpoint $h(X(t_{i+\frac{1}{2}}))$ or $h((X(t_i)+X(t_{i+1}))/2)$ in the summand do not give meaningful solution, either.

A satisfying resolution on the definition of Stratonovich integral to achieve chain rule can be realized through Wong-Zakai type smoothing limit technique. It has been pursued by several authors.[21–23,28,29] Due to Marcus' pioneering work on this topic, it is usually named as *Marcus integral* in the literature,[5] which will be denoted as

$$\int_0^t g(X(s),s)\diamond \mathrm{d}L(s)$$

in this paper. Below we state three main approaches proposed in the literature, which are formally equivalent but totally different in numerical performance. For simplicity, in Secs. II A, II B, and II C, we will take $f=0$, $g(X(t),t)=g(X(t))$ and a specific single-jump realization of the Poisson noise $L(t)=R_0H(t-t_0)$ in Eq. (3). That is, we consider the following equation

$$\mathrm{d}X(t)=g(X(t))\diamond \mathrm{d}L(t), \tag{8}$$

in Secs. II A, II B, and II C. The Marcus integral for more general SDE will be presented in Sec. II D.

A. Taylor series formulation by Di Paola and Falsone

The Taylor series formulation for Marcus integral is proposed by Di Paola and Falsone.[28,29] Let us briefly state the idea of this formulation in this subsection. For any analytical function $\phi(x)$ near X, we can represent the increment of ϕ through Taylor expansion as

$$\Delta\phi=\sum_{j=1}^{\infty}\frac{\mathrm{d}^j\phi}{j!}, \tag{9}$$

where $\mathrm{d}^j\phi$ is defined recursively as

$$\mathrm{d}^j\phi=\frac{\partial(\mathrm{d}^{j-1}\phi)}{\partial x}\mathrm{d}X,\ \mathrm{d}^1\phi=\frac{\partial\phi}{\partial x}\mathrm{d}X. \tag{10}$$

More concretely,

$$\mathrm{d}^2\phi=\frac{\partial(\mathrm{d}^1\phi)}{\partial x}\mathrm{d}X=\frac{\partial^2\phi}{\partial x^2}(\mathrm{d}X)^2+\frac{\partial\phi}{\partial x}\mathrm{d}^2X,$$

$$\mathrm{d}^3\phi=\frac{\partial(\mathrm{d}^2\phi)}{\partial x}\mathrm{d}X=\frac{\partial^3\phi}{\partial x^3}(\mathrm{d}X)^3+3\frac{\partial^2\phi}{\partial x^2}\mathrm{d}X\mathrm{d}^2X+\frac{\partial\phi}{\partial x}\mathrm{d}^3X.$$

Note that $\mathrm{d}X$ is the increment of the argument of ϕ. It may be the ultimate source of the increment, in which case $\mathrm{d}^jX=0$ for $j>1$ and we obtain the usual Taylor expansion. It is also possible that x may further depend on other variables, say t, then $\mathrm{d}^1X=X'(t)\mathrm{d}t$, $\mathrm{d}^2X=X''(t)(\mathrm{d}t)^2+X'(t)\mathrm{d}^2t$, etc. In both cases the formula (9) gives a unified representation.

For SDE (8), the *Di Paola-Falsone's formulation* is defined as

$$\mathrm{d}X(t)=\sum_{j=1}^{\infty}\frac{g_j(X(t-))}{j!}(\mathrm{d}L(t))^j. \tag{11}$$

The underlying idea is to apply (9) with $\phi(x)=x$. Then,

$$\Delta X(t_0)=X(t_0)-X(t_0-)=\sum_{j=1}^{\infty}\frac{\mathrm{d}^jX}{j!}, \tag{12}$$

where $\mathrm{d}^jX=g_j(X(t_0-))R_0^j$ and

$$g_1(x)=g(x),\quad g_j(x)=\frac{\partial g_{j-1}(x)}{\partial x}g_1(x) \tag{13}$$

by definitions (8) and (10). The final result reads

$$\Delta X(t_0)=\sum_{j=1}^{\infty}\frac{g_j(X(t_0-))}{j!}(\mathrm{d}L(t_0))^j, \tag{14}$$

which exactly is Eq. (11).

The validity of (14) also can be understood from the smoothing argument. Consider a smoothed version of $L(t)$ as $L_\delta(t)$ (Fig. 1) and we let $\delta\to 0$. Define $\mathrm{d}X_\delta(t)=g(X_\delta(t))\mathrm{d}L_\delta(t)$. We have

$$\Delta X(t_0)\approx X_\delta\left(t_0+\frac{\delta}{2}\right)-X_\delta\left(t_0-\frac{\delta}{2}\right)=\sum_{j=1}^{\infty}\frac{\mathrm{d}^jX_\delta}{j!}$$

$$=\sum_{j=1}^{\infty}\frac{g_j(X_\delta(t_0-\delta/2))}{j!}(\mathrm{d}L_\delta(t_0))^j$$

$$\to\sum_{j=1}^{\infty}\frac{g_j(X(t_0-))}{j!}(\mathrm{d}L(t_0))^j$$

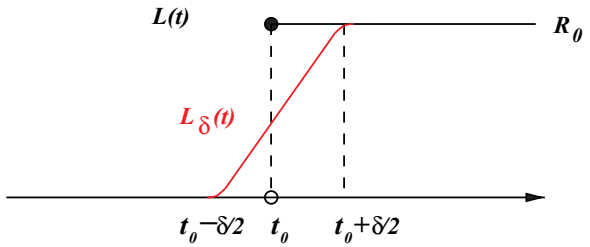

FIG. 1. The smoothed function $L_\delta(t)$ of the single-jump realization $L(t)$.

as δ goes to 0. Note that in this limit $\mathrm{d}t = \delta \to 0$ and $L'_\delta(t) \to \infty$, but the product $\mathrm{d}L_\delta(t) = L'_\delta(t)\mathrm{d}t$ remains finite and tends to the jump size R_0.

We remark that the Di Paola-Falsone's Taylor series formulation is equivalent to the * integral formula recently proposed in Ref. 2. For * integral, it is given that for any function $h(x)$, $\mathrm{d}Y(t) = h(X(t)) \diamond \mathrm{d}L(t)$ and $X(t)$ satisfies (8), then the jump of $Y(t)$

$$\Delta Y = \sum_{j=1}^{\infty} \frac{(\mathrm{d}L(t_0))^j}{j!} \left(\left(g(x) \frac{\partial}{\partial x} \right)^{j-1} h(x) \right) \Bigg|_{x=X(t_0-)}. \quad (15)$$

In case that $h(x) = g(x)$, the rhs of Eq. (15) equals the rhs of Eq. (14) and thus $\Delta Y = \Delta X$, if we notice that $g_j(x) = (g(x)\partial/\partial x)^{j-1} g(x)$ from the recursive relation (13). This verifies the equivalence of Di Paola-Falsone's formulation and * integral proposed in Ref. 2.

Interestingly, we can observe that the first order truncation of (14)

$$\Delta X = g(X(t_0-))\mathrm{d}L(t_0)$$

exactly corresponds to the Ito's definition of stochastic integral, i.e., the choice of the left-most endpoint. While the incorporation of the second order term does not correspond to any straightforward definition of stochastic integrals, neither equally weighted average for the endpoint function values like $(g(X(t_0-))+g(X(t_0)))dL(t_0)/2$ nor other types of combination like

$$(\theta g(X(t_0-)) + (1-\theta) g(X(t_0)))\, \mathrm{d}L(t_0),$$

where $\theta \in [0, 1]$.

B. Ordinary differential equation (ODE) formulation through Marcus mapping

One amazing thing about the series formulation (14) or (15) is that it admits an equivalent but more elegant ODE formulation. Some pioneering work has been done by Marcus[22,23] and the extension to general semi-martingales is discussed in Ref. 21.

To see this we first apply the chain rule

$$\mathrm{d}h(X(t)) = h'(X(t))g(X(t)) \diamond \mathrm{d}L(t).$$

The series (15) has the following formal representation at jump time t_0

$$\Delta h = \sum_{j=1}^{\infty} \frac{(\mathrm{d}L(t_0))^j}{j!} \left(\left(g(x) \frac{\partial}{\partial x} \right)^{j} h(x) \right) \Bigg|_{x=X(t_0-)}$$
$$= (e^{\mathrm{d}L(t_0)\cdot g(x)\frac{\partial}{\partial x}} - 1)h(x) \Big|_{x=X(t_0-)}.$$

Note that the exponential operator $e^{\mathrm{d}L(t_0)\cdot g(x)\frac{\partial}{\partial x}} h(x)$ at $x = X(t_0-)$ is exactly the solution mapping of the following ODEs

$$\frac{\mathrm{d}x}{\mathrm{d}s} = g(x)\mathrm{d}L(t_0), \quad s \in [0, 1], \quad (16)$$

$$\frac{\mathrm{d}y}{\mathrm{d}s} = g(x)h'(x)\mathrm{d}L(t_0), \quad s \in [0, 1], \quad (17)$$
$$x(0) = X(t_0-),$$
$$y(0) = h(X(t_0-)),$$

and $X(t_0) = x(1)$, $h(X(t_0)) = y(1)$. The equation for $x(s)$ is called *Marcus mapping* in the literature. Define the flow map associated with x as

$$\Phi_g(\,\cdot\,; dL(t_0)) : x(0) \mapsto x(1) \in \mathbb{R}.$$

The *Marcus integral* for SDE (8) with the realization (2) is defined as

$$X(t) = X(0) + \sum_{k=1}^{N(t)} [\Phi_g(X(\tau_k-), R_k) - X(\tau_k-)]. \quad (18)$$

It has been shown in Ref. 21 that this definition can be viewed as the Wong-Zakai type smoothing limit of the considered SDE. It also satisfies the Newton-Leibnitz chain rule, which can be verified directly or obtained from the equivalence to the Di Paola-Falsone formulation. We should remark that though these two formulations are formally equivalent, the ODE formulation requires less smoothness assumption on $g(x)$ and $h(x)$. More importantly, they are totally different in the numerical performance. Usually the truncation cause severe error and the resulting scheme is not stable enough. On the other hand, we will observe that the path $x(s)$ generated by Marcus mapping is also an indispensable component of the solution though the trajectory $X(t)$ itself does not see this information! This point will be further clarified in Sec. IV.

C. Equivalence between the ∗ integral and Marcus integral

In this subsection we give a direct derivation of the Marcus mapping for the Marcus integral through smoothing approach. As we have stated in the Introduction, the limiting solution obtained by smoothing approach is often more physically relevant. The smoothing idea is indeed taken by Marcus[22] and other researchers,[2,21] and it is also implicitly used in Ref. 28 as we show in Sec. II A. This direct derivation also shows the equivalence between the $*$ integral proposed in Ref. 2 and Marcus integral.

Now consider a special realization of the Poisson noise with a single jump with height R_0 at $t = 0$, i.e., $L(t) = R_0 H(t)$. The corresponding Poisson white noise $\xi(t) = \mathrm{d}L(t)/\mathrm{d}t = R_0\delta(t)$, where $\delta(\,\cdot\,)$ is the Dirac's δ-function. We take the same smoothing

$$\xi_\varepsilon(t) = \frac{1}{\varepsilon} \int_{t-\varepsilon}^{t} \mathrm{d}L(s), \quad L_\varepsilon(t) = \int_{-\infty}^{t} \xi_\varepsilon(s)\mathrm{d}s \quad (19)$$

as adopted in Refs. 2 and 21. The idea to define the solution of Eq. (8) through smoothing is to consider

$$\mathrm{d}X_\varepsilon(t) = g(X_\varepsilon(t), t)\mathrm{d}L_\varepsilon(t) \tag{20}$$

and take the limit $X_\varepsilon(t) \to X(t)$ as the smoothing parameter $\varepsilon \to 0$. We equip Eq. (20) with an initial condition, say $X_\varepsilon(-1) = x_0$.

For Eq. (20), it is straightforward to obtain the limit $X(t) = x_0$ for any $t \le 0$ since $L_\varepsilon(t) = 0$ for any $t \le 0$. For any $t > 0$, we have the limit $X(t) = \lim_{\varepsilon \to 0} X_\varepsilon(t) = X_\varepsilon(\varepsilon)$ when ε is small enough, where

$$\frac{\mathrm{d}X_\varepsilon(t)}{\mathrm{d}t} = \frac{R_0}{\varepsilon} g(X_\varepsilon(t)), \quad X_\varepsilon(0) = x_0, \quad t \in [0, \varepsilon].$$

Define $x(t) = X_\varepsilon(\varepsilon t)$, then $X(t) = x(1)$ and $x(t)$ satisfies

$$\frac{\mathrm{d}x}{\mathrm{d}t} = g(x) \cdot R_0, \quad x(0) = x_0, \quad t \in [0, 1]. \tag{21}$$

This is exactly the Marcus mapping introduced in Subsection II B and it is easy to observe that all of the above derivations hold in the vectorial case.

Furthermore, let us consider $Y(t)$ for equation

$$\mathrm{d}Y(t) = h(X(t))\mathrm{d}L(t), \ Y(-1) = y_0 \tag{22}$$

defined through the following smoothing procedure

$$\mathrm{d}Y_\varepsilon(t) = h(X_\varepsilon(t))\mathrm{d}L_\varepsilon(t), \ Y_\varepsilon(-1) = y_0,$$

where $X(t)$ and $X_\varepsilon(t)$ are the limit solution and the smoothed solution of (20), respectively. We will have $Y(t) = y_0$ for any $t \le 0$. For any $t > 0$, we have

$$Y(t) = \lim_{\varepsilon \to 0} Y_\varepsilon(t) = y_0 + \frac{1}{\varepsilon} \int_0^\varepsilon h(X_\varepsilon(s))R_0 \mathrm{d}s,$$

when ε is small enough. With the same definition for $x(t)$ as in (21), we have

$$Y(t) = y_0 + \int_0^1 h(x(\tau))R_0 \mathrm{d}\tau.$$

This is equivalent to say $Y(t) = y(1)$, where $y(t)$ satisfies

$$\frac{\mathrm{d}y}{\mathrm{d}t} = h(x)R_0, \quad y(0) = y_0, \quad t \in [0, 1]. \tag{23}$$

In fact, this result has been contained in interpreting (20) by using (21) since it also holds for multidimensional case. With the definition

$$\boldsymbol{X}(t) = (X(t), Y(t))^T, \ \boldsymbol{g}(\boldsymbol{X}(t)) = (g(X(t)), h(X(t)))^T,$$

we have

$$\mathrm{d}\boldsymbol{X}(t) = \boldsymbol{g}(\boldsymbol{X}(t))\mathrm{d}L(t) \tag{24}$$

by combining the equations for X and Y together, where the solution is interpreted as the smoothing limit. Thus the application of the multidimensional version of (21) gives (23) directly.

For $Z(t) = h(X(t))$, we will show it satisfies

$$\mathrm{d}Z(t) = h'(X(t))\mathrm{d}X(t) = h'(X(t))g(X(t))\mathrm{d}L(t)$$

in the sense of smoothing limit. From the result for the SDE (22), we have the smoothing limit $Z(t) = z(1)$ where $z(t)$ satisfies

$$\frac{\mathrm{d}z}{\mathrm{d}t} = h'(x)g(x)R_0, \quad z(0) = z_0, \quad t \in [0, 1], \tag{25}$$

and $z(0) = h(x_0)$. It has the unique solution $z(t) = h(x(t))$. So the smoothing limit $Z(t) = h(x_0)$ if $t \le 0$ and $Z(t) = h(x(1))$ if $t > 0$. This coincides with the definition of $Z(t) = h(X(t))$. This verifies the definition by smoothing limit satisfies the chain rule.

We should remark that we have skipped the right continuity condition of the solution at the discontinuity point with the above limiting process. Another important fact about the above derivations is that the smoothing (20) and the limit of (20) exactly embodies the idea of the construction of $*$ integral discussed in Ref. 2. For $X(t)$ satisfying SDE (8), the $*$ *integral* is defined as

$$\int_0^t g(X(t)) * \xi(t)\mathrm{d}t := \lim_{\varepsilon \to 0} \lim_{\delta t \to 0} \sum_i \delta t \xi_\varepsilon(t_i) g(X(t_i)), \tag{26}$$

in which $\{t_i\}$ is a subdivision of the time interval and δt is the corresponding stepsize. The first limit as $\delta t \to 0$ corresponds to take the continuous integral and the second limit as $\varepsilon \to 0$ corresponds to take the smoothing limit. These procedures are implicitly taken in the above derivations. This again explains the equivalence between the $*$ integral and Marcus integral. We note that since Marcus integral and $*$ integral can be obtained by the same smoothing approach, these two integrals should be equivalent in the sense of "almost surely." A slight difference between these two may be that the Marcus integral needs to introduce a new variable to form a simultaneous system while the $*$ integral does not need such an operation each time when one wants to evaluate a new stochastic integral.

D. General formulations

Consider the following multidimensional SDE

$$\mathrm{d}\boldsymbol{X}(t) = \boldsymbol{f}(\boldsymbol{X}(t), t)\mathrm{d}t + \boldsymbol{g}(\boldsymbol{X}(t), t) \diamond \mathrm{d}\boldsymbol{L}(t), \tag{27}$$

where

$$\boldsymbol{L}(t) = \sum_{k=1}^{N(t)} \boldsymbol{R}_k H(t - \tau_k),$$

$\boldsymbol{X} \in \mathbb{R}^d$, $\boldsymbol{f} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, $\boldsymbol{g} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^{d \times u}$, and $\boldsymbol{R} \in \mathbb{R}^u$. Then, the Marcus integral for this SDE through Marcus mapping is

$$\begin{aligned}\boldsymbol{X}(t) = \ & \boldsymbol{X}(0) + \int_0^t \boldsymbol{f}(\boldsymbol{X}(s), s)\mathrm{d}s \\ & + \sum_{k=1}^{N(t)} [\Phi_{\boldsymbol{g}}(\boldsymbol{X}(\tau_k-), \boldsymbol{R}_k) - \boldsymbol{X}(\tau_k-)].\end{aligned} \tag{28}$$

The flow map $\Phi_{\boldsymbol{g}}$ at $t = \tau_k$ is defined through the ODEs

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}s} &= \boldsymbol{g}(\boldsymbol{x}, \tau_k)\boldsymbol{R}_k, \quad s \in [0, 1], \\ \boldsymbol{x}(0) &= \boldsymbol{X}(\tau_k-), \end{aligned} \tag{29}$$

and $\Phi_{\boldsymbol{g}}(\boldsymbol{X}(\tau_k-), \boldsymbol{R}_k) = \boldsymbol{x}(1)$.

For the case that

$$\boldsymbol{L}(t) = \sum_{k=1}^{N(t)} \boldsymbol{R}_k H(t - \tau_k) + b\boldsymbol{W}(t), \tag{30}$$

where $\boldsymbol{W}(t)$ is a u-dimensional standard Brownian motion, the Marcus integral for SDE (27) through Marcus mapping is defined as

$$\begin{aligned} \boldsymbol{X}(t) = \boldsymbol{X}(0) &+ \int_0^t \boldsymbol{f}(\boldsymbol{X}(s), s)\mathrm{d}s \\ &+ b\int_0^t \boldsymbol{g}(\boldsymbol{X}(s), s) \circ \mathrm{d}\boldsymbol{W}(s) \\ &+ \sum_{k=1}^{N(t)} [\Phi_{\boldsymbol{g}}(\boldsymbol{X}(\tau_k-), \boldsymbol{R}_k) - \boldsymbol{X}(\tau_k-)]. \end{aligned} \tag{31}$$

For the case that $\boldsymbol{L}$ is a more general Lévy process, we refer the readers to Ref. 5.

Finally we should comment that the Marcus integral is only defined for the integrands which are only arbitrary functions of the solution of the SDEs (27). It can be easily extended to the integrand which has explicit dependence on t.[5] But for an arbitrary function $h(t)$ which is not known to depend on $\boldsymbol{X}$ explicitly, there will be difficulty to define the Marcus integral.

III. PATHWISE SIMULATION ALGORITHM

With the mathematical description of the Marcus integral in Sec. II, it is ready to propose the pathwise simulation algorithm based on the Marcus mapping. We note that it is also proposed in Ref. 26.

A. Algorithm and its convergence analysis

For simplicity, we refer $\mathcal{E}xp(\lambda)$ to exponentially distributed random variable with mean $1/\lambda$. We propose the pathwise simulation algorithm for the following SDE:

$$\mathrm{d}X(t) = f(X(t))\mathrm{d}t + g(X(t)) \diamond \mathrm{d}L(t). \tag{32}$$

When $L(t) = \sum_{k=1}^{N(t)} R_k H(t - \tau_k)$, we can get the following algorithm.

Algorithm 1 (Pathwise Simulation Algorithm)

1. *Given $t = 0$, initial state $X(0)$, and the end time T.*
2. *Generate a waiting time $\tau \sim \mathcal{E}xp(\lambda)$ and a jump size $R \sim \mu$ where μ is the distribution of random jumps.*
3. *Solve the following ODE with initial value $X(t)$ until time $s = \tau$ to get its solution $X(u)$ $(u \in [t, t+\tau))$.*

$$\frac{\mathrm{d}y}{\mathrm{d}s} = f(y), y(0) = X(t). \tag{33}$$

4. *Solve the following ODE with initial value $X((t+\tau)-)$ until time $s = 1$ to get $X(t+\tau)$.*

$$\frac{\mathrm{d}x}{\mathrm{d}s} = g(x) \cdot R, x(0) = X((t + \tau)-). \tag{34}$$

5. *Set $t = t+\tau$, go to Step 2 unless $t \geq T$.*

We note that this algorithm can be easily generalized into the vectorial case. We remark that this pathwise simulation algorithm can be also easily extended into SDEs which are driven by Gaussian and non-Gaussian noises simultaneously. For instance, when

$$L(t) = \sum_{k=1}^{N(t)} R_k H(t - \tau_k) + bW(t), \tag{35}$$

where W_t is a standard Brownian motion, with formulation (31), we can get a pathwise simulation algorithm below.

Algorithm 1′ (Pathwise Simulation Algorithm)

1. *Given $t = 0$, initial state $X(0)$ and the end time T.*
2. *Generate a waiting time $\tau \sim \mathcal{E}xp(\lambda)$ and a jump size $R \sim \mu$ where μ is the distribution of random jumps.*
3. *Solve the following SDE with initial value $X(t)$ until time $s = \tau$ to get its solution $X(u)$ $(u \in [t, t+\tau))$.*

$$\mathrm{d}y = f(y)\mathrm{d}s + g(y) \circ \mathrm{d}W_s, y(0) = X(t). \tag{36}$$

4. *Solve the following ODE with initial value $X((t+\tau)-)$ until time $s = 1$ to get $X(t+\tau)$.*

$$\frac{\mathrm{d}x}{\mathrm{d}s} = g(x) \cdot R, x(0) = X((t + \tau)-). \tag{37}$$

5. *Set $t = t+\tau$, go to Step 2 unless $t \geq T$.*

We define the ODEs (33) and (34) as *drift* and *jump ODE*, respectively, in the later texts for simplicity. The numerical integration of them can be done with any existing ODE solver. Since the successive approximation will induce the accumulation of the errors, we should make an analysis to ensure the convergence of the overall scheme. Below we state the strong convergence result of this algorithm.

Theorem 1 (Convergence of pathwise simulations). *For the SDE (32), assume that f and g are Lipschitz functions with Lipschitz constants L_f and L_g respectively. Denote λ the jump intensity and R the random jumpsize. Assume the exponential moment $K = \mathbb{E}\exp(L_g|R|)$ exists. If we apply a*

pth order ODE solver to (33) with stepsize h_1 and a qth order ODE solver to (34) with stepsize h_2, we will have

$$\mathbb{E}|X^n(T) - X(T)| \le C_1\mathbb{E}|X^n(0) - X(0)| + C_2h_1^p + C_3h_2^q,$$

where $X^n(t)$ is the numerical solution and $X(t)$ is the exact solution. Here

$$C_1 = \exp(L_fT + \lambda T(K-1)),$$
$$C_2 = \hat{C}_2(\exp(L_fT) - 1)(\exp(\lambda T(K-1)) + 1),$$
$$C_3 = \hat{C}_3\exp(L_fT)(\exp(\lambda T(K-1)) - 1),$$

where $\hat{C}_2$ and $\hat{C}_3$ are independent of h_1, h_2, T, λ, K, L_f, L_g, p, and q.

The proof is shown in the Appendix.

It is interesting to observe that the strong convergence result for Algorithm 1 is quite different from the case when the SDE is driven by Gaussian white noise.[30] Usually the strong convergence order of the Euler-Maruyama scheme is 1/2. However, in the considered case, it is very easy to achieve high order accuracy, which is due to the simplicity of the compound Poisson process. We can observe from the theorem that the initial error will be amplified depending on the driving process and the final time T. High order numerical methods for (33) and (34) is useful in general. In Algorithm 1′, however, the strong convergence order would be limited by step 3 where the SDE is driven by Gaussian white noise. We also note that the strong convergence for Algorithm 1′ could be easily obtained by mimicking the proof of Theorem 1.

B. Numerical results

Now we apply the pathwise simulation algorithm to the so-called quasi-linear model

$$\mathrm{d}X(t) = -X(t)\mathrm{d}t + X(t)\diamond\mathrm{d}L(t), \quad X(0) = 1. \tag{38}$$

in Di Paola's paper.[31] It is a stochastic process driven by the double well potential $U(x) = x^2/2$ and the fluctuations depending linearly on the state and impulses. In our example, we choose $L(t)$ to be a compound Poisson process with jump size $N(0, \sigma^2)$ and intensity λ.

At first let us check the difference between the Ito integral and Marcus integral. It is easy to find that $X(t) = X(0)e^{-t+L(t)}$. The simulation for a specific realization of the solution compared with the Ito integral is shown in Fig. 2. We can observe that the pathwise simulation coincides with the exact solution perfectly well and the clear difference between different definitions of stochastic integrals.

Next we check the convergence order of the scheme and compare it with our theoretical estimate in Theorem 1. We use pth and qth order Runge-Kutta methods for (33) and (34) respectively and choose different stepsizes to extract convergence order. The stepsize for (33) and (34) are chosen to be the same. We find the numerical convergence order is quite close to the theoretical value min $\{p, q\}$, which confirms our estimate (shown in Table I). To further examine the dependence on initial error, we artificially add a small perturbation to the initial value. This is done by choosing $X^n(0) = X(0) + h$, where h is the stepsize. Thus the convergence order should be 1 based on our theorem if p and q are both bigger than 1. The convergence order is computed as shown in the left panel of Fig. 3, which gives the numerical order 1.0157. The right panel gives the numerical slope of error against stepsize curve as 4.1905, while the theoretical estimate is $C_1 = 4.0716$ by our theorem. All these results show that our theorem gives a good estimate to the convergence order. However, it is hard to provide error analysis for either the smoothing method[2] or the method based on the Taylor series (14). In this sense, the pathwise simulation algorithm is more tractable.

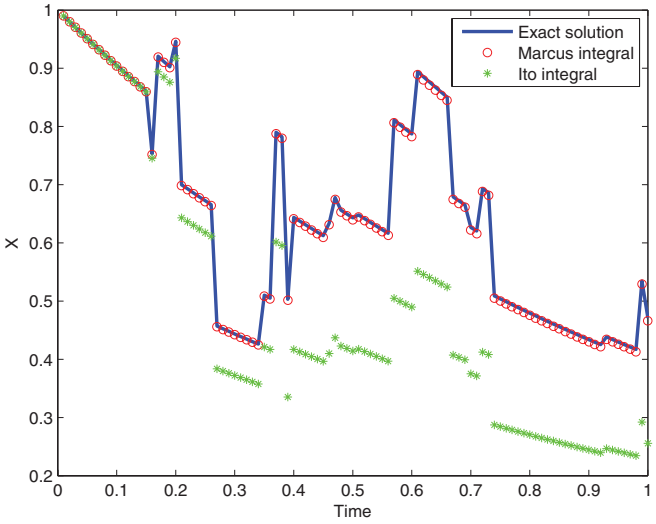

FIG. 2. Shown in the figure is one specific realization of the solution. The blue solid line is the theoretical solution. The star symbol shows the numerical result by pathwise simulation algorithm and the red circle corresponds to the Ito integral. The second order Runge-Kutta methods are used for both drift and jump ODEs and the time stepsize $\delta t = 0.01$. Some other parameters are $\lambda = 20$ and $\sigma = 0.2$.

IV. COMPUTATION OF THERMODYNAMIC QUANTITIES

A. Computational strategy

Physical quantities like heat, potential, and energy are important concepts in realistic problems. In classical mechanics and thermodynamics, these quantities are deterministic variables. In the context of stochastic energetics,[8,32,33] these quantities become stochastic, depending on the individual realization of the stochastic system. In Ref. 2, the stochastic energetics for non-Gaussian processes was established based

TABLE I. For ODEs (33) and (34), we use pth and qth order Runge-Kutta methods, respectively. The time stepsize is chosen to be $\Delta t = 0.01$, 0.008, 0.005, 0.004, 0.002, and 0.001. Some parameters are $\lambda = 20$, $\sigma = 0.2$, and $T = 1$. Two thousand samples are simulated. The numbers shown in the table are the slope by linear fitting compared with the theoretical value in parenthesis.

	$q=2$	$q=3$	$q=4$
$p=2$	1.9909 (2)	1.9851 (2)	1.9827 (2)
$p=3$	1.9889 (2)	2.9915 (3)	2.9937 (3)
$p=4$	1.9427 (2)	3.0012 (3)	3.9937 (4)

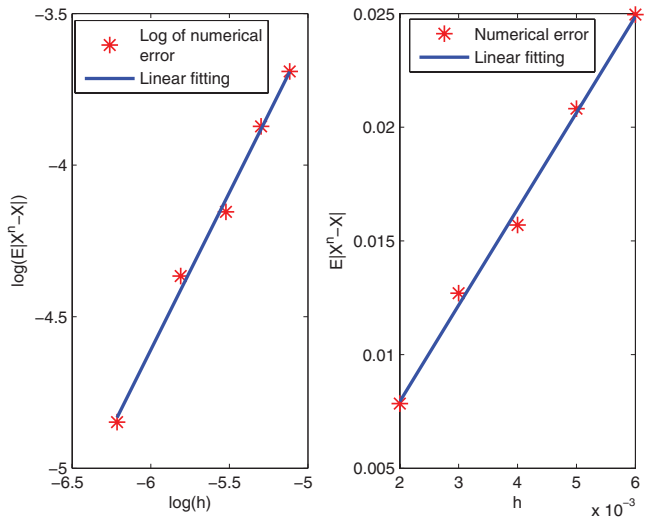

FIG. 3. For ODEs (33) and (34), we use the Runge-Kutta methods with $p = 3$ and $q = 3$. The time stepsize is chosen to be $\Delta t = 0.006, 0.005, 0.004, 0.003$, and 0.002. Some parameters are $\lambda = 20$, $\sigma = 0.2$, $T = 1$. Three thousand samples are simulated. Shown in the left panel is the log-log plot of the error of mean versus time stepsizes, which gives numerical order 1.0157. The right panel shows the linear fitting of error against stepsize curve, which gives slope 4.1905.

on $*$ integral, and a computational strategy was proposed to compute thermodynamic quantities based on the smoothing techniques. Realizing the equivalence between $*$ integral and Marcus integral, here we aim to provide another strategy to pathwisely compute these thermodynamic quantities based on Marcus integral.

Consider the d-dimensional SDEs (27) and assume the thermodynamic quantity Y satisfies

$$\mathrm{d}Y(t) = h(\boldsymbol{X}(t), t)\mathrm{d}t + \boldsymbol{q}(\boldsymbol{X}(t), t) \diamond \mathrm{d}\boldsymbol{L}(t), \tag{39}$$

where $\boldsymbol{q}(t), \boldsymbol{L}(t) \in \mathbb{R}^u$. Treating $(\boldsymbol{X}^T(t), Y(t))^T$ as a $(d+1)$-dimensional variable, we can obtain

$$\mathrm{d}\begin{pmatrix} \boldsymbol{X}(t) \\ Y(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}(\boldsymbol{X}(t), t) \\ h(\boldsymbol{X}(t), t) \end{pmatrix} \mathrm{d}t + \begin{pmatrix} \boldsymbol{g}(\boldsymbol{X}(t), t) \\ \boldsymbol{q}(\boldsymbol{X}(t), t) \end{pmatrix} \diamond \mathrm{d}\boldsymbol{L}(t). \tag{40}$$

According to the multidimensional version of the Marcus mapping (29), the induced state change of $(\boldsymbol{X}^T(t), Y(t))^T$ at jump time t of $\boldsymbol{L}(t)$ with jump size $\Delta\boldsymbol{L}(t)$ can be computed by solving the ODEs

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}s} &= \boldsymbol{g}(\boldsymbol{x}, t-) \cdot \Delta\boldsymbol{L}(t), \quad \boldsymbol{x}(0) = \boldsymbol{X}(t-), \\ \frac{\mathrm{d}y}{\mathrm{d}s} &= \boldsymbol{q}(\boldsymbol{x}, t-) \cdot \Delta\boldsymbol{L}(t), \quad y(0) = Y(t-), \end{aligned} \tag{41}$$

and $\boldsymbol{X}(t) = \boldsymbol{x}(1)$, $Y(t) = y(1)$. We note that the same idea has also been used in (16) and (17).

The above formulation directly leads to a computational method for any thermodynamic quantity $Y(t)$ satisfying (39) by numerically integrating the ODE system (41). But this can be done more efficiently when a realization of $\boldsymbol{X}(t)$ is given, which is the usual case in practice. Now suppose we have already got one trajectory of $\boldsymbol{X}(t)$, and the jump of $\boldsymbol{L}(t)$ is characterized by $(\tau_i, \boldsymbol{R}_i)$ indexed from $i = 1$, where τ_i and $\boldsymbol{R}_i$ are the jump time and the jump amplitude, respectively. In particular, we denote the Marcus mapping at time τ_i as $\boldsymbol{x}_i(s)$ for $s \in [0, 1]$. Then, we can obtain the following algorithm to compute thermodynamic quantities.

Algorithm 2 (Computing thermodynamic quantities)

1. *Given $t = 0$ (denoted as τ_0), initial state $Y(0)$ and the end time T. Let $i = 0$.*
2. *Integrate the drift part, i.e., compute the following integral ΔY_{i+1}^{out} by using a numerical quadrature scheme (for example, the midpoint scheme), and let $Y(\tau_{i+1}-) = Y(\tau_i) + \Delta Y_{i+1}^{out}$. Here*

$$\begin{aligned} \Delta Y_{i+1}^{out} &= \int_{\tau_i}^{\tau_{i+1}} h(\boldsymbol{X}(t), t)\mathrm{d}t \\ &\approx \frac{1}{2}\sum_{j=0}^{n-1} (h(\boldsymbol{X}(t_j), t_j) + h(\boldsymbol{X}(t_{j+1}), t_{j+1}))\delta t_j, \end{aligned} \tag{42}$$

   where $\tau_i = t_0 < t_1 < \ldots < t_n = \tau_{i+1}$ and $\delta t_j = t_{j+1} - t_j$.
3. *Integrate the jump part, i.e., compute the following integral ΔY_{i+1}^{in} by using a numerical quadrature scheme (for example, the midpoint scheme), and let $Y(\tau_{i+1}) = Y(\tau_{i+1}-) + \Delta Y_{i+1}^{in}$. Here*

$$\begin{aligned} \Delta Y_{i+1}^{in} &= \int_0^1 \boldsymbol{q}(\boldsymbol{x}_{i+1}(s), \tau_{i+1}-) \cdot \boldsymbol{R}_{i+1}\mathrm{d}s \\ &\approx \frac{1}{2}\sum_{j=0}^{n-1} (p(s_j) + p(s_{j+1}))\delta s_j, \end{aligned} \tag{43}$$

   where $p(s) := \boldsymbol{q}(\boldsymbol{x}_{i+1}(s), \tau_{i+1}-) \cdot \boldsymbol{R}_{i+1}$, $0 = s_0 < s_1 < \ldots < s_n = 1$ and $\delta s_j = s_{j+1} - s_j$.
4. *Set $i = i+1$, go to Step 2 unless $\tau_i \geq T$.*

We note that the midpoint quadrature can be replaced by any other higher order methods as well. We highlight that not only the history of $\boldsymbol{X}(t)$ is necessary for computing $Y(t)$ but also the hidden path $\boldsymbol{x}_i(s)$—path generated by Marcus mapping—is an indispensable component of the solution. This observation also provides insights about how the non-Gaussianity of noises affect the statistic properties of stochastic systems. Moreover, this algorithm only involves the numerical integration without any smoothing step or solving ODEs. In Subsections IV B and IV C, we will use this computational strategy to compute different thermodynamic quantities and demonstrate its efficiency.

B. The first law of thermodynamics in an overdamped Langevin equation

Consider the following one-dimensional overdamped Langevin equation

$$\mathrm{d}X(t) = -\nabla U(X(t))\mathrm{d}t + g(X(t)) \diamond \mathrm{d}L(t). \tag{44}$$

where $X(t)$, t, and $U(X(t))$ are dimensionless position, time, and potential energy. L is a compound Poisson process with intensity $\lambda/2$ corresponding to the jumpsize $\pm I$, respectively. Here we choose $U(x) = x^2/2 + \epsilon x^4/4 + \sin x$ and $g(x) = x$. In the terminology of stochastic energetics,[2] U is the total energy

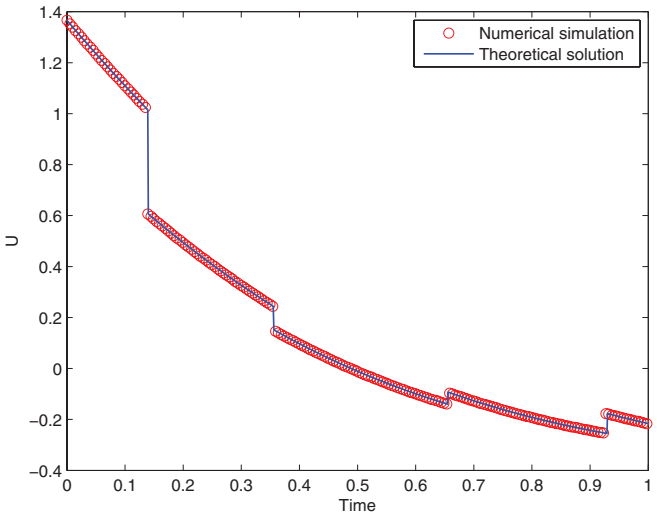

FIG. 4. The red circle corresponds to the heat $\hat{U}$ based on Marcus integral and the blue solid line corresponds to the total energy U. Some parameters are set as $\lambda = 5$, $\lambda I^2 = 1$, $\epsilon = 0.1$ and the time stepsize $\delta s = 0.01$ for solving both drift and jump ODEs.

and $\hat{U}$ defined as $\int_0^t \nabla U(X(s)) \diamond \mathrm{d}X(s)$ is the heat. The first law of thermodynamics tells us that $\hat{U}(X(t))$ should be identical to $U(X(t)) - U(X(0))$ in the pathwise sense. Now we come to verify this relation numerically. According to the chain rule of Marcus integral, $\hat{U}(X(t))$ satisfies

$$\begin{aligned} \mathrm{d}\hat{U}(X(t)) &= \nabla U(X(t)) \diamond \mathrm{d}X(t) \\ &= -|\nabla U(X(t))|^2 \mathrm{d}t + \nabla U(X(t)) g(X(t)) \diamond \mathrm{d}L(t). \end{aligned}$$

We use Algorithm 1 to solve the SDE (44). Then, with the information of SDE (44), $\hat{U}$ can be computed by Algorithm 2. Numerically, We choose $\epsilon = 0.1$, $\lambda = 5$, $\lambda I^2 = 1$, $\Delta t = 0.01$ for both (33) and (34), $X(0) = 1$ and $T = 1$. Fig. 4 shows one sample of the result. The heat $\hat{U}$ (shown with $*$) is almost the same as the total energy U (shown with solid line), demonstrating the first law of thermodynamics.

C. Heat measurement formula

Consider the following underdamped Langevin equation

$$\mathrm{d}x = \frac{p}{m}\mathrm{d}t, \tag{45}$$

$$\mathrm{d}p = -\gamma p \mathrm{d}t - \nabla U(x, \alpha)\mathrm{d}t + g(x, p) \diamond \mathrm{d}L. \tag{46}$$

In the context of stochastic energetics,[2,8] the total energy difference is divided into two parts

$$\mathrm{d}E = \mathrm{d}Q + \mathrm{d}W, \tag{47}$$

where $E = p^2/2m + U(x, \alpha)$ and

$$\begin{aligned} \mathrm{d}W &= \frac{\partial U}{\partial \alpha}\mathrm{d}\alpha, \\ \mathrm{d}Q &= \frac{-\gamma p^2}{m^2}\mathrm{d}t + \frac{g(x, p)p}{m} \diamond \mathrm{d}L. \end{aligned} \tag{48}$$

Here, W and Q are respectively termed as applied work and heat respectively.

In Ref. 2, a formula for heat measurement is developed for additive non-Gaussian noises. However, this formula may become very complicate when noise is multiplicative, as stated in Ref. 2. With the strategy developed in Sec. IV A, we can obtain a compact formula for heat measurement. Now let us denote $(x, p)^T$ and Q as $\boldsymbol{X}$ and Y respectively, and let $\boldsymbol{L} = L$. Then, the functions in Eq. (40) have the form $f(\boldsymbol{X}) = (p/m, -\gamma p - \nabla U(x, \alpha))^T$, $\boldsymbol{g}(\boldsymbol{X}) = (0, g(x, p))^T$, $h(\boldsymbol{X}) = -\gamma p^2/m^2$, and $\boldsymbol{q}(\boldsymbol{X}) = g(x, p)p/m$. Now, using the formula (41), we can obtain the following heat measurement formula

$$\begin{aligned} Q_t = Q_0 &- \int_0^t \frac{\gamma p^2}{m^2}\mathrm{d}s \\ &+ \sum_{k=1}^{N(t)} \{\Phi_g((x, p, Q)|_{t=\tau_k-}, \Delta L(\tau_k)) - Q(\tau_k-)\}, \end{aligned} \tag{49}$$

where $\Phi_g((x, p, Q)|_{t=\tau_k-}, \Delta L(\tau_k))$ denotes the solution of z of the following ODEs:

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}s} &= 0, \quad x|_{s=0} = x(\tau_k-), \\ \frac{\mathrm{d}p}{\mathrm{d}s} &= g(x, p), \quad p|_{s=0} = p(\tau_k-), \\ \frac{\mathrm{d}z}{\mathrm{d}s} &= \frac{g(x, p)p}{m}, \quad z|_{s=0} = Q(\tau_k-), \end{aligned}$$

at time $s = 1$.

For the numerical aspect of heat Q, we consider the non-dimensional form. Choose $U(x, \alpha) = x^2/2$ and $g(x, p) = x$. In this situation, the first law of thermodynamics becomes that $\mathrm{d}E = \mathrm{d}Q$. We compute the heat Q by using Algorithm 2 and compare it with the total energy E. The results are shown in Fig. 5. The perfect matching between Q and E verifies the first law of thermodynamics in the case of multiplicative non-Gaussian noise.

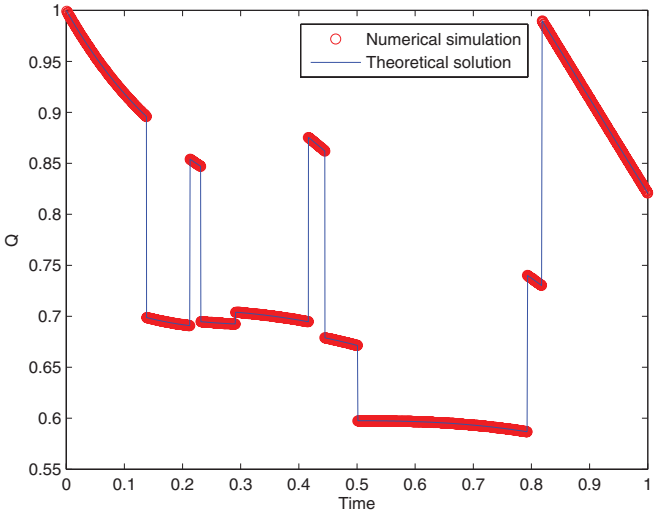

FIG. 5. The red circle corresponds to the numerical heat Q and the blue solid line corresponds to the theoretical energy E which equals to Q. $L(t)$ is a compound Poisson process with rate $\lambda = 10$ and $P(x \to x \pm I) = \lambda/2$, where I satisfies $\lambda I^2 = 1$. The stepsizes $\delta s = 0.01$ for both drift and jump ODEs.

V. TAU-LEAPING ALGORITHM

In some cases, we care more about the statistical average of physical quantities such as the mean, variance, or the distribution than the exact path. In these cases, it is not necessary to simulate the paths exactly and we are aiming at a speed-up of the computation. This speed-up can be realized through a fixed time stepping but with a cost of losing the exactness. This is similar to the problem in chemical reaction kinetics.[24] Gillespie proposed an algorithm called tau-leaping[24] instead of accurately simulating the trajectories in pathwise sense.[34] In this section we borrow the idea of tau-leaping to design a fixed time stepping scheme for stochastic differential equations driven by non-Gaussian noise.

A. Algorithm construction

Motivated by the tau-leaping algorithm in chemical reaction kinetics, we propose the following deterministic time stepping algorithm for Eq. (32) instead of simulating trajectories along each jump time. We restrict $L(t)$ to the class of compound Poisson process with finite kinds of jumps, such as the ATP reception model.[2] In mathematical setup, we suppose the jump intensity $\lambda = \sum_{j=1}^{k} \lambda_j$, where λ_j is the intensity corresponding to each kind of jump with jumpsize ΔL_j, respectively.

Algorithm 3 (Tau-leaping Algorithm)

1. *Given $t = 0, X(0)$, time stepsize Δt_f, Δt_g and the end time T. Here Δt_f and Δt_g are the time stepsizes for solving ODEs (33) and (34), respectively.*
2. *Tau-leaping step.*
   a. *Generate k Poisson random variables N_j with parameter $\lambda_j \Delta t_f$ for $j = 1{:}\ k$.*
   b. *Solve ODE (33) with any ODE solver, say forward Euler scheme, $x^f = X(t) + f(X(t))\Delta t_f$.*
   c. *Solve ODE (34) with initial date $X(t)$ to time $s = 1$ with jump size ΔL_j to get solution x_j^g for $j = 1{:}\ k$.*
3. *Update state*

$$X(t + \Delta t_f) = x^f + \sum_{j=1}^{k} \left(x_j^g - X(t)\right) N_j.$$

Set $t = t + \Delta t_f$. Go to Step 2 unless $t \geq T$.

A natural way to understand the above tau-leaping scheme can be explained as follows. Note that we can represent the solution of SDE (32) as

$$X(t + \Delta t) = X(t) + \int_t^{t+\Delta t} f(X(s))\mathrm{d}s + \sum_{j=1}^{k} \sum_{i=1}^{N_j(\Delta t)} \left\{\Phi_g\left(X\left(\tau_i^j -\right), \Delta L_i\right) - X\left(\tau_i^j -\right)\right\}. \tag{50}$$

where $N_j(\Delta t)$ is number of jumps occurring in $[t, t+\Delta t]$ in the jth class which are Poisson random variables with parameter $\lambda_j \Delta t$, and τ_i^j is the ith jump time in the jth class. To construct the tau-leaping algorithm, we introduce the forward Euler type approximation. That is, we freeze $X(s)$ and $X(\tau_i^j -)$ on the rhs as $X(t)$, which is the idea of forward Euler discretization in ODEs and chemical reaction kinetics. We obtain the numerical scheme

$$\hat{X}(t + \Delta t) = \hat{X}(t) + f(\hat{X}(t))\Delta t + \sum_{j=1}^{k} \sum_{i=1}^{N_j(\Delta t)} \{\Phi_g(\hat{X}(t), \Delta L_j) - \hat{X}(t)\}, \tag{51}$$

where $\hat{X}(t)$ is the numerical solution. The implementation is exactly the tau-leaping algorithm presented above.

To further improve the efficiency of the above algorithm, we can reduce the number of ODEs to be solved in case of the jump size can be linearly sorted. We notice that when $\Delta L \in \mathbb{R}^+$ the ODE

$$\frac{\mathrm{d}y}{\mathrm{d}s} = g(y)\Delta L,\ y(0) = x,\ s \in [0, 1] \tag{52}$$

can also be rewritten as

$$\frac{\mathrm{d}y}{\mathrm{d}s} = g(y),\ y(0) = x,\ s \in [0, \Delta L]. \tag{53}$$

This motivates us to sort the jumps in a linear order $0 < \Delta L_1 < \Delta L_2 < \cdots < \Delta L_k$. We only need to solve one ODEs from $s = 0$ to ΔL_k and the solution at intermediate time can be extracted automatically. When the jump $\Delta L_j < 0$, we may consider

$$\frac{\mathrm{d}y}{\mathrm{d}s} = -g(y),\ y(0) = x,\ s \in [0, -\Delta L_j]. \tag{54}$$

This trick is very useful when there are many types of jumps which will be shown in Sec. VI.

Usually it costs much more time to generate a Poisson random variable than an exponentially distributed random variable. We can further increase the computational efficiency if we generate less Poisson random variables. Recall the property that the sum of two independent Poisson random variables with intensity λ_1 and λ_2 is a Poisson random variable with intensity $\lambda_1 + \lambda_2$. So we propose the following efficient modified tau-leaping algorithm. We assume that ΔL_j has been sorted in increasing order.

Algorithm 4 (Modified Tau-leaping Algorithm)

1. *Given $t = 0$, $X(0)$, time step size Δt_f, Δt_g and the end time T. Here Δt_f and Δt_g are the time stepsize for solving ODEs (33) and (34), respectively.*
2. *Tau-leaping step.*
   a. *Generate a Poisson random variable N with parameter $\lambda \Delta t_f$, and generate N uniform distributed random variables u_i, $i = 1{:}\ N$.*
   b. *Set the number of jumps $N_j = 0$ $(j = 1{:}\ k)$. For $i = 1{:}\ N$, search for j such that $\sum_{l=1}^{j-1} \lambda_l \leq u_i < \sum_{l=1}^{j} \lambda_l$, set $N_j = N_j + 1$.*
   c. *Solve ODE (33) with any ODE solver, say forward Euler scheme, $x^f = X(t) + f(X(t))\Delta t_f$.*
   d. *If $\Delta L_k > 0$, solve ODE (53) with initial date $X(t)$ to time ΔL_k and keep track of $x^g(\Delta L_j)$.*
   e. *If $\Delta L_1 < 0$, solve ODE (54) with initial date $X(t)$ to time $-\Delta L_1$ and keep track of $x^g(-\Delta L_j)$.*

3. *Update state*

$$X(t+\Delta t_f) = x^f + \sum_{j=1}^{k} (x^g(\Delta L_j) - X(t))N_j.$$

Set $t = t+\Delta t_f$. Go to Step 2 unless $t \geq T$.

Though we will only focus on the Poisson noise in this paper, we simply remark that by mimicking the idea in constructing Algorithm 1′, we can extend the tau-leaping algorithm to SDEs driven by the Gaussian and non-Gaussian noises simultaneously. For instance, if the driving process is like (35), we only need to replace Step 2(c) in Algorithm 4 with the Step 3 in Algorithm 1′.

B. Tau-leaping condition

The main purpose to choose tau-leaping instead of pathwise algorithm is to speed up the simulation. It is expected to have as larger time stepsize as possible. However, it is widely known that large time step size may cause numerical instability and large numerical error. Thus we need to add some constrains on Δt to gain speedup but not lose accuracy at the same time. It is natural to require that $X(t)$ does not suffer a relatively big change, i.e.,

$$|X(t+\Delta t) - X(t)| \leq \epsilon |X(t)|, \tag{55}$$

where ϵ is a prescribed small number, say 0.1. Back to the tau-leaping scheme, the condition (55) is equivalent to demand

$$\begin{aligned} |\Phi_f(X, \Delta t) - X| &\leq \epsilon X, \\ \lambda \Delta t |\Phi_g(X, \Delta L) - X| &\leq \epsilon X \end{aligned} \tag{56}$$

in the average sense.

To make it more concise and explicit, we shall give an estimation to the solution of (34). Without loss of generality, we focus on one dimensional case at first. Due to the fact that we restrict the relative change to be small, we can expand the rhs in Taylor series and keep the terms up to order 1

$$\frac{\mathrm{d}x}{\mathrm{d}s} \approx g(x_0)R + g'(x_0)(x - x_0)R, s \in [0, 1].$$

Solving this equation to $s = 1$, we have an approximation to the exact solution

$$x(1) = x_0 + \frac{g(x_0)}{g'(x_0)}(e^{g'(x_0)R} - 1).$$

Putting this solution into (56), we get the more explicit tau-leaping condition

$$\Delta t \leq \min\left\{ \frac{\epsilon x_0 g'(x_0)}{\lambda g(x_0)(e^{g'(x_0)R} - 1)}, \frac{\epsilon x_0 f'(x_0)}{f(x_0)(e^{f'(x_0)} - 1)} \right\} \tag{57}$$

with another part from the ODE (33).

If $g'(x_0)R$ and $f'(x_0)$ is small, we can take $e^{g'(x_0)R} \approx 1 + g'(x_0)R$ and $e^{f'(x_0)} \approx 1 + f'(x_0)$. This leads to the following more concise form of tau-leaping condition:

$$\Delta t \leq \min\left\{ \frac{\epsilon x_0}{\lambda g(x_0)R}, \frac{\epsilon x_0}{f(x_0)} \right\}. \tag{58}$$

The above analysis motivates us the idea of switching between two algorithms when necessary. That is to say, when the timestep given by tau-leaping condition is very small, it is a better choice to do pathwise simulations. We can give a threshold Δt_0 so that when $\Delta t > \Delta t_0$ we use tau-leaping simulation and switch to pathwise simulation otherwise. However, this depends heavily on our knowledge about the SDEs we simulate. In Sec. V C, we will give a more reasonable switching strategy.

All the procedures above can be extended to high dimensional case by replacing x_0, R, g with $\boldsymbol{x}_0 \in \mathbb{R}^d$, $\boldsymbol{R} \in \mathbb{R}^u$, $\boldsymbol{g} \in \mathbb{R}^{d\times u}$. Following the previous analysis, we can give the more general form of tau-leaping condition as below

$$\Delta t \leq \min_i \left\{ \frac{\epsilon x_0^i}{\lambda y^i}, \frac{\epsilon x_0^i}{f^i(\boldsymbol{x}_0)} \right\}, \tag{59}$$

where $\boldsymbol{x}_0 = (x_0^1, x_0^2, \ldots, x_0^d)^T$, $\boldsymbol{y} = (y^1, y^2, \ldots, y^d)^T = \boldsymbol{g}(\boldsymbol{x}_0)\boldsymbol{R}$, and $\boldsymbol{f}(\boldsymbol{x}_0) = (f^1(\boldsymbol{x}_0), f^2(\boldsymbol{x}_0), \ldots, f^d(\boldsymbol{x}_0))$.

C. Efficiency analysis

When is the tau-leaping method more efficient than the pathwise simulation? If there are very few jumps, it is obvious that the performance of tau-leaping method may be not superior than the pathwise simulations. Thus we require the condition

$$\lambda \Delta t \gg 1 \tag{60}$$

to be satisfied, which means the jump must occur frequently. Otherwise, there is no need to apply tau-leaping algorithm. Below we analyze the efficiency of modified tau-leaping method when tau-leaping condition and (60) are satisfied.

The computational costs are comprised of three parts: (i) solving ODEs (33), (ii) solving ODEs (53) and (54), (iii) generating random numbers. The cost saving for Part (i) is obvious since we only need to solve one ODE in tau-leaping but the pathwise simulation needs to solve $\mathcal{N}(\Delta t)$ ODEs in each time step, where $\mathcal{N}(\Delta t)$ is the Poisson random number with parameter $\lambda \Delta t \gg 1$. For the Part (ii), the pathwise simulation algorithm needs to solve λT ODEs in the whole time interval $[0, T]$ in average, but the tau-leaping algorithm only needs to solve $2T/\Delta t$ ODEs. Notice that the end time of the ODEs in pathwise simulation and tau-leaping are different, we take the notation t_g^τ for the average time of solving the ODEs (53) and (54) once in tau-leaping simulation and t_g^p for solving ODEs (34) in pathwise simulation. If we denote T_g^τ and T_g^p the total computational cost in solving jump ODEs for tau-leaping and pathwise simulation respectively, we can define the *acceleration ratio* (or *boosting factor*)

$$r = \frac{T_g^p}{T_g^\tau} \approx \frac{\lambda \Delta t t_g^p}{2 t_g^\tau} \tag{61}$$

to characterize the speedup on solving ODEs in Part (ii). For a given SDE, the boosting factor r depends only on Δt. This leads to the following switching strategy naturally. We generate a stepsize Δt by the tau-leaping condition (58) at first. Then put it into (61) to get the boosting factor r. We may choose tau-leaping algorithm if $r > r_0$ and switch to pathwise simulation otherwise, where r_0 is a threshold set up by users. This choice means we can achieve *at least* r_0 times speedup *if the whole simulation is done with tau-leaping under this thresholding strategy*. The threshold r_0 also gives a definition of (60). At the same time, we should note that the real computational cost is not an increasing function of r_0 since bigger r_0 will make the condition of using tau-leaping more stringent and thus switching to pathwise simulation occurs more frequently. This may increase the computational effort of course, but result in better accuracy. For concrete systems, there will be a trade-off between the efficiency and accuracy by choosing a suitable r_0. This point is shown in the numerical example C in Sec. VI.

Now let us discuss the efficiency associated with Part (iii). T_r^{τ}, $T_r^{\tau'}$, and T_r^{p} denote the computational cost for generating random variables in modified tau-leaping (Algorithm 3), primitive tau-leaping (Algorithm 4), and pathwise simulation, respectively. τ_p, τ_u, and τ_e denote the cost for generating single Poisson random variable, uniformly distributed random variable, and exponentially distributed random variable, respectively. We have

$$T_r^{\tau} \approx \frac{T}{\Delta t}\tau_p + \lambda T \tau_u + t_{search}, \tag{62}$$

$$T_r^{\tau'} \approx \frac{kT}{\Delta t}\tau_p, \tag{63}$$

$$T_r^{p} \approx \lambda T \tau_e. \tag{64}$$

Usually $\tau_p \gg \tau_u$, while $\tau_u \sim \tau_e$. Thus if the jump type k is large, we expect our modified tau-leaping scheme spends less time on generating random variables than primitive tau-leaping from Eqs. (62) and (63). Comparing (62) and (64), we can see that tau-leaping scheme spends more time on generating Poisson random variables and doing search. Since the time for searching is usually small due to well-developed quick search algorithm and the problem we want to solve is usually of high dimensions, T_g will dominate the computation thus T_r for the extra cost for generating random variables can be neglected. In this case, we achieve the goal of speedup.

VI. NUMERICAL RESULTS BY TAU-LEAPING METHOD

A. Random motion near two parallel walls

The classical Brownian dynamics describing the movement of a particle perturbed by the noise has the form

$$\mathrm{d}X(t) = -\nabla U(X(t))\mathrm{d}t + g(X(t)) \diamond \mathrm{d}L(t). \tag{65}$$

Usually the diffusion function $D(x) = g^2(x)$ is a constant and $L(t)$ is chosen to be Brownian motion. But recently researchers are also interested in stochastic models in which the diffusion function depends on the position when the particle is bounded by two walls.[35,36] Under this circumstance, the diffusion function is given by $D(x) = 1-(x/B)^2$, where B is the half of the distance between two walls. Here we replace $L(t)$ with a compound Poisson process with equal jump intensity $\lambda_j = \lambda/6 = 400/6$ corresponding to jumpsizes $I = \pm 0.1, \pm 0.2$, and ± 0.3, respectively. With this choice we keep the strength of the noise $\lambda I^2 \sim O(1)$. The driving potential is chosen as $U(x) = (x-A)^2/2$. We want to compare the numerical solution $X(t)$ by pathwise and tau-leaping simulations. The other parameters are chosen as $A = 500$, $B = 1000$, and $X(0) = 500$ in this example.

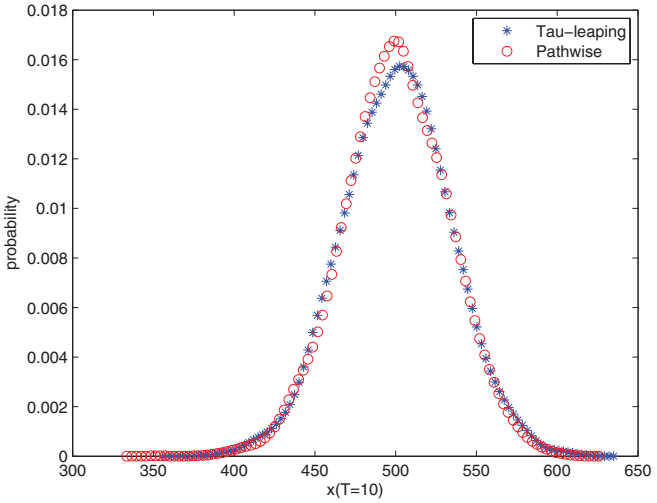

FIG. 6. The distribution of X at time $T = 10$. The symbol $*$ shows the distribution obtained by tau-leaping algorithm and $\circ$ shows the distribution obtained by pathwise simulations. Some parameters are $\lambda = 400$, $A = 500$, and $X(0) = B/2 = 500$. The time step is $\delta t = 0.01$ for pathwise simulations and $\Delta t = 0.06$ for tau-leaping method. Five thousand samples are simulated.

For pathwise simulations, we choose the second order Runge-Kutta method with uniform stepsize $\delta t = 0.01$ for both drift and jump ODEs, which guarantees the stability and accuracy. The jump intensity $\lambda = 400$ will force the time between two jumps usually smaller than δt. We will march with this smaller jump time to solve the drift ODEs instead of 0.01 in this case. We remark that the time stepsize δt is kept fixed just for confirming our previous efficiency analysis. To set the stepsize for tau-leaping algorithm, we need some knowledge about the SDE trajectories. Running pathwise simulations for 5000 times with stepsize $\delta t = 0.01$ until the end time $T = 10$, we find that the solution varies from 300 to 700. If we allow a relative change no more than 5%, an easy calculation with the tau-leaping condition (58) with $X = 300$ and $\epsilon = 0.05$ gives the largest allowed stepsize 0.065. Thus we choose $\Delta t = 0.06$ in the tau-leaping algorithm.

The accuracy of the tau-leaping algorithm is remarkable. We compare the histogram of X at $T = 10$ for both methods in Fig. 6, which shows that the tau-leaping algorithm catches the distribution very well. The mean value and standard deviation are listed in Table II. The relative error of these two quantities are 0.02% and 3.5%, respectively, which is accurate enough.

Now let us compare the efficiency of these two algorithms. The time cost for each part of simulations are listed in Table II. We can observe that the acceleration ratio for solving

TABLE II. t_r, t_f, t_g, and t_{total} are the time for generating random variables, solving the drift ODE (33), solving the jump ODEs (53) and (54), and the total computation time, respectively. Mean and Std are the mean value and standard deviation of $X(T = 10)$. Five thousand simulations are performed. The parameters are the same as those in Fig. 6.

	t_r	t_f	t_g	t_{total}	Mean	Std
Tau-leaping	0.062	0.031	3.45	3.55	500.038	29.754
Pathwise	0.051	0.61	27.62	28.28	499.985	30.882

the jump ODE (34) is $27.62/3.45 \approx 8.0058$. Now let us check our efficiency analysis for this. In tau-leaping method we need to solve the ODE to the end time $s = 0.3$ corresponding to the largest jumpsize and in pathwise simulations we need to solve the ODE to the end time $s = 0.2$ corresponding to the mean jumpsize. Thus the cost ratio for solving the jump ODE once is $t_g^p/t_g^\tau \approx 2/3$. With this, we get the expected acceleration ratio for solving the jump ODEs as

$$r \approx \frac{\lambda \Delta t}{2} \cdot \frac{t_g^p}{t_g^\tau} = 8$$

from (61), which is quite close to the previous numerical acceleration ratio. The whole acceleration ratio $r_{total} = t_{total}^p/t_{total}^\tau = 28.28/3.55 \approx 7.966$ which is also close to 8. This is because solving ODE (34) dominates the whole computations in this example.

From Table II we can also observe that the time for generating random variables is increased from 0.051 s to 0.062 s when using tau-leaping methods. This indicates that if the time for generating random variables cannot be neglected, we need a careful consideration on evaluating the efficiency of tau-leaping method. But in the current example, the time for generating random variables is very small compared to the whole time cost. It will be similar in the examples considered in Subsections VI B–VI D. Thus we will only compare the whole time cost instead of treating them separately. It can be easily seen that if the jump intensity increases, the number of ODEs needed to be solved increases too. Our example is a simple scalar model, and it is expected more time will be consumed for solving the high dimensional ODEs or more complicated ODE forms. Thus tau-leaping should be more preferred in those cases.

B. Langevin equation with double well potential

Now let us consider the stochastic differential equations

$$dX(t) = -\nabla U(X(t))dt + g(X(t)) \diamond dL(t), \tag{66}$$

with a double well potential $U(x) = A[(x-a)^2/b^2-1]^2$ whose plot is shown in Fig. 7(a). Here A describes the depth of the well, a is the local maximum point, and b is somehow the width of the well. The particle will fall into one of the two deep wells while the noise may drive the particle moving from one well to another. We expect a bistability of the distribution of X. The coefficient $g(x)$ of the noise is chosen the same as the previous example, that is $g(x) = \sqrt{1-(x/B)^2}$. $L(t)$ is a symmetric compound Poisson process with intensity λ equally divided by two jumps with sizes $\pm I$, respectively. We note that this equation can potentially model the adenosine triphosphate reception by red blood cell membranes.[2,18,19]

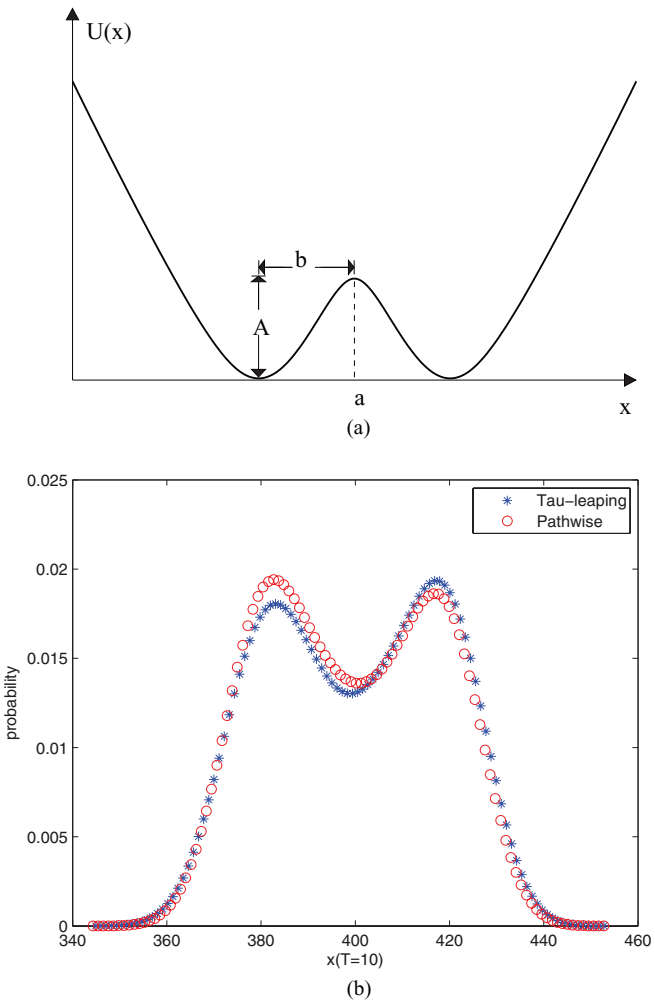

FIG. 7. Comparison between the pathwise and tau-leaping simulations with adaptive stepsize for double well potential. (a) Plot of the double well potential and (b) the distribution of X at time $T = 10$. Shown with blue stars is the distribution obtained by tau-leaping algorithm and red circles is the distribution obtained by pathwise algorithm. Five thousand realizations are simulated.

In the following numerical test, we choose the parameters as $A = 20$, $a = 400$, $b = 20$, $\lambda = 200$, $I = 0.6$, $B = 500$, $X_0 = 400$, and the parameter $\epsilon = 0.05$ in the tau-leaping condition. We want to find the distribution of X at time $T = 10$. In previous example, we fixed the time stepsize in tau-leaping algorithm. However, we will use tau-leaping condition to achieve adaptive time stepping here. The strategy is that at each time t, we utilize the tau-leaping condition to get a largest allowed stepsize and perform tau-leaping with it. In the considered example, the stepsize suggested by tau-leaping condition is always accepted and we do not need to switch to pathwise simulations as done in Subsection VI C. For the pathwise algorithm in comparison, we choose stepsize $\delta t = 0.01$ which is sufficient since the jump intensity $\lambda = 200$. We do 5000 realizations for each algorithm and compare their statistical behavior.

TABLE III. Comparison between the pathwise and tau-leaping simulations with adaptive stepsize for double well potential. The computation time is for 5000 simulations. Mean and Std are the mean value and standard deviation of $X(10)$.

	Time	Mean	Std
Tau-leaping	9.61*s*	400.12	17.73
Pathwise	97.7*s*	399.17	17.27

The distribution of X is shown in Fig. 7(b). We can observe that the distribution of X is bimodal as we expected. The distribution shown with blue $*$ symbol obtained by tau-leaping algorithm matches the result with red $\circ$ symbol obtained by pathwise algorithm well. The mean value and standard variation of $X(10)$ are also shown in Table III. The relative error for both mean and standard deviation are 0.24% and 2.67%, which is acceptable. This shows our tau-leaping condition works well and the tau-leaping algorithm keeps the accuracy in a tolerable range. At the same time tau-leaping algorithm only cost 9.61 s while the pathwise algorithm cost 97.7 s for the whole simulation. We achieve more than ten times acceleration.

From this example, we can see our algorithm works well for double well potential case. The distribution and some statistical quantities are caught with good accuracy. It is clear that how well the tau-leaping algorithm performs depends on the choice of time stepsize. Larger stepsize will surely accelerate the simulation more but it may cause the loss of accuracy. Our tau-leaping condition is shown to be effective on the trade-off between these two. Further improvement of the condition will be discussed in the continued works.

C. Langevin equation with periodic forcing

We consider the stochastic differential equations with periodic forcing

$$\mathrm{d}X(t) = (-\nabla U(X(t)) + f(t))\mathrm{d}t + g(X(t)) \diamond \mathrm{d}L(t) \quad (67)$$

in this subsection. Here $f(t)$ is an externally applied periodic forcing to the system. Without $f(t)$, the particle will move under the potential in the same way as the previous examples. The periodic driving force will push the particle periodically away from the equilibrium. We use such an example to examine our switching strategy between tau-leaping algorithm and pathwise algorithm. We call it *adaptive tau-leaping algorithm*. We choose the potential $U(x) = (x-A)^2/2$ and $f(t) = a\sin t$ where a is the strength of the periodic force. $g(x)$ and $L(t)$ have the same form as previous examples.

In the numerical test, we choose the parameters as $A = 0$, $a = 400$, $\lambda = 400$, $I = 0.4$, $B = 500$, $X(0) = 300$, $\epsilon = 0.05$ in the tau-leaping condition and the threshold of the boosting factor $r_0 = 1$ for switching. We want to find the distribution of X at time $T = 10$. Our switching strategy is as follows. In each time step we get a largest allowed stepsize Δt and compute the boosting factor r by (61). If $r > r_0$, we do one step tau-leaping with stepsize Δt, otherwise we switch to pathwise simulation until the next jump of $L(t)$. We choose stepsize in the pathwise algorithm as the previous example. Five thousand realizations are performed for extracting the statistical quantities.

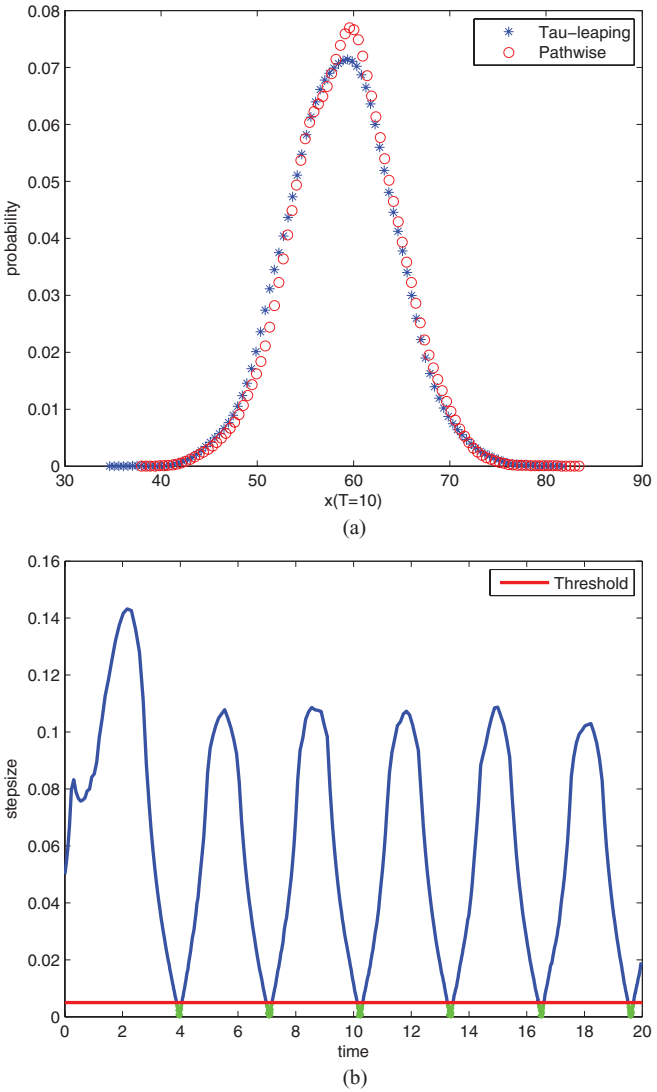

FIG. 8. Comparison between the pathwise and tau-leaping simulations with adaptive switching for periodic forcing. (a) The distribution of X at time $T = 10$. Shown with blue stars is the distribution obtained by tau-leaping algorithm and red circles is the distribution obtained by pathwise algorithm. Five thousand realizations are simulated. (b) Time history of utilized stepsizes by adaptive tau-leaping algorithm. The red horizontal line corresponds to the threshold Δt_0. The blue curve corresponds to the stepsizes used in tau-leaping steps while the green dots corresponds to the stepsizes used in pathwise simulation steps.

The comparison of the distribution of X between adaptive tau-leaping and pathwise simulations is shown in Fig. 8(a). We can observe that the tau-leaping algorithm catches the distribution very well. The mean value and standard variation of X are also shown in Table IV. The relative error for both mean and standard deviations are 0.46% and 0.79% which is quite accurate. If we check one typical sample of the utilized time stepsizes in the adaptive tau-leaping algorithm, we have Fig. 8(b). Based on the boosting factor threshold r_0, we have the stepsize threshold $\Delta t_0 = 2r_0/\lambda = 0.005$ in tau-leaping algorithm, which is shown as the red horizontal line in Fig. 8(b). When the stepsize Δt given by tau-leaping condition is above the threshold Δt_0, we choose tau-leaping which

TABLE IV. Comparison between the pathwise and adaptive tau-leaping simulations with different thresholds r_0 for periodic forcing. The computation time is for 5000 simulations. Mean and Std are the mean value and standard deviation of $X(10)$.

	Time (s)	Mean	Std
Pathwise	162.61	59.062	5.314
Tau-leaping $r_0 = 1$	29.28	58.798	5.356
Tau-leaping $r_0 = 2$	30.93	58.853	5.420
Tau-leaping $r_0 = 3$	35.15	58.444	5.361
Tau-leaping $r_0 = 4$	39.95	58.785	5.302
Tau-leaping $r_0 = 5$	45.06	58.818	5.360

corresponds to the blue curve in Fig. 8(b). Otherwise we switch to pathwise simulation until next jump occurs, which gives the stepsizes shown with green dots in Fig. 8(b). This plot of the history of utilized stepsizes shows the effectiveness of our adaptive strategy. The computational cost for both methods is shown in Table IV. We observe that the adaptive tau-leaping achieves about five to six times boosting than the pathwise simulations but with reasonable accuracy.

The threshold r_0 plays an important role here. Bigger r_0 means accelerating the simulation more when doing tau-leaping steps but also putting more time on pathwise simulations which will cost more time. The time cost of adaptive tau-leaping simulation with different r_0 is also shown in Table IV. It is clear that time cost increases as r_0 increases. On the other hand, It is natural to think that if we do more pathwise simulations, the accuracy should be better. However, for this example, the accuracy is not improved monotonically. We attribute this to the statistical fluctuations since the result has been accurate enough.

We emphasize that it is necessary to switch to pathwise simulations when the permitted time stepsize for tau-leaping is too small. Furthermore, tau-leaping may introduce some non-physical solutions such as the negative populations as in chemical reaction kinetics.[24] The switching to pathwise simulations can somehow alleviate this possibility in many situations. In any case, the adaptivity is necessary and effective.

D. High dimensional case with multiplicative noise

We consider the application of the tau-leaping algorithm for high dimensional case in this subsection. The equation reads

$$\mathrm{d}\boldsymbol{X}(t) = -\nabla U(\boldsymbol{X}(t))\mathrm{d}t + \boldsymbol{g}(\boldsymbol{X}(t)) \diamond \mathrm{d}L(t), \tag{68}$$

where the potential is chosen to be $U(\boldsymbol{x}) = \sum_{i=1}^{4}(x_i - A_i)^2/20$, the jumping term is $g_i(\boldsymbol{x}) = \sqrt{1-(x_i/B_i)^2}$ and L is a compound Poisson process with equal intensity $\lambda/2$ corresponding to jumpsizes $\pm I$. We want to check our high dimensional tau-leaping condition (59) with this example.

In our numerical test, we choose the parameters as $\lambda = 200$, $I = 1$, $A_1 = A_3 = 400$, $A_2 = A_4 = 420$, $B_1 = B_2 = 1000$, $B_3 = B_4 = 900$, $\boldsymbol{X}(0) = (370, 400, 380, 410)^T$. We want to find the distribution of $\boldsymbol{X}$ at $T = 10$. The stepsize in pathwise simulation is choosen to be 0.01 and the stepsize in tau-leaping simulation is chosen by tau-leaping condition (59). In this example, the stepsize suggested by tau-leaping condition is always accepted thus we do not need to switch to the pathwise simulations. Both algorithms are simulated 5000 times.

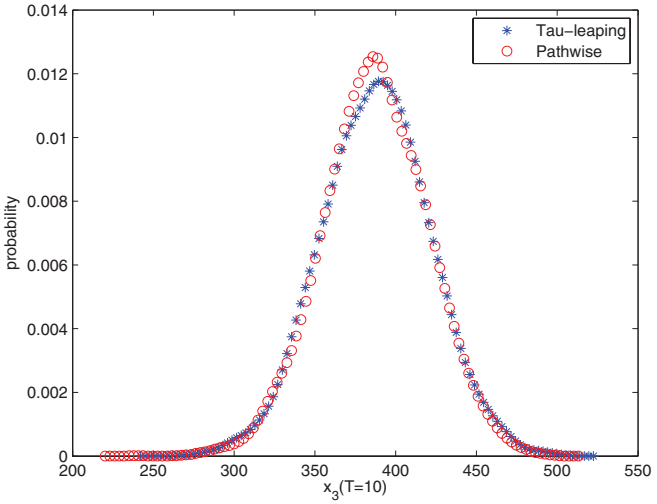

FIG. 9. The distribution of X_3 at time $T = 10$. The symbol $*$ shows the distribution obtained by tau-leaping algorithm and $\circ$ shows the distribution obtained by pathwise simulations.

The comparison of the distribution of X_3 with adaptive tau-leaping and pathwise simulation is shown in Fig. 9. We can observe that the tau-leaping algorithm catches the distribution well. The mean value and standard variation of $\boldsymbol{X}$ are shown in Table V. The relative error of mean and standard variation for all variables are smaller than 3.15%, which is highly accurate. This shows our tau-leaping condition for high dimensional cases works well and the tau-leaping algorithm keeps the accuracy in a tolerable range. At the same time, tau-leaping simulation costs only 45.59 s while the pathwise simulation costs 944.9 s. The acceleration is amazing due to the high dimensionality of the stochastic differential equations.

In this example, the jump ODEs needed to be solved are not so complicated. But the efficiency of the tau-leaping has been exhibited. In the case of much more complicate jump ODEs involved, tau-leaping will be more preferred if applicable since it will save much time for solving the jump ODEs for pathwise simulations if the jump occurs very frequently. That is why tau-leaping algorithm functions for the considered examples.

TABLE V. Comparison between the pathwise and tau-leaping simulations with adaptive stepsize for high dimensional equations. The computation time is for 5000 simulations. In each column of X_i, the mean value and standard variation of $X_i(10)$ are listed.

	Time	X_1	X_2
Tau-leaping	45.69s	377.67/33.63	408.71/33.51
Pathwise	944.9s	377.25/32.60	407.61/32.49
	X_3	X_4	
Tau-leaping	387.72/33.27	418.12/33.14	
Pathwise	386.21/32.26	419.01/32.14	

VII. CONCLUSION

In this paper, we give a comprehensive introduction to Marcus integral and compare two equivalent definitions in the literature: the Taylor series formulation by Di Paola-Falsone formula and the ODE formulation by Marcus mapping. We introduce the exact pathwise simulation algorithm based on the Marcus mapping and give the error analysis. We show how to compute the thermodynamic quantities in stochastic energetics based on the Marcus integral instead of the smoothing approach existing in the literature. We further propose the tau-leaping algorithm, which advances the process with deterministic time steps when tau-leaping condition is satisfied. The efficiency analysis shows that it can significantly speed up the simulation even for small systems without losing much accuracy. The numerical experiments verify the analysis. We believe the proposed tau-leaping algorithm is promising in the context of SDEs driven by non-Gaussian processes. Further studies including applications to larger systems will be investigated in the future.

ACKNOWLEDGMENTS

The authors acknowledge the support from the National Science Foundation of China (NSFC) under Grant Nos. 11171009 and 91130005 and the National Science Foundation for Excellent Young Scholars (Grant No. 11222114).

APPENDIX: PROOF OF THEOREM 1

Firstly, let us consider an arbitrary ODE

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(y), \quad y(0) = y_0. \tag{A1}$$

We take the notation $y(t)$ and $y^n(t)$ for the exact solution and numerical solution. Define the absolute error $e(t) = |y(t) - y^n(t)|$. If the numerical solution is obtained by using an ODE solver of pth order and stepsize h, we have the following estimate:

$$e(T) \le K_f e(0) + (K_f - 1)Ch^p, \tag{A2}$$

where $K_f = \exp(L_f T)$, L_f is the Lipschitz constant of function f and C is independent of h and T.

Based on this, we focus on one sample path of the SDE

$$\mathrm{d}X(t) = f(X(t))\mathrm{d}t + g(X(t)) \diamond \mathrm{d}L(t). \tag{A3}$$

Denote σ_i the ith jump time and $\sigma_0 = 0$. Thus $\tau_i = \sigma_i - \sigma_{i-1}$ is the gap time between the ith and $(i-1)$th jump. Recall that the pathwise simulation algorithm is composed of alternatively solving the drift ODE from σ_{i-1} to σ_i using a pth order ODE solver with stepsize h_1 and solving the jump ODE at time σ_i using a qth order ODE solver with stepsize h_2. With the general error estimate (A2), we get the error estimates in the two alternating steps as

$$e(\sigma_i -) \le F_i e(\sigma_{i-1}) + (F_i - 1)C_1 h_1^p,$$

$$e(\sigma_i) \le G_i e(\sigma_i -) + (G_i - 1)C_2 h_2^q,$$

where $F_i = \exp(L_f \tau_i)$ and $G_i = \exp(L_g |R_i|)$. L_g is the Lipschitz constant of function g. C_1 and C_2 are independent of h and τ, and R_i is the ith random jumpsize. Combining the two estimates above together, we get the error propagation from σ_{i-1} to σ_i as below

$$\begin{aligned} e(\sigma_i) \le\ & F_i G_i e(\sigma_{i-1}) + C_1(F_i - 1)G_i h_1^p \\ & + C_2(G_i - 1)h_2^q, \end{aligned} \tag{A4}$$

where $F_i = \exp(L_f \tau_i)$ and $G_i = \exp(L_g |R_i|)$.

Now we iterate the estimation from $t = 0$ to $\sigma_1, \sigma_2 \cdots \sigma_N$, $t = T$ and we get

$$\begin{aligned} e(T) \le\ & K_f \left(\prod_{i=1}^{N} G_i \right) e(0) + C_1(\exp(L_f(T - \sigma_N)) - 1)h_1^p \\ & + C_1 \sum_{i=1}^{N-1} \left(\prod_{j=i+1}^{N} F_j G_j \right) (F_i - 1)G_i h_1^p \exp(L_f(T - \sigma_N)) \\ & + C_2 \sum_{i=1}^{N-1} \left(\prod_{j=i+1}^{N} F_j G_j \right) (G_i - 1)h_2^q \exp(L_f(T - \sigma_N)), \end{aligned}$$

where N is a Poisson random number with parameter λT. Note that $1 \le \prod_{j=i}^{N} F_j \exp(L_f(T - \sigma_N)) = \exp(L_f(T - \sigma_{i-1})) \le K_f$ and $\prod_{j=i}^{N} G_j \le \prod_{j=1}^{N} G_j$, we have

$$\begin{aligned} e(T) \le\ & K_f \left(\prod_{i=1}^{N} G_i \right) e(0) + C_1(\exp(L_f(T - \sigma_N)) - 1)h_1^p \\ & + C_1 \sum_{i=1}^{N-1} \left(\prod_{j=i}^{N} F_j - \prod_{j=i+1}^{N} F_j \right) \exp(L_f(T - \sigma_N)) \\ & \times \left(\prod_{j=1}^{N} G_j \right) h_1^p + C_2 \sum_{i=1}^{N-1} \left(\prod_{j=i}^{N} G_j - \prod_{j=i+1}^{N} G_j \right) K_f h_2^q \\ \le\ & K_f \left(\prod_{i=1}^{N} G_i \right) e(0) + C_1(K_f - 1) \left(\prod_{j=1}^{N} G_j + 1 \right) h_1^p \\ & + C_2 \left(\prod_{j=1}^{N} G_j - G_N \right) K_f h_2^q. \end{aligned}$$

With the assumption $K_g = \mathbb{E} \exp(L_g |R|) < \infty$ and taking the conditional expectation on the equation above, we obtain

$$\begin{aligned} \mathbb{E}(e(T)|N = n) \le\ & K_f K_g^n \mathbb{E}(e(0)|N = n) \\ & + C_1(K_g^n + 1)(K_f - 1)h_1^p \\ & + C_2 K_f (K_g^n - 1)h_2^q. \end{aligned}$$

Finally let us take expectation with respect to the Poisson random variable N and using the fact $\mathbb{E} K_g^N = \exp(\lambda T(K_g - 1))$, we have

$$\begin{aligned} \mathbb{E}e(T) \le\ & \exp(L_f T + \lambda T(K - 1))\mathbb{E}e(0) \\ & + C_1(\exp(\lambda T(K - 1)) + 1)(\exp(L_f T) - 1)h_1^p \\ & + C_2 \exp(L_f T)(\exp(\lambda T(K - 1)) - 1)h_2^q, \end{aligned} \tag{A5}$$

which ends the proof.

[1]Ya. M. Blanter and M. Büttiker, Phys. Rep. **336**, 1 (2000).
[2]K. Kanazawa, T. Sagawa, and H. Hayakawa, Phys. Rev. Lett. **108**, 210601 (2012).
[3]W. Schoutens, *Levy Processes in Finance: Pricing Financial Derivatives* (Wiley, 2003).
[4]M. Schürmann, *White Noise on Bialgebras*, Lecture Notes in Mathematics Vol. 1544 (Springer-Verlag, Berlin, 1991).
[5]D. Applebaum, *Lévy Processes and Stochastic Calculus* (Cambridge University Press, Cambridge, 2004).
[6]K. Sato, *Lévy Process and Infinitely Divisible Distributions* (Cambridge University Press, Cambridge, 1999).
[7]C. Bustamante, J. Liphardt, and F. Ritort, Phys. Today **58**(7), 43 (2005).
[8]K. Sekimoto, *Stochastic Energetics* (Springer-Verlag, Berlin, 2010).
[9]U. Seifert, Rep. Prog. Phys. **75**, 126001 (2012).
[10]J. Liphardt, S. Dumont, S. B. Smith, I. Tinoco, Jr., and C. Bustamante, Science **296**, 1832 (2002).
[11]S. Toyabe *et al.*, Nat. Phys. **6**, 988 (2010).
[12]D. Collin *et al.*, Nature (London) **437**, 231 (2005).
[13]E. H. Trepagnier *et al.*, Proc. Natl. Acad. Sci. U.S.A. **101**, 15038 (2004).
[14]E. Wong and M. Zakai, Ann. Math. Stat. **36**, 1560 (1965).
[15]H. Sussmann, Ann. Probab. **6**, 19 (1978).
[16]P. Reimann, Phys. Rep. **361**, 57 (2002).
[17]J. Luczka, T. Czernik, and P. Hangii, Phys. Rev. E **56**, 3968 (1997).
[18]N. Gov, Phys. Rev. Lett. **93**, 268104 (2004).
[19]E. Ben-Isaac, Y. K. Park, G. Popescu, F. L. H. Brown, N. S. Gov, and Y. Shokef, Phys. Rev. Lett. **106**, 238103 (2011).
[20]M. Hoare, Adv. Chem. Phys. **20**, 135 (1971).
[21]T. G. Kurtz, E. Pardoux, and P. Protter, Ann. Inst. H. Poincare B **31**, 351 (1995) (available online at http://cat.inist.fr/?aModele=afficheN&cpsidt=3452872).
[22]S. I. Marcus, IEEE Trans. Inf. Theory **24**(2), 164 (1978).
[23]S. I. Marcus, Stochastics **4**, 223 (1981).
[24]D. Gillespie, J. Phys. Chem. **115**, 1716 (2001).
[25]T. Li, Multiscale Model. Simul. **6**, 417 (2007).
[26]X. Sun, J. Duan, and X. Li, Probab. Eng. Mech. **32**, 1 (2013).
[27]C. W. Gardiner, *Handbook of Stochastic Methods for Physics, Chemistry and Natural Sciences* (Springer-Verlag, Berlin, 2004).
[28]M. Di Paola and G. Falsone, Probab. Eng. Mech. **8**(3), 197 (1993).
[29]M. Di Paola and G. Falsone, ASME J. Appl. Mech. **60**, 141 (1993).
[30]P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations* (Springer-Verlag, Berlin, 1999).
[31]M. Di Paola and M. Vasta, Int. J. Non-Linear Mech. **32**(5), 855 (1997).
[32]U. Seifert, Phys. Rev. Lett. **95**, 040602 (2005).
[33]V. Blickle, T. Speck, L. Helden, U. Seifert, and C. Bechinger, Phys. Rev. Lett. **96**, 070603 (2006).
[34]D. Gillespie, J. Comput. Phys. **22**, 403 (1976).
[35]L. P. Faucheux and A. J. Libchaber, Phys. Rev. E **49**, 5158 (1994).
[36]A. W. C. Lau and T. C. Lubnesky, Phys. Rev. E **76**, 011123 (2007).