Computer Physics Communications 278 (2022) 108417

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

Fast algorithms of bath calculations in simulations of quantum system-bath dynamics ☆,☆☆

Zhenning Cai [a], Jianfeng Lu [b], Siyao Yang [a,*]

[a] Department of Mathematics, National University of Singapore, Level 4, Block S17, 10 Lower Kent Ridge Road, 119076, Singapore
[b] Department of Mathematics, Department of Physics, and Department of Chemistry, Duke University, Box 90320, Durham, NC 27708, USA

ARTICLE INFO

Article history:
Received 24 February 2022
Received in revised form 6 May 2022
Accepted 15 May 2022
Available online 20 May 2022

Keywords:
Dyson series
Inchworm Monte Carlo method
Integro-differential equation
Accelerated bath calculation
Fast algorithms

ABSTRACT

We present fast algorithms for the summation of Dyson series and the inchworm Monte Carlo method for quantum systems that are coupled with harmonic baths. The algorithms are based on evolving the integro-differential equations where the most expensive part comes from the computation of bath influence functionals. To accelerate the computation, we design fast algorithms based on reusing the bath influence functionals computed in the previous time steps to reduce the number of calculations. It is proven that the proposed fast algorithms reduce the number of such calculations by a factor of $O(N)$, where N is the total number of time steps. Numerical experiments are carried out to show the efficiency of the method and to verify the theoretical results.

1. Introduction

In classical thermodynamics, many processes are irreversible due to the dissipation of energy. To describe such an effect at the quantum level, quantum dissipation has been widely studied in the literature, and one of the successful approaches is the Caldeira-Leggett model [8,9], which assumes that the quantum system is coupled with a harmonic bath. The presence of the bath leads to non-Markovian and irreversible dynamics of the quantum system. The system-bath dynamics has also been extensively used to study quantum decoherence, which leads to classical behavior of the quantum systems. In addition to its theoretical importance, the model is widely used to describe interaction of a quantum system with its environment, and has applications in a number of fields including quantum optics [5], quantum computation [38], and dynamical mean field theory [20].

The main challenge for simulating Caldeira-Leggett type models lies in the huge degrees of freedom associated with the harmonic bath, which makes the direct calculation of the wave function impossible in practice. For decades, many techniques for dimension reduction have been developed in order to avoid solving the harmonic bath directly. Some classical numerical methods based on path integrals, such as the quasi-adiabatic propagator path integral (QuAPI) method [28,32], the iterative QuAPI-based methods [27,30] and the hierarchical equations of motion (HEOM) [45], introduce the bath effects using the influence functional [19] and can produce numerically exact results, while a considerably large memory cost is often required. A wave function-based approach known as the multiconfiguration time-dependent Hartree (MCTDH) method [2,33], as well as its multilayer formulation (ML-MCTDH) [47], has achieved impressive success in molecular systems, although they may become harder to converge for the nonequilibrium heat transport in the Caldeira-Leggett model [11].

Another conventional approach to the system-bath dynamics is the generalized quantum master equation (GQME) [52,37,34] obtained by applying the Nakajima-Zwanzig projection operator, which reduces the dissipative bath term to a memory kernel. Such formulation provides an exact integro-differential equation for simulating the reduced dynamics. However, the evaluation of the memory kernel could

☆ Zhenning Cai's work was supported by the Academic Research Fund of the Ministry of Education of Singapore under grant A-0004592-00-00. The work of JL was supported in part by the National Science Foundation via grants DMS-2012286 and CHE-2037263.
☆☆ The review of this paper was arranged by Prof. N.S. Scott.
* Corresponding author.
E-mail addresses: matcz@nus.edu.sg (Z. Cai), jianfeng@math.duke.edu (J. Lu), matsiya@nus.edu.sg (S. Yang).

https://doi.org/10.1016/j.cpc.2022.108417
0010-4655/

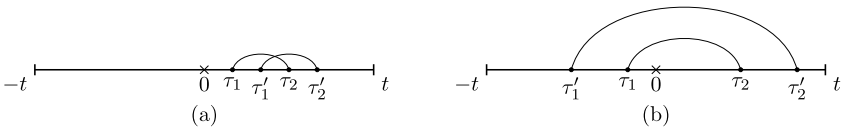


Fig. 1. Two cases of bath correlation invariance $B(\tau_1, \tau_2) = B(\tau_1', \tau_2')$.

be challenging due to its dependence on the projector. To alleviate this difficulty, [42,51] have proposed new approaches to calculating the memory kernel based on its projection-free formulations. The transfer tensor method (TTM) [10] based on the discretization of GQME is later introduced, which is also applied [41] in a method called the time evolving density matrix using orthogonal polynomials algorithm (TEDOPA) [13,39] to reduce the size of the propagator. Further development on the evaluation of memory kernel includes [24] where the memory kernel is related to the evolution of a reduced system propagator which is numerically computed by ML-MCTDH, and [22] which computes the memory kernel based on semiclassical trajectories.

An alternative to these deterministic approaches gaining popularity in the recent years is a class of stochastic methods known as the diagrammatic quantum Monte Carlo (dQMC) [40,49], which have been shown to be powerful in describing the equilibrium physics of impurity models. The underlying idea is to replace the expensive high-dimensional integrals in the Dyson series of the quantum observable by the average of unbiased samples of diagrammatic expansions [12,35,48]. For example in GQME, the memory kernel can be evaluated stochastically using the real time path integral Monte Carlo [16,15]. However, such an approach severely suffers from the notorious numerical sign problem [12,7,6], meaning that the variance of the numerical solution grows at least exponentially with time. To maintain the accuracy of the results, a large number of Monte Carlo samples need to be drawn as time increases, leading to an extremely expensive computational cost on the evaluation of the bath influence functional. To mitigate the sign problem, many techniques such as stochastic unraveling of influence functionals [44] and multilevel blocking Monte Carlo [26,18,36] have emerged throughout the past several decades. Recently, the inchworm Monte Carlo method [14,12] based on the partial resummation of Dyson series has been proposed, which has proven impressive capability to relieve the sign problem both numerically [11,7] and theoretically [6]. Nevertheless, the computations of the bath influence functional remain to be the major bottleneck [4,50] even after such reductions. In this paper, we consider a strategy to further reduce the cost of bath calculations in the summation of Dyson series and inchworm Monte Carlo method.

The central idea to reduce bath calculations lies in the invariance of the influence functional in Dyson series or inchworm method, which is formulated as a summation over some pairwise bath interactions. In detail, the expected value of an observable O can be written as $\mathrm{tr}\big(\rho(0)\mathrm{e}^{\mathrm{i}tH} O \mathrm{e}^{-\mathrm{i}tH}\big)$ with $\rho(0)$ being the initial density matrix and H the quantum Hamiltonian. In the diagrammatic Monte Carlo methods, such an expression is often denoted using the unfolded Keldysh contour [21] plotted in Fig. 1. By Wick's theorem, computing the trace requires us to evaluate the correlation function $B(\tau_1, \tau_2)$ of two time points $-t \le \tau_1 \le \tau_2 \le t$, which can be diagrammatically represented as an arc in Fig. 1. This two-point correlation function satisfies the translational invariance (when τ_1 and τ_2 are on the same side of the origin) and the stretching invariance (when τ_1 and τ_2 are on different sides of the origin). Making use of this property can greatly reduce the computational cost for the bath calculation.

Let us remark that the invariance of the two-point correlation function is also utilized in the recently proposed SMatPI (small matrix decomposition of the path integral) method [31], which is an improved version of the iterative QuAPI method. In SMatPI, the bath integrand factor is computed using the Feynman-Vernon influence functional. The SMatPI method groups a number of paths into some small matrices, and using the translational invariance of the Feymann-Vernon influence functional, the information in the small matrices can be directly used in future time steps without being recalculated. In our method, the bath influence functional is the sum of a lot of diagrams, and the reuse of previously calculated functionals avoids recomputation of all the translated or stretched diagrams included, which significantly enhances the computational efficiency.

The rest of this paper is organized as follows. In Section 2, we introduce the spin-boson model and its Dyson series expansion. An integro-differential equation associated with Dyson series is then derived, based on which we propose a fast algorithm where the previous bath calculations are reused. An analysis on computational cost is included to examine the performance of the proposed algorithm. Such framework is then applied to Section 3 where the more complicated inchworm Monte Carlo method is studied. Some numerical experiments are carried out in Section 4 to verify the theoretical results in Section 2 and 3, and test the order of convergence of the fast algorithms. Finally, some conclusions and discussions are given in Section 5.

2. Fast calculation of time evolution of Dyson series

2.1. Introduction to spin-boson model and Dyson series

We study the system-bath dynamics described by the von Neumann equation for the density matrix $\rho(t)$

$$\mathrm{i}\frac{\mathrm{d}\rho}{\mathrm{d}t} = [H, \rho] := H\rho - \rho H, \tag{1}$$

where the Schrödinger picture Hamiltonian H is a Hermitian operator on the Hilbert space $\mathcal{H} = \mathcal{H}_s \otimes \mathcal{H}_b$, with $\mathcal{H}_s$ and $\mathcal{H}_b$ representing respectively the Hilbert spaces associated with the system and the bath. The Hamiltonian H consists of the Hamiltonians of the system and the bath, as well as a coupling term describing the interaction of the system and the bath. Assuming that the coupling term has the tensor-product form, we have

$$H = H_s \otimes \mathrm{Id}_b + \mathrm{Id}_s \otimes H_b + W_s \otimes W_b,$$

where $H_s, W_s \in \mathcal{H}_s$, $H_b, W_b \in \mathcal{H}_b$, and $\mathrm{Id}_s, \mathrm{Id}_b$ are the identity operators for the system and the bath, respectively. In our paper, we take the common assumption that the bath is modeled by a larger number of harmonic oscillators. While the algorithms discussed in this work can be easily generalized to any multiple-state open quantum systems, we only consider the simplest system modeled by a single spin.

Such a problem contains most difficulties in the treatment of the system-bath coupling, which is known as the spin-boson model to be introduced below.

2.1.1. Spin-boson model

As one fundamental example of the system-bath dynamics [46,23,17], the spin-boson model assumes that

$$\mathcal{H}_s = \mathrm{span}\{|0\rangle, |1\rangle\}, \qquad \mathcal{H}_b = \bigotimes_{l=1}^{L} \left(L^2(\mathbb{R}^3)\right),$$

where L is the number of harmonic oscillators in the bath. The corresponding Hamiltonians are

$$H_s = \epsilon\hat{\sigma}_z + \Delta\hat{\sigma}_x, \qquad H_b = \sum_{l=1}^{L} \frac{1}{2}(\hat{p}_l^2 + \omega_l^2\hat{q}_l^2)$$

Here $\hat{\sigma}_x$, $\hat{\sigma}_z$ are Pauli matrices satisfying $\hat{\sigma}_x|0\rangle = |1\rangle$, $\hat{\sigma}_x|1\rangle = |0\rangle$, $\hat{\sigma}_z|0\rangle = |0\rangle$, $\hat{\sigma}_z|1\rangle = -|1\rangle$, and the parameters ϵ, Δ are respectively the energy difference between two spin states and the frequency of the spin flipping. In the bath Hamiltonian H_b, the notations $\hat{p}_l$, $\hat{q}_l$ and ω_l are respectively the momentum operator, the position operator and the frequency of the lth harmonic oscillator. The coupling operators are given by

$$W_s = \hat{\sigma}_z, \qquad W_b = \sum_{l=1}^{L} c_l\hat{q}_l,$$

where c_l is the coupling intensity between the lth harmonic oscillator and the spin.

The density matrix solving (1) can be written as $\rho(t) = \mathrm{e}^{-\mathrm{i}tH}\rho(0)\mathrm{e}^{\mathrm{i}tH}$, and we assume its initial value has the separable form $\rho(0) = \rho_s \otimes \rho_b$ with the initial bath ρ_b being the thermal equilibrium $\exp(-\beta H_b)$, where β is the inverse temperature. We are interested in the evolution of the expectation for a given observable $O = O_s \otimes \mathrm{Id}_b$ acting only on the system, defined by

$$\langle O(t)\rangle := \mathrm{tr}(O\rho(t)) = \mathrm{tr}(O\,\mathrm{e}^{-\mathrm{i}tH}\rho(0)\mathrm{e}^{\mathrm{i}tH}) = \mathrm{tr}(\rho_s \otimes \rho_b \mathrm{e}^{\mathrm{i}tH} O_s \mathrm{e}^{-\mathrm{i}tH}) = \mathrm{tr}_s(\rho_s G(-t,t)) \tag{2}$$

where the propagator $G(-t,t) := \mathrm{tr}_b(\rho_b \mathrm{e}^{\mathrm{i}tH} O_s \mathrm{e}^{-\mathrm{i}tH})$ is a 2×2 Hermitian matrix due to the cyclic property of the trace operator:

$$G(-t,t)^\dagger = \mathrm{tr}_b(\mathrm{e}^{\mathrm{i}tH^\dagger} O_s^\dagger \mathrm{e}^{-\mathrm{i}tH^\dagger}\rho_b^\dagger) = \mathrm{tr}_b(\mathrm{e}^{\mathrm{i}tH} O_s \mathrm{e}^{-\mathrm{i}tH}\rho_b) = \mathrm{tr}_b(\rho_b \mathrm{e}^{\mathrm{i}tH} O_s \mathrm{e}^{-\mathrm{i}tH}) = G(-t,t). \tag{3}$$

2.1.2. Dyson series

Due to the high dimensionality of the space $\mathcal{H}_b$, it is impractical to solve $\mathrm{e}^{\pm\mathrm{i}tH}$ directly. One feasible approach is to apply the method of quantum Monte Carlo to approximate $G(-t,t)$ numerically. It is well known that $G(-t,t)$ can be expanded into the following *Dyson series* (for derivation, see [7]):

$$G(-t,t) = \mathrm{e}^{\mathrm{i}tH_s} O_s \mathrm{e}^{-\mathrm{i}tH_s} + \sum_{m=1}^{+\infty} \mathrm{i}^m \int_{-t\le \boldsymbol{s}\le t} \mathrm{d}\boldsymbol{s}(-1)^{\#\{\boldsymbol{s}<0\}} \mathcal{U}^{(0)}(-t,\boldsymbol{s},t)\cdot\mathcal{L}_b(\boldsymbol{s}), \text{ for } t\ge 0. \tag{4}$$

The above formula is interpreted as:

- Integral notation: for any $a \le A$

$$\int_{a\le \boldsymbol{s}\le A} \mathrm{d}\boldsymbol{s} := \int_a^A \mathrm{d}s_m \int_a^{s_m} \mathrm{d}s_{m-1}\cdots\int_a^{s_2} \mathrm{d}s_1.$$

- $\#\{\boldsymbol{s}<0\}$: number of components in $\boldsymbol{s} = (s_1, s_2, \cdots, s_m)$ that are less than 0.
- System associated functional $\mathcal{U}^{(0)}$:

$$\mathcal{U}^{(0)}(-t,\boldsymbol{s},t) = G_s^{(0)}(s_m,t)W_sG_s^{(0)}(s_{m-1},s_m)W_s\cdots W_sG_s^{(0)}(s_1,s_2)W_sG_s^{(0)}(-t,s_1), \tag{5}$$

where

$$G_s^{(0)}(s_\mathrm{i},s_\mathrm{f}) = \begin{cases} \mathrm{e}^{-\mathrm{i}(s_\mathrm{f}-s_\mathrm{i})H_s}, & \text{if } s_\mathrm{i}\le s_\mathrm{f}<0,\\ \mathrm{e}^{-\mathrm{i}(s_\mathrm{i}-s_\mathrm{f})H_s}, & \text{if } 0\le s_\mathrm{i}\le s_\mathrm{f},\\ \mathrm{e}^{\mathrm{i}s_\mathrm{f}H_s}O_s\mathrm{e}^{\mathrm{i}s_\mathrm{i}H_s}, & \text{if } s_\mathrm{i}<0\le s_\mathrm{f}. \end{cases} \tag{6}$$

- Bath influence functional $\mathcal{L}_b$:

$$\mathcal{L}_b(s_1,\cdots,s_m) = \begin{cases} 0, & \text{if } m \text{ is odd};\\ \displaystyle\sum_{\mathrm{q}\in\mathcal{Q}(\boldsymbol{s})}\prod_{(s_j,s_k)\in\mathrm{q}} B(s_j,s_k), & \text{if } m \text{ is even}, \end{cases} \tag{7}$$

where $B:\{(\tau_1,\tau_2)\mid \tau_1\le\tau_2\}\to\mathbb{C}$ is the *two-point bath correlation* whose value only relies on the difference of the absolute values of the two variables:

$$B(\tau_1,\tau_2)=B^*(\Delta\tau)=\frac{1}{\pi}\int_0^\infty J(\omega)\left[\coth\left(\frac{\beta\omega}{2}\right)\cos(\omega\Delta\tau)-\mathrm{i}\sin(\omega\Delta\tau)\right]\mathrm{d}\omega \tag{8}$$

with

$$\Delta\tau=|\tau_1|-|\tau_2|.$$

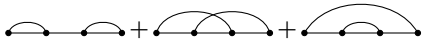
The explicit formula of the single-variable function $B^*(\cdot)$ depends on the real-valued spectral density $J(\omega)$. The set $\mathcal{Q}(\boldsymbol{s})$ is given by:

$$\mathcal{Q}(s_1,\cdots,s_m)=\left\{\{(s_{j_1},s_{k_1}),\cdots,(s_{j_{m/2}},s_{k_{m/2}})\}\,\middle|\,\{j_1,\cdots,j_{m/2},k_1,\cdots,k_{m/2}\}=\{1,\cdots,m\},\quad j_l<k_l \text{ for any } l=1,\cdots,m/2\right\}. \tag{9}$$

To get some intuition behind the definition of the bath influence functional, we consider a simple case $m=4$, where the equation (7) turns out to be

$$\mathcal{L}_b(s_1,s_2,s_3,s_4)=B(s_1,s_2)B(s_3,s_4)+B(s_1,s_3)B(s_2,s_4)+B(s_1,s_4)B(s_2,s_3), \tag{10}$$

which can be graphically represented by the following diagrams:

$\mathcal{L}_b(s_1,s_2,s_3,s_4)=$  . (11)

In the diagrammatic representation above, each diagram refers to a product $B(\cdot,\cdot)B(\cdot,\cdot)$ where each arc connecting a pair of bullets denotes the corresponding two-point correlation. For general m, the value of the corresponding bath influence functional is the sum of all possible combinations of such pairings, and the number of these diagrams is $(m-1)!!$. Since the bath influence functional vanishes when m is odd, the right-hand side of (4) actually only sums over terms with even m.

To evaluate $G(-t,t)$, one may truncate the Dyson series at a sufficiently large even integer $\bar{M}$ and evaluate those high-dimensional integrals on the right-hand side using Monte Carlo integration, resulting in the bare dQMC. More specifically, one can draw $\mathcal{M}$ samples of $\{m^{(i)},\boldsymbol{s}^{(i)}\}$ independently according to a certain distribution $\mathbb{P}(m,\boldsymbol{s})$ for $m=2,\cdots,\bar{M}$ and $\boldsymbol{s}$ satisfying $-t\le s_1\le\cdots\le s_m\le t$. Then $G(-t,t)$ can be approximated by

$$G(-t,t)\approx \mathrm{e}^{\mathrm{i}tH_s}O_s\mathrm{e}^{-\mathrm{i}tH_s}+\frac{1}{\mathcal{M}}\sum_{i=1}^{\mathcal{M}}\frac{1}{\mathbb{P}(m^{(i)},\boldsymbol{s}^{(i)})}\cdot\mathrm{i}^{m^{(i)}}(-1)^{\#\{\boldsymbol{s}^{(i)}<0\}}\mathcal{U}^{(0)}(-t,\boldsymbol{s}^{(i)},t)\mathcal{L}_b(\boldsymbol{s}^{(i)}). \tag{12}$$

The numerical solution obtained via bare dQMC has been proved to have a variance that grows double exponentially with respect to t [6]. Therefore, the number of samples $\mathcal{M}$ should increase with t accordingly to achieve sufficient accuracy at the final time. Hence, the computational cost of the Monte Carlo approximation, especially the expensive evaluation of the bath influence functional $\mathcal{L}_b$, also grows double exponentially with time. To mitigate this problem, in the next section, we will formulate an integro-differential equation which gives the time evolution of $G(-t,t)$. Thus some bath influence functionals obtained when computing $G(-t',t')$ with $t'<t$ can be reused when computing $G(-t,t)$. Before that, however, we first present the following useful properties of the bivariate functions $G_s^{(0)}(\cdot,\cdot)$ and $B(\cdot,\cdot)$ appearing in the definitions of $\mathcal{U}^{(0)}$ and $\mathcal{L}_b$:

Proposition 1.

- *For any $s_\mathrm{i}\le s_\mathrm{f}$, we have*

$$\begin{aligned}&G_s^{(0)}(-s_\mathrm{f},-s_\mathrm{i})=G_s^{(0)}(s_\mathrm{i},s_\mathrm{f})^\dagger \text{ for } s_\mathrm{i}\neq 0 \text{ and } s_\mathrm{f}\neq 0,\\ &B(-s_\mathrm{f},-s_\mathrm{i})=\overline{B(s_\mathrm{i},s_\mathrm{f})}.\end{aligned} \tag{13}$$

- *For any $s_\mathrm{i}\le s_\mathrm{f}$ and $\Delta t\ge 0$, we have*

$$B(s_\mathrm{i},s_\mathrm{f})=\begin{cases}B(s_\mathrm{i}-\Delta t,s_\mathrm{f}-\Delta t), & \text{if } s_\mathrm{i}\le s_\mathrm{f}<0,\\ B(s_\mathrm{i}+\Delta t,s_\mathrm{f}+\Delta t), & \text{if } 0<s_\mathrm{i}\le s_\mathrm{f},\\ B(s_\mathrm{i}-\Delta t,s_\mathrm{f}+\Delta t), & \text{if } s_\mathrm{i}<0\le s_\mathrm{f}.\end{cases} \tag{14}$$

- *For any s_i, s_f and Δt satisfying $s_\mathrm{i}\le s_\mathrm{f}\le 0\le s_\mathrm{i}+\Delta t\le s_\mathrm{f}+\Delta t$, we have*

$$B(s_\mathrm{i}+\Delta t,s_\mathrm{f}+\Delta t)=\overline{B(s_\mathrm{i},s_\mathrm{f})}. \tag{15}$$

(13) can be verified by a case-by-case argument under different settings of s_i and s_f and its detailed proof is placed in Appendix A. (14) and (15) are the results derived by the definition of the two-point correlation (8). We remark that due to the existence of O_s in the definition of $G_s^{(0)}(\cdot,\cdot)$, the first equality in (13) does not hold when s_i or s_f equal to 0.

2.2. Integro-differential equation for the propagator

To derive the integro-differential equation, we begin with calculating the derivative of $G(-t,t)$. By definition (4),

$$\begin{aligned}
&G(-(t+\Delta t),(t+\Delta t))\\
&= \mathrm{e}^{\mathrm{i}(t+\Delta t)H_s} O_s \mathrm{e}^{-\mathrm{i}(t+\Delta t)H_s} + \sum_{\substack{m=2\\ m \text{ is even}}}^{+\infty} \mathrm{i}^m \int_{-t\le \boldsymbol{s}\le t} \mathrm{d}\boldsymbol{s}(-1)^{\#\{\boldsymbol{s}<0\}} \mathcal{U}^{(0)}(-(t+\Delta t),\boldsymbol{s},t+\Delta t)\cdot \mathcal{L}_b(\boldsymbol{s})\\
&\quad - \sum_{\substack{m=2\\ m \text{ is even}}}^{+\infty} \mathrm{i}^m \int_{-(t+\Delta t)}^{-t} \mathrm{d}s_1 \int_{s_1\le s_2\cdots\le s_m\le t} \mathrm{d}s_2\cdots \mathrm{d}s_m (-1)^{\#\{\{s_i\}_{i=2}^m<0\}} \mathcal{U}^{(0)}(-(t+\Delta t),\boldsymbol{s},t+\Delta t)\cdot \mathcal{L}_b(\boldsymbol{s})\\
&\quad + \sum_{\substack{m=2\\ m \text{ is even}}}^{+\infty} \mathrm{i}^m \int_{t}^{t+\Delta t} \mathrm{d}s_m \int_{-(t+\Delta t)\le s_1\le\cdots s_{m-1}\le s_m} \mathrm{d}s_1\cdots \mathrm{d}s_{m-1} (-1)^{\#\{\{s_i\}_{i=1}^{m-1}<0\}} \mathcal{U}^{(0)}(-(t+\Delta t),\boldsymbol{s},t+\Delta t)\cdot \mathcal{L}_b(\boldsymbol{s}).
\end{aligned}$$

Here we split all integrals into three parts based on the distribution of the time sequences. Note that a minus sign is added before the second summation above since s_1 is restricted within $[-(t+\Delta t),-t]$ in this term and thus $(-1)^{\#\{\boldsymbol{s}<0\}} = -(-1)^{\#\{\{s_i\}_{i=2}^m<0\}}$. This expression allows us to differentiate $G(-t,t)$ by the definition of the derivative:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}G(-t,t) &= \lim_{\Delta t\to 0}\frac{G(-(t+\Delta t),(t+\Delta t)) - G(-t,t)}{\Delta t}\\
&= \frac{\mathrm{d}}{\mathrm{d}t}\left(\mathrm{e}^{\mathrm{i}tH_s} O_s \mathrm{e}^{-\mathrm{i}tH_s}\right) + \sum_{\substack{m=2\\ m \text{ is even}}}^{+\infty} \mathrm{i}^m \int_{-t\le \boldsymbol{s}\le t} \mathrm{d}\boldsymbol{s}(-1)^{\#\{\boldsymbol{s}<0\}} \frac{\mathrm{d}}{\mathrm{d}t}\mathcal{U}^{(0)}(-t,\boldsymbol{s},t)\cdot \mathcal{L}_b(\boldsymbol{s})\\
&\quad - \sum_{\substack{m=2\\ m \text{ is even}}}^{+\infty} \mathrm{i}^m \int_{-t\le s_2\le\cdots\le s_m\le t} \mathrm{d}s_2\cdots \mathrm{d}s_m (-1)^{\#\{\{s_i\}_{i=2}^m<0\}} \mathcal{U}^{(0)}(-t,-t,\underline{s_2,\cdots,s_m},t)\cdot \mathcal{L}_b(-t,\underline{s_2,\cdots,s_m})\\
&\quad + \sum_{\substack{m=2\\ m \text{ is even}}}^{+\infty} \mathrm{i}^m \int_{-t\le s_1\le\cdots\le s_{m-1}\le t} \mathrm{d}s_1\cdots \mathrm{d}s_{m-1} (-1)^{\#\{\{s_i\}_{i=1}^{m-1}<0\}} \mathcal{U}^{(0)}(-t,\underline{s_1,\cdots,s_{m-1}},t,t)\cdot \mathcal{L}_b(\underline{s_1,\cdots,s_{m-1}},t).
\end{aligned} \tag{16}$$

Using the definition (5) of $\mathcal{U}^{(0)}$, the derivative in the first series is computed by

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{U}^{(0)}(-t,\boldsymbol{s},t) &= \frac{\mathrm{d}}{\mathrm{d}t}\left(G_s^{(0)}(s_m,t)\right) W_s G_s^{(0)}(s_{m-1},s_m) W_s \cdots W_s G_s^{(0)}(s_1,s_2) W_s G_s^{(0)}(-t,s_1)\\
&\quad + G_s^{(0)}(s_m,t) W_s G_s^{(0)}(s_{m-1},s_m) W_s \cdots W_s G_s^{(0)}(s_1,s_2) W_s \frac{\mathrm{d}}{\mathrm{d}t}\left(G_s^{(0)}(-t,s_1)\right)\\
&= \mathrm{i}H_s \mathcal{U}^{(0)}(-t,\boldsymbol{s},t) - \mathrm{i}\mathcal{U}^{(0)}(-t,\boldsymbol{s},t)H_s.
\end{aligned}$$

Note that $\frac{\mathrm{d}}{\mathrm{d}t}\left(\mathrm{e}^{\mathrm{i}tH_s} O_s \mathrm{e}^{-\mathrm{i}tH_s}\right) = \mathrm{i}H_s \mathrm{e}^{\mathrm{i}tH_s} O_s \mathrm{e}^{-\mathrm{i}tH_s} - \mathrm{i}\mathrm{e}^{\mathrm{i}tH_s} O_s \mathrm{e}^{-\mathrm{i}tH_s} H_s$, which yields

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\mathrm{e}^{\mathrm{i}tH_s} O_s \mathrm{e}^{-\mathrm{i}tH_s}\right) + \sum_{\substack{m=2\\ m \text{ is even}}}^{+\infty} \mathrm{i}^m \int_{-t\le \boldsymbol{s}\le t} \mathrm{d}\boldsymbol{s}(-1)^{\#\{\boldsymbol{s}<0\}} \frac{\mathrm{d}}{\mathrm{d}t}\mathcal{U}^{(0)}(-t,\boldsymbol{s},t)\cdot \mathcal{L}_b(\boldsymbol{s}) = \mathrm{i}[H_s, G(-t,t)]. \tag{17}$$

As for the other two series on the right-hand side of (16), we can simplify them by using

$$\mathcal{U}^{(0)}(-t,-t,\boldsymbol{s},t) = W_s \mathcal{U}^{(0)}(-t,\boldsymbol{s},t), \qquad \mathcal{U}^{(0)}(-t,\boldsymbol{s},t,t) = \mathcal{U}^{(0)}(-t,\boldsymbol{s},t) W_s.$$

Summarizing all the simplifications of (16), we obtain

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}G(-t,t) = \mathrm{i}[H_s, G(-t,t)] + \sum_{\substack{m=1\\ m \text{ is odd}}}^{+\infty} \mathrm{i}^{m+1}\Big(&\int_{-t\le \boldsymbol{s}\le t} \mathrm{d}\boldsymbol{s}(-1)^{\#\{\boldsymbol{s}<0\}} W_s \mathcal{U}^{(0)}(-t,\boldsymbol{s},t)\mathcal{L}_b(\boldsymbol{s},t)\\
&- \int_{-t\le \boldsymbol{s}\le t} \mathrm{d}\boldsymbol{s}(-1)^{\#\{\boldsymbol{s}<0\}} \mathcal{U}^{(0)}(-t,\boldsymbol{s},t) W_s \mathcal{L}_b(-t,\boldsymbol{s})\Big).
\end{aligned} \tag{18}$$

Note that m takes odd values in (18) since the underlined time sequences in (16) have odd numbers of components. The equation (18) has already provided us an integro-differential equation to work on. However, we may make further simplification by combining the two integrals into one using the following lemma:

Lemma 1. *For any time sequence $-t < s_1 < \cdots < s_m < t$, define $s'_i = -s_{m+1-i}$ for $i = 1, \cdots, m$. Then $-t < s'_1 < \cdots < s'_m < t$ and*

$$\mathcal{U}^{(0)}(-t, \boldsymbol{s}', t) = \mathcal{U}^{(0)}(-t, \boldsymbol{s}, t)^\dagger \text{ for } s_i \neq 0,\ i = 1, \cdots, m, \tag{19}$$

$$\mathcal{L}_b(-t, \boldsymbol{s}') = \overline{\mathcal{L}_b(\boldsymbol{s}, t)}. \tag{20}$$

The statement (19) for the system associated $\mathcal{U}^{(0)}$ can be checked by

$$\begin{aligned}
\mathcal{U}^{(0)}(-t, \boldsymbol{s}', t) &= G_s^{(0)}(s'_m, t) W_s G_s^{(0)}(s'_{m-1}, s'_m) W_s \cdots W_s G_s^{(0)}(s'_1, s'_2) W_s G_s^{(0)}(-t, s'_1) \\
&= G_s^{(0)}(-s_1, t) W_s G_s^{(0)}(-s_2, -s_1) W_s \cdots W_s G_s^{(0)}(-s_m, -s'_{m-1}) W_s G_s^{(0)}(-t, -s_m) \\
&= G_s^{(0)}(-t, s_1)^\dagger W_s G_s^{(0)}(s_1, s_2)^\dagger W_s \cdots W_s G_s^{(0)}(s_{m-1}, s_m)^\dagger W_s G_s^{(0)}(s_m, t)^\dagger \\
&= \mathcal{U}^{(0)}(-t, \boldsymbol{s}, t)^\dagger
\end{aligned}$$

using (13). The equation (20) can also be verified using (13). The rigorous proof can be found in Appendix A.

Now we apply the change of variables as shown in Lemma 1 to the second integral in (18). Note that (19) holds almost everywhere in the domain of integration. We then have

$$\begin{aligned}
\int_{-t \le \boldsymbol{s} \le t} \mathrm{d}\boldsymbol{s} (-1)^{\#\{\boldsymbol{s}<0\}} \mathcal{U}^{(0)}(-t, \boldsymbol{s}, t) W_s \mathcal{L}_b(-t, \boldsymbol{s}) &= \int_{-t \le \boldsymbol{s}' \le t} \mathrm{d}\boldsymbol{s}' (-1)^{\#\{\boldsymbol{s}'>0\}} \mathcal{U}^{(0)}(-t, \boldsymbol{s}', t)^\dagger W_s \overline{\mathcal{L}_b(\boldsymbol{s}', t)} \\
&= -\int_{-t \le \boldsymbol{s}' \le t} \mathrm{d}\boldsymbol{s}' (-1)^{\#\{\boldsymbol{s}'<0\}} \left(W_s \mathcal{U}^{(0)}(-t, \boldsymbol{s}', t) \mathcal{L}_b(\boldsymbol{s}', t) \right)^\dagger.
\end{aligned} \tag{21}$$

In the last equality above, we have used the fact that $\boldsymbol{s}' = (s'_1, \cdots, s'_m)$ has odd number of components and thus $(-1)^{\#\{\boldsymbol{s}'>0\}} = -(-1)^{\#\{\boldsymbol{s}'<0\}}$ for almost every $\boldsymbol{s}'$. Inserting (21) back to (18), we reach a simpler integral-differential equation for $G(-t, t)$:

Proposition 2. *The propagator $G(-t, t)$ satisfies the integro-differential equation*

$$\frac{\mathrm{d}}{\mathrm{d}t} G(-t, t) = \mathrm{i}[H_s, G(-t, t)] + \sum_{\substack{m=1 \\ m \text{ is odd}}}^{+\infty} \mathrm{i}^{m+1} \int_{-t \le \boldsymbol{s} \le t} \mathrm{d}\boldsymbol{s} (-1)^{\#\{\boldsymbol{s}<0\}} (\mathcal{K}(\boldsymbol{s}, t) + \mathcal{K}(\boldsymbol{s}, t)^\dagger) \tag{22}$$

for $t > 0$, where

$$\mathcal{K}(\boldsymbol{s}, t) = W_s \mathcal{U}^{(0)}(-t, \boldsymbol{s}, t) \mathcal{L}_b(\boldsymbol{s}, t).$$

Based on the evolution equation (22), one can consider solving $G(-t, t)$ iteratively using Runge-Kutta type methods. To avoid large values of m in the computation, we truncate the series up to a certain odd integer, and evaluate the high-dimensional integrals stochastically via Monte Carlo approximation. Compared to the original bare dQMC for the Dyson series (12), solving (22) should be more efficient as m decreases by 1 for each term of the summation. We also point out that the numerical methods based on the integro-differential equation preserve the Hermitian property (3) of $G(-t, t)$ as one can easily check that the right-hand side of (22) is always Hermitian under Monte Carlo approximation, while this is not guaranteed by bare dQMC (12) and may be badly violated when the number of samples $\mathcal{M}$ is insufficient. Moreover, since the equation provides us the time evolution of $G(-t, t)$, we are now able to reuse the calculated bath influence functionals which will give the major improvement on the efficiency of the algorithm. Our numerical method will be detailed in the following section.

2.3. Numerical method

To discretize (22), we consider a numerical scheme inspired by the second-order Heun's method. For a general ordinary differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t} u(t) = f(t, u(t)), \quad t \in [0, t_{\max}],$$

the scheme reads

$$\begin{aligned}
U_i^* &= U_{i-1} + h f(t_{i-1}, U_{i-1}), \\
U_i &= \frac{1}{2}(U_{i-1} + U_i^*) + \frac{1}{2} h f(t_i, U_i^*),
\end{aligned} \tag{23}$$

where h is the time step length, $t_i = i \cdot h$, and U_i is the numerical approximation of $u(t_i)$. For our integro-differential equation, the sums over high-dimensional integrals should be evaluated in the same way as the bare dQMC (12) using Monte Carlo approximation. In the ith time step, suppose we have $\mathcal{M}_i$ samples of time sequences $\mathcal{S}_i = \{\boldsymbol{s}_i^{(j)}\}_{j=1}^{\mathcal{M}_i}$ drawn from the domain

$$T_i = \bigcup_{\substack{m=1 \\ m \text{ is odd}}}^{\bar{M}} T_i^{(m)}, \tag{24}$$

where $T_i^{(m)}$ is the m-dimensional simplex defined by

$$T_i^{(m)} := \{\boldsymbol{s} = (s_1, \cdots, s_m) \mid -t_i \le s_1 \le \cdots \le s_m \le t_i\}, \tag{25}$$

and each sampled time sequence satisfies the probability density function $\mathbb{P}_i(m, \boldsymbol{s})$ for $m = 1, 3, \cdots, \bar{M}$. Thereby, the scheme coupling Heun's method with Monte Carlo integration to approximate $G(-t_i, t_i)$ is formulated as

$$\begin{aligned} G_i^* &= G_{i-1} + h\mathrm{i}[H_s, G_{i-1}] + \frac{h}{\mathcal{M}_{i-1}} \sum_{j=1}^{\mathcal{M}_{i-1}} \frac{1}{\mathbb{P}_{i-1}(m_{i-1}^{(j)}, \boldsymbol{s}_{i-1}^{(j)})} \cdot \mathrm{i}^{m_{i-1}^{(j)}+1} \cdot (-1)^{\#\{\boldsymbol{s}_{i-1}^{(j)} < 0\}} (\mathcal{K}(\boldsymbol{s}_{i-1}^{(j)}, t_{i-1}) + \mathcal{K}(\boldsymbol{s}_{i-1}^{(j)}, t_{i-1})^\dagger), \\ G_i &= \frac{1}{2}(G_{i-1} + G_i^*) + \frac{1}{2}h\mathrm{i}[H_s, G_i^*] + \frac{h}{2\mathcal{M}_i} \sum_{j=1}^{\mathcal{M}_i} \frac{1}{\mathbb{P}_i(m_i^{(j)}, \boldsymbol{s}_i^{(j)})} \cdot \mathrm{i}^{m_i^{(j)}+1} \cdot (-1)^{\#\{\boldsymbol{s}_i^{(j)} < 0\}} (\mathcal{K}(\boldsymbol{s}_i^{(j)}, t_i) + \mathcal{K}(\boldsymbol{s}_i^{(j)}, t_i)^\dagger), \text{ for } \boldsymbol{s}_i^{(j)} \in \mathcal{S}_i \end{aligned} \tag{26}$$

for $i = 1, 2, \cdots, N$ where $Nh = t_{\max}$ with initial condition $G_0 = O_s$. The set of samples $\mathcal{S}_i$ are drawn independently according to the distribution $\mathbb{P}_i$. We remark that one can apply higher order schemes to achieve better order of accuracy with respect to step length h. Throughout the current work, however, we use the Heun's method which can already provide satisfactory numerical results. The accuracy of discretization will be verified by numerical tests later in Section 4.2. We also refer readers to the numerical experiments in [7, Section 7], where Heun's method is applied to a number of spin-boson simulations and shows good performance.

The major computational cost lies in the evaluation of $\mathcal{K}(\boldsymbol{s}, t_i)$ in each time step. While evaluating each $\mathcal{K}(\boldsymbol{s}, t_i)$, the bath influence functional $\mathcal{L}_b(\boldsymbol{s}, t_i)$ is generally much more expensive than the $\mathcal{U}^{(0)}(-t_i, \boldsymbol{s}, t_i)$, especially when m is large. In fact, the computational cost of $\mathcal{L}_b$, which is essentially the hafnian of a matrix [1], grows at least exponentially with respect to m using some recent indirect methods such as Björklund's algorithm [3] or the inclusion-exclusion principle [50], while the cost of $\mathcal{U}^{(0)}$ grows only linearly with m since $\mathcal{U}^{(0)}$ is a product of $2m+1$ matrices as defined in (5). A comparison of the computational time for these two parts will be performed later in Section 4.3.

Due to the high computational cost of $\mathcal{L}_b$, the purpose of this paper is to reduce the number of bath influence functionals to be computed during the evolution of $G(-t, t)$. While the straightforward application of the numerical scheme (26) requires computation of different bath influence functionals in different time steps, by the invariance of the two-point bath correlation given in Propositions (13) to (15), we can actually reuse some bath influence functionals that have been calculated in previous time steps to improve the overall efficiency. This idea utilizes the following property of $\mathcal{L}_b$, which can be easily derived from (14):

Proposition 3. *Given $\boldsymbol{s} = (s_1, \cdots, s_m) \in T_i^{(m)}$ for $i = 1, \cdots, N-1$ and odd number $m = 1, 3, \cdots, \bar{M}$, define the operator $\mathcal{I}_j(\boldsymbol{s}) = (\tilde{s}_1, \cdots, \tilde{s}_m)$ such that*

$$\tilde{s}_k = \begin{cases} s_k + jh, & \text{if } s_k \ge 0, \\ s_k - jh, & \text{if } s_k < 0 \end{cases} \tag{27}$$

for $k = 1, \cdots, m$ and $j = 0, 1, \cdots, N-i$. We have $\mathcal{I}_j(\boldsymbol{s}) \in T_{i+j}^{(m)}$ and

$$\mathcal{L}_b(\mathcal{I}_j(\boldsymbol{s}), t_{i+j}) = \mathcal{L}_b(\boldsymbol{s}, t_i).$$

This proposition shows that a class of bath influence functionals has the same value, and thus we just need to compute one of them if multiple influence functionals appear in our computation. To illustrate how such reuse can be applied to the scheme (26), we consider the following simple example, where we only sample one time sequence with $m = 1$ (so the sequence actually reduces to a point) in each time step and consider the time evolution of the scheme up to $t = 3h$:

(**i**) in the first time step, we pick a sample $s_1 \in (-h, h)$. Here we assume s_1 is negative which can be denoted by the black dot in the top panel of Fig. 2. The corresponding bath influence functional $\mathcal{L}_b(s_1, h) = B(s_1, h)$ is then calculated and can be denoted by blue arc;

(**ii**) in the second time step, $\mathcal{I}_1(s_1) = s_1 - h$ is a sample in $T_2^{(1)}$ whose bath influence functional can be directly obtained from $\mathcal{L}_b(\mathcal{I}_1(s_1), 2h) = \mathcal{L}_b(s_1, h)$ according to Proposition 3. Such reuse of computed bath influence functionals can be visualized as a stretch of the blue arc by length h in Fig. 2, and the value of the blue arc is invariant after being stretched. In addition to reuse of calculations, we sample a new time point $s_2 \in (-2h, 2h)$ and calculate $\mathcal{L}_b(s_2, 2h)$. In Fig. 2, we assume s_2 is positive and $\mathcal{L}_b(s_2, 2h)$ is represented by the red arc;

(**iii**) at $t = 3h$, the blue arc can be further stretched by another time step h and the value remains the same, meaning that we again obtain the bath influence functional directly using $\mathcal{L}_b(\mathcal{I}_2(s_1), 3h) = \mathcal{L}_b(s_1, h)$ where $\mathcal{I}_2(s_1) = s_1 - 2h \in T_3^{(1)}$. Similarly, we can also reuse $\mathcal{L}_b(\mathcal{I}_1(s_2), 3h) = \mathcal{L}_b(s_2, 2h)$ with $\mathcal{I}_1(s_2) = s_2 + h \in T_3^{(1)}$, which corresponds to shifting the red arc to the right by h. Afterwards, we draw another new sample $s_3 \in (-3h, 3h)$ and calculate $\mathcal{L}_b(s_3, 3h)$ denoted by the green arc.

For general m, this reuse of bath influence functionals can be similarly understood by replacing the arcs by the summation of diagrams such as in (10). We remark that such invariance does not hold for the system functional $\mathcal{U}^{(0)}$, which does not have a similar property as Proposition 3 due to the existence of O_s in its definition.

As can be observed from Fig. 2, given any time sequence $\boldsymbol{s}_j$ at the jth time step for $j < i$, shifting or stretching it to $\mathcal{I}_{i-j}(\boldsymbol{s}_j)$ always moves the nodes away from $t = 0$ by at least length h. This means all the samples obtained by stretching or shifting have no time points falling between $-h$ and h. As a result, the samples for the ith time step cannot be only inherited from previous time steps. To complete the sampling of T_i, we also need to draw extra samples from $\hat{T}_i = \bigcup_{\substack{m=1 \\ m \text{ is odd}}}^{\bar{M}} \hat{T}_i^{(m)}$ where

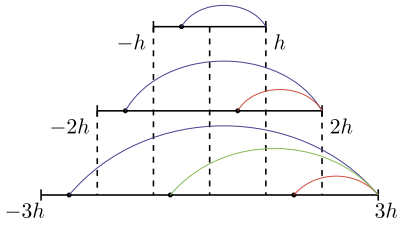


Fig. 2. Calculation reuse of $\mathcal{L}_b(\boldsymbol{s}_i, t_i)$ for $m = 1$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$\hat{T}_i^{(m)} = \left\{(s_1, \cdots, s_m) \in T_i^{(m)} \;\middle|\; \exists\, s_j \text{ such that } -h < s_j < h\right\}. \tag{28}$$

For example, in Fig. 2, the nodes building up the red diagrams in $(-2h, 2h)$ and green diagrams $(-3h, 3h)$ should be newly drawn from $\hat{T}_2$ and $\hat{T}_3$ respectively since these diagrams can never be obtained from shifting or stretching diagrams at previous time steps. Based on the definition (28), we may express $T_i^{(m)}$ as

$$T_i^{(m)} = \bigcup_{j=1}^{i} \mathcal{I}_{i-j}(\hat{T}_j^{(m)})$$

where $\mathcal{I}_{i-j}(\hat{T}_j^{(m)})$ is the collection of time sequences which are shifted or stretched from jth step:

$$\mathcal{I}_{i-j}(\hat{T}_j^{(m)}) = \{\mathcal{I}_{i-j}(\boldsymbol{s}) \mid \boldsymbol{s} \in \hat{T}_j^{(m)}\}.$$

One may easily see that $\mathcal{I}_{i-j}(\hat{T}_j^{(m)})$ are pairwise disjoint for $j = 1, \cdots, i$ and thus

$$\sum_{j=1}^{i} |\hat{T}_j^{(m)}| = \sum_{j=1}^{i} |\mathcal{I}_{i-j}(\hat{T}_j^{(m)})| = |T_i^{(m)}|. \tag{29}$$

Hence the volume of each $\hat{T}_j^{(m)}$ can be calculated by

$$|\hat{T}_j^{(m)}| = |T_j^{(m)}| - |T_{j-1}^{(m)}| = \frac{1}{m!}[(2t_j)^m - (2t_{j-1})^m]. \tag{30}$$

To implement the numerical scheme (26), we sample time sequences $\hat{\mathcal{S}}_i \subset \hat{T}_i$ in each step and evaluate the corresponding bath influence functionals. Afterwards, we construct $\mathcal{S}_i$ by combining the new samples $\hat{\mathcal{S}}_i$ with the old samples $\mathcal{I}_{i-j}(\hat{\mathcal{S}}_j)$ for $j = 1, \cdots, i-1$ whose bath influence functionals can be directly reused by Proposition 3, and then evaluate G_i according to (26). Such a procedure is described by the Algorithm 1.

Algorithm 1 Dyson series

input $\hat{\mathcal{S}}_i = \{\boldsymbol{s}_i^{(j)}\}_{j=1}^{\hat{\mathcal{M}}_i} \subset \hat{T}_i$ for $i = 1, \cdots, N$
Set $G_0 \leftarrow \mathrm{Id}$
for i from 1 to N **do**
  Compute $\hat{L}_i = \{\mathcal{L}_b(\boldsymbol{s}_i^{(j)}, t_i)\}_{j=1}^{\hat{\mathcal{M}}_i}$
  Set $\mathcal{S}_i \leftarrow \bigcup_{j=1}^{i} \mathcal{I}_{i-j}(\hat{\mathcal{S}}_j)$ ▷ *Shift/stretch samples*
  Set $L_i \leftarrow \bigcup_{j=1}^{i} \hat{L}_j$ ▷ *Reuse bath calculation*
  Compute G_i by scheme (26) based on $\mathcal{S}_i$ and L_i
end for
return G_i for $i = 1, \cdots, N$

To complete the implementation, we need to specify the sampling strategy for the input $\hat{\mathcal{S}}_i$, which is associated with the probability density function $\mathbb{P}_i(m, \boldsymbol{s})$ in (26). Ideally, the number of samples in $\hat{\mathcal{S}}_i$ should be proportional to the integral of the absolute value of the bath influence functional:

$$\hat{\mathcal{M}}_i^{(m)} \propto \int_{\boldsymbol{s} \in \hat{T}_i^{(m)}} \mathrm{d}\boldsymbol{s} |\mathcal{L}_b(\boldsymbol{s}, t_i)| = \int_{\boldsymbol{s} \in \hat{T}_i^{(m)}} \mathrm{d}\boldsymbol{s} \left| \sum_{\mathfrak{q} \in \mathcal{Q}(\boldsymbol{s}, t_i)} \prod_{(s_j, s_k) \in \mathfrak{q}} B(s_j, s_k) \right|. \tag{31}$$

In practice, as the integral is difficult to evaluate, we replace $B(s_j, s_k)$ by an empirical constant $\mathcal{B} \in (0, \max|B|)$, so that

$$\hat{\mathcal{M}}_i^{(m)} = \frac{\hat{\mathcal{M}}_1^{(1)}}{\hat{\lambda}} \cdot |\hat{T}_i^{(m)}| \cdot m!! \mathcal{B}^{\frac{m+1}{2}} = \frac{\hat{\mathcal{M}}_1^{(1)}}{\hat{\lambda}} \cdot \frac{(2t_i)^m - (2t_{i-1})^m}{(m-1)!!} \cdot \mathcal{B}^{\frac{m+1}{2}} \tag{32}$$

where $\hat{\lambda} = 2\mathcal{B}h$ is the normalizing factor. In the numerical implementation, one may first assign $\hat{\mathcal{M}}_1^{(1)} = \hat{\mathcal{M}}_0$, and the other $\hat{\mathcal{M}}_i^{(m)}$ can then be set as the nearest integer to the right-hand side of the formula above. Afterwards, we generate each time sequence $\boldsymbol{s} = (s_1, \cdots, s_m) \in \hat{T}_i^{(m)}$ by drawing a sample from the uniform distribution $U(\hat{T}_i^{(m)})$. The following theorem provides the explicit expression for the probability density $\mathbb{P}_i(m, \boldsymbol{s})$ appearing in scheme (26):

Proposition 4. *For any $i = 1, 2, \cdots, N$ and $m = 1, 3, \cdots, \bar{M}$, $\mathbb{P}_i(m, \boldsymbol{s})$ is given by*

$$\mathbb{P}_i(m, \boldsymbol{s}) = \frac{1}{\lambda_i} \cdot m!! \mathcal{B}^{\frac{m+1}{2}} \tag{33}$$

where

$$\lambda_i = \sum_{\substack{m'=1 \\ m' \text{ is odd}}}^{\bar{M}} \frac{(2t_i)^{m'}}{(m'-1)!!} \cdot \mathcal{B}^{\frac{m'+1}{2}}$$

Proof. For any time sequence $\boldsymbol{s}$ which is obtained by either reuse or newly sampling in each step, we have $\mathbb{P}(\boldsymbol{s} \in T_i^{(m)}) \propto \mathcal{M}_i^{(m)}$ where $\mathcal{M}_i^{(m)}$ is the number of time sequences with m components in ith step. According to our sampling strategy,

$$\mathcal{M}_i^{(m)} = \sum_{j=1}^{i} \hat{\mathcal{M}}_j^{(m)} = \frac{\hat{\mathcal{M}}_0}{\hat{\lambda}} \cdot m!! \mathcal{B}^{\frac{m+1}{2}} \cdot \sum_{j=1}^{i} |\hat{T}_j^{(m)}| = \frac{\hat{\mathcal{M}}_0}{\hat{\lambda}} \cdot m!! \mathcal{B}^{\frac{m+1}{2}} \cdot |T_i^{(m)}|$$

where we have used the relation (29) for the last inequality. Note that the time sequences $\mathcal{S}_i$ used in scheme (26) are constructed the samples drawn from the pairwise disjoint $U[\mathcal{I}_{i-j}(\hat{T}_j^{(m)})]$, and the number of these samples locating in each $\mathcal{I}_{i-j}(\hat{T}_j^{(m)})$ is proportional to the volume $|\mathcal{I}_{i-j}(\hat{T}_j^{(m)})|$ according to (32). Therefore, any time sequence in $\mathcal{S}_i$ can be considered as a sample drawn in $U[T_i^{(m)}]$ and thus we reach the conclusion (33) by

$$\begin{aligned}
\mathbb{P}_i(m, \boldsymbol{s}) &= \mathbb{P}(\boldsymbol{s} \in T_i^{(m)} | \boldsymbol{s} \in T_i) \cdot \frac{1}{|T_i^{(m)}|} \\
&= \frac{\mathbb{P}(\boldsymbol{s} \in T_i^{(m)})}{\sum_{\substack{m'=1 \\ m' \text{ is odd}}}^{\bar{M}} \mathbb{P}(\boldsymbol{s} \in T_i^{(m')})} \cdot \frac{1}{|T_i^{(m)}|} = \frac{1}{\sum_{\substack{m'=1 \\ m' \text{ is odd}}}^{\bar{M}} |T_i^{(m')}| \cdot m'!! \mathcal{B}^{\frac{m'+1}{2}}} \cdot m!! \mathcal{B}^{\frac{m+1}{2}}. \quad \square
\end{aligned}$$

2.4. Implementation of Algorithm 1 with low memory cost

In general, the reuse of bath calculations described in Algorithm 1 requires storing of all time sequences (Line 5) as well as bath influence functionals (Line 6) in a simulation, which will lead to a high memory cost when the number of samples is large. However for Dyson series, the linearity of its governing equation (22) allows us to implement the reuse algorithm at a much lower memory cost. To begin with, we apply the scheme (26) recursively and get the following explicit formula for any G_i:

$$G_i = \tilde{\alpha}^i O_s + \frac{1}{2} h \sum_{k=1}^{i-1} \tilde{\alpha}^{i-k-1} (\alpha + \tilde{\alpha})(\beta_k + \beta_k^\dagger) + \frac{1}{2} h (\beta_i + \beta_i^\dagger) \tag{34}$$

where the operator $\alpha = 1 + \mathrm{i}h[H_s, \cdot]$ and $\tilde{\alpha} = \frac{1}{2}(1 + \alpha^2)$. β_i is the average of Monte Carlo samples $\boldsymbol{s} = (s_1, \cdots, s_m)$

$$\beta_i = \frac{1}{\mathcal{M}_i} \sum_{j=1}^{\mathcal{M}_i} \gamma_i(\boldsymbol{s}_i^{(j)}) \cdot \mathcal{U}^{(0)}(-t_i, \boldsymbol{s}_i^{(j)}, t_i) \mathcal{L}_b(\boldsymbol{s}_i^{(j)}, t_i) \quad \text{with} \quad \gamma_i(\boldsymbol{s}) = \frac{1}{\mathbb{P}_i(m, \boldsymbol{s})} \cdot \mathrm{i}^{m+1} \cdot (-1)^{\#\{\boldsymbol{s}<0\}}.$$

Note that the direct evaluation of G_i by (34) requires the storage of all reusable $\mathcal{L}_b$. To avoid this, we consider the following resummation of β_i according to where the samples are originally generated:

$$\beta_i = \theta_{i1} + \theta_{i2} + \cdots \theta_{ii} \tag{35}$$

where the partial sum

$$\theta_{ik} := \frac{1}{\mathcal{M}_i} \sum_{j=1}^{\hat{\mathcal{M}}_k} \gamma_i \left(\mathcal{I}_{i-k}(\hat{\boldsymbol{s}}_k^{(j)}) \right) \cdot \mathcal{U}^{(0)}(-t_i, \mathcal{I}_{i-k}(\hat{\boldsymbol{s}}_k^{(j)}), t_i) \mathcal{L}_b(\mathcal{I}_{i-k}(\hat{\boldsymbol{s}}_k^{(j)}), t_i)$$

stands for the part of calculations where the samples are shifted or stretched from kth step. Note that any kth column of

$$\begin{aligned}
&\theta_{11}, \\
&\theta_{21}, \theta_{22}, \\
&\cdots, \cdots, \cdots, \\
&\theta_{i1}, \theta_{i2}, \cdots, \cdots, \theta_{ii}, \\
&\cdots, \cdots, \cdots, \cdots, \cdots, \\
&\theta_{N1}, \theta_{N2}, \cdots, \cdots, \cdots, \theta_{NN},
\end{aligned} \tag{36}$$

shares the same bath influence functionals with the value $\{\mathcal{L}_b(\hat{\boldsymbol{s}}_k^{(j)}, t_k)\}_{j=1}^{\hat{\mathcal{M}}_k}$ according to the Proposition 3. Therefore, once we have computed one $\mathcal{L}_b(\hat{\boldsymbol{s}}_k^{(j)}, t_k)$, it is added to all θ_{ik} for $i = k, \cdots, N$ and then can be discarded and thus we need to only store one single bath influence functional to obtain all $\theta_{kk'}$ for $1 \le k' \le k \le N$. In the end, the total memory cost for computing G_i for $i = 1, \cdots, N$ will only be the storage of these $\theta_{kk'}$, which are essentially $(N+1)N/2$ two-by-two matrices.

2.5. Analysis on computational cost

To conclude the discussion on the summation of Dyson series, we examine the computational cost that is saved by reusing the bath influence functionals. Specifically, we consider the ratio $1 - \#\{\hat{\boldsymbol{s}}^{(m)}\}/\#\{\boldsymbol{s}^{(m)}\}$ for various m where

$$\#\{\hat{\boldsymbol{s}}^{(m)}\} = \hat{\mathcal{M}}_1^{(m)} + \hat{\mathcal{M}}_2^{(m)} + \cdots + \hat{\mathcal{M}}_N^{(m)},$$
$$\#\{\boldsymbol{s}^{(m)}\} = \mathcal{M}_1^{(m)} + \mathcal{M}_2^{(m)} + \cdots + \mathcal{M}_N^{(m)}.$$

Here $\#\{\hat{\boldsymbol{s}}^{(m)}\}$ denotes the number of $(m+1)$-point bath influence functionals that one needs to evaluate up to Nth time step in our algorithm, and $\#\{\boldsymbol{s}^{(m)}\}$ denotes the corresponding number if all bath influence functionals are to be calculated. For example in Fig. 2, we have $\#\{\hat{\boldsymbol{s}}^{(1)}\} = 3$ as the blue and red diagrams need to be computed only once. However, without reusing the existing information, one then has to draw all the time sequences independently and compute the corresponding $\#\{\boldsymbol{s}^{(1)}\} = 6$ (3 blue arcs, 2 red arcs and 1 green arc) bath influence functionals. Therefore, we have achieved a 50% reduction of the computational cost in this example. In general, by (32) we have

$$\begin{aligned}\#\{\hat{\boldsymbol{s}}^{(m)}\} &= \frac{\hat{\mathcal{M}}_0}{\hat{\lambda}} \cdot \left(|\hat{T}_1^{(m)}| + |\hat{T}_2^{(m)}| + \cdots + |\hat{T}_N^{(m)}|\right) \cdot m!! \mathcal{B}^{\frac{m+1}{2}} \\ &= \frac{\hat{\mathcal{M}}_0}{\hat{\lambda}} \cdot |T_N^{(m)}| \cdot m!! \mathcal{B}^{\frac{m+1}{2}} = \frac{\hat{\mathcal{M}}_0}{\hat{\lambda}} \cdot \frac{\sqrt{\mathcal{B}}(2\sqrt{\mathcal{B}} t_N)^m}{(m-1)!!}\end{aligned}$$

and

$$\begin{aligned}\#\{\boldsymbol{s}^{(m)}\} &= \frac{\hat{\mathcal{M}}_0}{\hat{\lambda}} \cdot \left(N|\hat{T}_1^{(m)}| + (N-1)|\hat{T}_2^{(m)}| + \cdots + |\hat{T}_N^{(m)}|\right) \cdot m!! \mathcal{B}^{\frac{m+1}{2}} \\ &= \frac{\hat{\mathcal{M}}_0}{\hat{\lambda}} \cdot \sum_{i=1}^{N} |T_i^{(m)}| \cdot m!! \mathcal{B}^{\frac{m+1}{2}} = \frac{\hat{\mathcal{M}}_0}{\hat{\lambda}} \cdot \sum_{i=1}^{N} \frac{\sqrt{\mathcal{B}}(2\sqrt{\mathcal{B}} t_i)^m}{(m-1)!!}.\end{aligned} \tag{37}$$

Hence, for a given m, the percentage of the computational cost that one can save is given by

$$1 - \frac{\#\{\hat{\boldsymbol{s}}^{(m)}\}}{\#\{\boldsymbol{s}^{(m)}\}} = 1 - \frac{N^m}{1^m + 2^m + \cdots + N^m} =: R^{(m)}(N) \tag{38}$$

which only relies on the number of time steps N.

Below we plot the graphs of $R^{(m)}$ (dashed lines) for various m up to $t = 5$ with the time step length $h = 0.05$ in Fig. 3, which are all monotonically increasing and thus one may benefit a higher reduction of computational cost from the bath calculation reuse for longer time simulations. In addition, the curves become lower as m grows, indicating that the bath influence functionals with smaller m are reused more frequently for fixed N. This observation can be diagrammatically understood in Fig. 2. In general, a time sequence $\boldsymbol{s} = (s_1, \cdots, s_m)$ with larger m is more likely to have one of its components falling in $(-h, h)$, so that its bath influence functional has to be newly evaluated. However, the value of m usually does not go too large for the purpose of computing Dyson series. When $m = \bar{M} = 25$, one can still expect an around 77% reduction in bath computations at $t = 5$. As time further evolves, we may apply the Faulhaber's formula [25]

$$1^m + 2^m + \cdots + N^m \sim \frac{N^{m+1}}{m+1} \text{ as } N \to +\infty \tag{39}$$

to get the asymptotic behavior of $R^{(m)}$:

$$R^{(m)}(N) \sim 1 - \frac{m+1}{N}. \tag{40}$$

Below we will take into account all choices of m and estimate the overall reduction of the computational cost. Let $\mathcal{T}^{(m)}$ denote the average wall clock time for the evaluation of $\mathcal{L}_b(s_1, \cdots, s_m, t)$, the overall savings of the computational time spent on bath computations is then estimated as

$$R_{\mathrm{T}}(N, h, \mathcal{B}) = 1 - \frac{\sum_{\substack{m=1 \\ m \text{ is odd}}}^{\bar{M}} \#\{\hat{\boldsymbol{s}}^{(m)}\} \cdot \mathcal{T}^{(m)}}{\sum_{\substack{m=1 \\ m \text{ is odd}}}^{\bar{M}} \#\{\boldsymbol{s}^{(m)}\} \cdot \mathcal{T}^{(m)}}. \tag{41}$$

In our implementation, the bath influence functional is computed using a recently proposed fast algorithm based on the inclusion-exclusion principle [50, Section 2], whose computational complexity is $O(2^m)$. Thereby, asymptotically we have

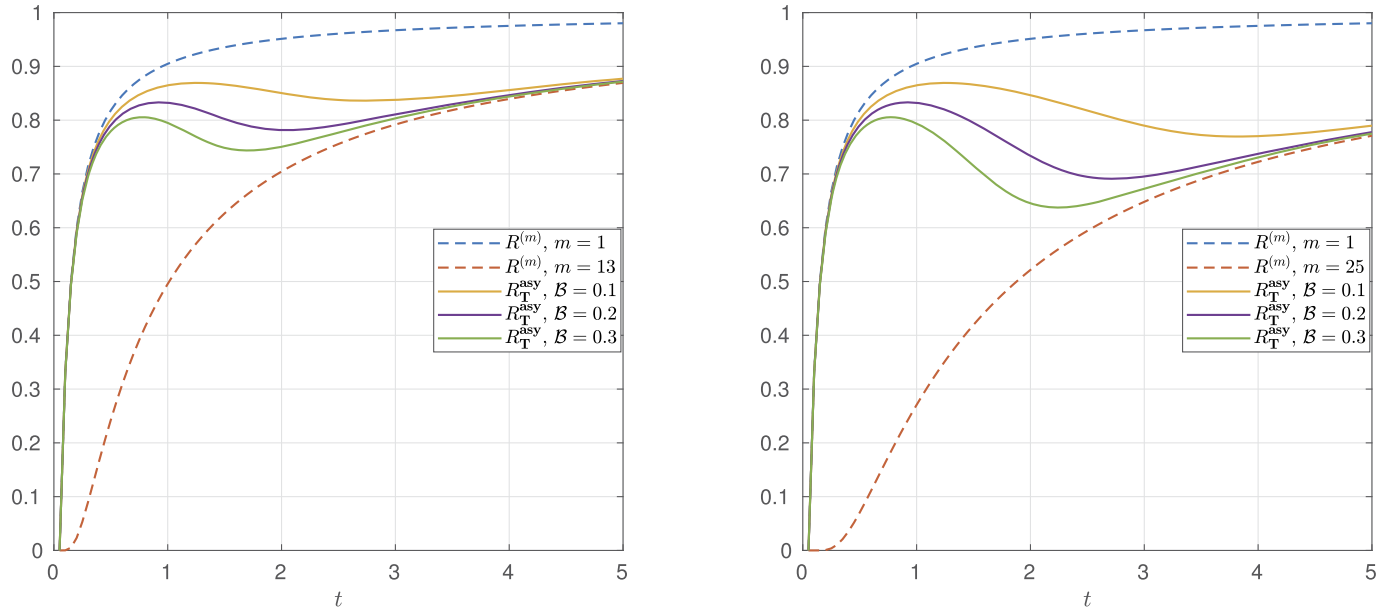


Fig. 3. Graphs of $R^{(m)}$ and $R^{\mathrm{asy}}_{\mathrm{T}}$ for Dyson series (left: $\bar{M} = 13$, right: $\bar{M} = 25$).

$$R_{\mathrm{T}} \sim R^{\mathrm{asy}}_{\mathrm{T}} := 1 - \frac{\sum_{\substack{m=1 \\ m \text{ is odd}}}^{\bar{M}} \#\{\hat{\boldsymbol{s}}^{(m)}\} \cdot 2^m}{\sum_{\substack{m=1 \\ m \text{ is odd}}}^{\bar{M}} \#\{\boldsymbol{s}^{(m)}\} \cdot 2^m} = 1 - \frac{\sum_{m=1}^{\frac{\bar{M}+1}{2}} \frac{(4\sqrt{\mathcal{B}} t_N)^{2m-1}}{(2m-2)!!}}{\sum_{m=1}^{\frac{\bar{M}+1}{2}} \sum_{i=1}^{N} \frac{(4\sqrt{\mathcal{B}} t_i)^{2m-1}}{(2m-2)!!}}$$

where $\mathcal{B}$ is the parameter describing the amplitude of two-point correlation. For large N, this can be approximated by

$$R^{\mathrm{asy}}_{\mathrm{T}} \sim 1 - \frac{\bar{M}+1}{N},$$

which agrees with Fig. 3 where $R^{\mathrm{asy}}_{\mathrm{T}}$ (solid lines) converges to $R^{(\bar{M})}$ as t grows. This behavior is due to the fact that more bath influence functionals with $m = \bar{M}$ are sampled when t gets larger, and the cost for the evaluation of these $(\bar{M}+1)$-point functionals becomes dominant. For the same reason, the graph of $R^{\mathrm{asy}}_{\mathrm{T}}$ becomes closer to $R^{(\bar{M})}$ as $\mathcal{B}$ increases.

3. Fast implementation of inchworm Monte Carlo method

The idea of the fast algorithm for summing Dyson series can also be applied to the inchworm Monte Carlo method introduced in [7], which computes the two-variable *full propagator* $G(s_{\mathrm{i}}, s_{\mathrm{f}})$, which generalizes $G(-t, t)$ defined in (4) to any initial time point $s_{\mathrm{i}} \in [-t, t]\backslash\{0\}$ and final time point $s_{\mathrm{f}} \in [s_{\mathrm{i}}, t]\backslash\{0\}$. Similar to (22), the inchworm method can also be formulated as an integro-differential equation with bath influence functionals of any time series between s_{i} and s_{f} inside the integral. This structure again allows us to reuse the bath influence functionals computed in previous time steps. Below we will review the formulas of the inchworm Monte Carlo method before introducing our numerical method.

3.1. Introduction to inchworm Monte Carlo method

3.1.1. Full propagator

The full propagator $G(s_{\mathrm{i}}, s_{\mathrm{f}})$ is formulated by

$$G(s_{\mathrm{i}}, s_{\mathrm{f}}) = G^{(0)}_s(s_{\mathrm{i}}, s_{\mathrm{f}}) + \sum_{\substack{m=2 \\ m \text{ is even}}}^{+\infty} \mathrm{i}^m \int_{s_{\mathrm{i}} \le \boldsymbol{s} \le s_{\mathrm{f}}} \mathrm{d}\boldsymbol{s} (-1)^{\#\{\boldsymbol{s}<0\}} \mathcal{U}^{(0)}(s_{\mathrm{i}}, \boldsymbol{s}, s_{\mathrm{f}}) \cdot \mathcal{L}_b(\boldsymbol{s}), \tag{42}$$

where $G^{(0)}_s(s_{\mathrm{i}}, s_{\mathrm{f}})$ is given in (6). When $s_{\mathrm{i}} = s_{\mathrm{f}}$, it is defined as $G(s_{\mathrm{i}}, s_{\mathrm{f}}) = \mathrm{Id}$. Note that this definition is consistent with the Dyson series (4) if we set $s_{\mathrm{i}} = -t$ and $s_{\mathrm{f}} = t$. The following properties of $G(\cdot, \cdot)$ will be found useful later in the numerical method:

Proposition 5.

- ***Shift invariance:*** *For any $\Delta t \ge 0$, if $s_{\mathrm{f}} + \Delta t < 0$ or $s_{\mathrm{i}} \ge 0$, we have*

$$G(s_{\mathrm{i}} + \Delta t, s_{\mathrm{f}} + \Delta t) = G(s_{\mathrm{i}}, s_{\mathrm{f}}). \tag{43}$$

- ***Conjugate symmetry:*** *For any $-t \le s_{\mathrm{i}} \le s_{\mathrm{f}} < t$, we have*

$$G(-s_{\mathrm{f}}, -s_{\mathrm{i}}) = G(s_{\mathrm{i}}, s_{\mathrm{f}})^{\dagger} \tag{44}$$

- ***Jump condition:*** *$G(\cdot,\cdot)$ is discontinuous on the line segments $[-t,0]\times\{0\}$ and $\{0\}\times[-t,0]$ and*

$$\begin{aligned}\lim_{s_\mathrm{f}\to 0^+} G(s_\mathrm{i},s_\mathrm{f}) &= O_s \lim_{s_\mathrm{f}\to 0^-} G(s_\mathrm{i},s_\mathrm{f});\\ \lim_{s_\mathrm{i}\to 0^-} G(s_\mathrm{i},s_\mathrm{f}) &= \lim_{s_\mathrm{i}\to 0^+} G(s_\mathrm{i},s_\mathrm{f}) O_s.\end{aligned} \tag{45}$$

The rigorous proofs for the statements above are omitted as (43) and (45) can be immediately derived by (14) and the definition of $\mathcal{U}^{(0)}$ respectively, and the proof of (44) is identical to that of (20) in Lemma 1 by changing the variables $\boldsymbol{s}$ in the integral to $\boldsymbol{s}'$.

3.1.2. Integro-differential equation for $G(s_\mathrm{i},s_\mathrm{f})$

The full propagator has been proved to satisfy the following integro-differential equation [7]:

$$\frac{\partial G(s_\mathrm{i},s_\mathrm{f})}{\partial s_\mathrm{f}} = \mathrm{sgn}(s_\mathrm{f})\left[\mathrm{i}H_s G(s_\mathrm{i},s_\mathrm{f}) + \sum_{\substack{m=1\\ m\text{ is odd}}}^{+\infty} \mathrm{i}^{m+1} \int_{s_\mathrm{i}\le \boldsymbol{s}\le s_\mathrm{f}} \mathrm{d}\boldsymbol{s}(-1)^{\#\{\boldsymbol{s}<0\}} W_s \mathcal{U}(s_\mathrm{i},\boldsymbol{s},s_\mathrm{f}) \mathcal{L}_b^c(\boldsymbol{s},s_\mathrm{f})\right]. \tag{46}$$

Here we recall that W_s is the perturbation associated with the system, and $\mathrm{sgn}(\cdot)$ is the sign function. $\mathcal{U}$ is defined similarly to $\mathcal{U}^{(0)}$ with the bare propagator $G_s^{(0)}(\cdot,\cdot)$ replaced by the full propagator $G(\cdot,\cdot)$:

$$\mathcal{U}(s_\mathrm{i},\boldsymbol{s},s_\mathrm{f}) = G(s_m,s_\mathrm{f}) W_s G(s_{m-1},s_m) W_s \cdots W_s G(s_1,s_2) W_s G(s_\mathrm{i},s_1). \tag{47}$$

The definition of $\mathcal{L}_b^c$ is similar to the bath influence functional $\mathcal{L}_b$:

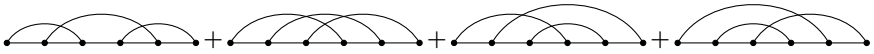
$$\mathcal{L}_b^c(s_1,\cdots,s_m,s_\mathrm{f}) = \sum_{\mathfrak{q}\in\mathcal{Q}^c(\boldsymbol{s},s_\mathrm{f})} \prod_{(s_j,s_k)\in\mathfrak{q}} B(s_j,s_k), \tag{48}$$

but $\mathcal{Q}^c$ is a subset of $\mathcal{Q}$ appearing in $\mathcal{L}_b$ which only includes "linked" pairings, which means in its diagrammatic representation any two points can be connected with each other using arcs as "bridges". For example when $m=3$, $\mathcal{L}_b^c(s_1,s_2,s_3,s_\mathrm{f})$ only contains one linked diagram in (11):

$\mathcal{L}_b^c(s_1,s_2,s_3,s_\mathrm{f}) =$  $= B(s_1,s_3)B(s_2,s_\mathrm{f}).$ (49)

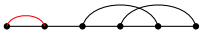
Another example for $m=5$ is given by

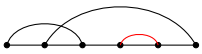
$\mathcal{L}_b^c(s_1,s_2,s_3,s_4,s_5,s_\mathrm{f})$

$=$ 

$$\begin{aligned}&:= B(s_1,s_3)B(s_2,s_5)B(s_4,s_\mathrm{f}) + B(s_1,s_4)B(s_2,s_5)B(s_3,s_\mathrm{f})\\ &\qquad + B(s_1,s_4)B(s_2,s_\mathrm{f})B(s_3,s_5) + B(s_1,s_5)B(s_2,s_4)B(s_3,s_\mathrm{f})\end{aligned} \tag{50}$$

which does not include the unlinked terms in the bath influence functional $\mathcal{L}_b(s_1,s_2,s_3,s_4,s_5,s_\mathrm{f})$ such as

 $:= B(s_1,s_2)B(s_3,s_5)B(s_4,s_\mathrm{f}),$

 $:= B(s_1,s_3)B(s_2,s_\mathrm{f})B(s_4,s_5), \quad \cdots$ (51)

where the pairs marked in red do not connect to the rest part of the diagrams via the arc bridges. Compared with the Dyson series, working with equation (46) is more advantageous as the series in the right-hand side has a faster convergence with respect to m. Also, $\mathcal{L}_b^c(s_1,\cdots,s_m,s_\mathrm{f})$ includes fewer diagrams than $\mathcal{L}_b(s_1,\cdots,s_m,t)$ in equation (22) for the Dyson series, making its direct evaluation cheaper than the bath influence functional for small m. However, asymptotically the number of diagrams in $\mathcal{L}_b^c(s_1,\cdots,s_m,s_\mathrm{f})$ also grows as a double factorial [43], and its evaluation for large m is even more expensive than $\mathcal{L}_b(s_1,\cdots,s_m,t)$ [50]. Therefore, we again look for possible reuse of computed bath influence functionals when evolving the numerical solution.

3.2. Numerical method

Again, we truncate the series in the integro-differential equation up to a finite $\tilde{M}$ as an approximation and apply the Runge-Kutta method for discretization on a uniform triangular mesh plotted in Fig. 4(a). For simplicity, we first consider the first-order forward Euler scheme:

$$\tilde{G}_{j,k} = \tilde{G}_{j,k-1} + \mathrm{sgn}(t_{k-1})h\left[\mathrm{i}H_s\tilde{G}_{j,k-1} + \sum_{\substack{m=1\\ m\text{ is odd}}}^{\tilde{M}} \mathrm{i}^{m+1} \int_{t_j\le \boldsymbol{s}\le t_{k-1}} \mathrm{d}\boldsymbol{s}(-1)^{\#\{\boldsymbol{s}<0\}} W_s \tilde{\mathcal{U}}_I(t_j,\boldsymbol{s},t_{k-1}) \mathcal{L}_b^c(\boldsymbol{s},t_{k-1})\right] \tag{52}$$

for $-N\le j<k\le N$ with $N=t_{\max}/h$. Here each $\tilde{G}_{j,k}$ is the approximation of the exact solution $G(t_j,t_k)$ and is denoted by a dot in Fig. 4(a). Since $\mathcal{U}$ defined in (53) contains $G(s_k,s_{k-1})$ not on the grid points, we need to approximate $\mathcal{U}$ using $\tilde{\mathcal{U}}_I$ defined by

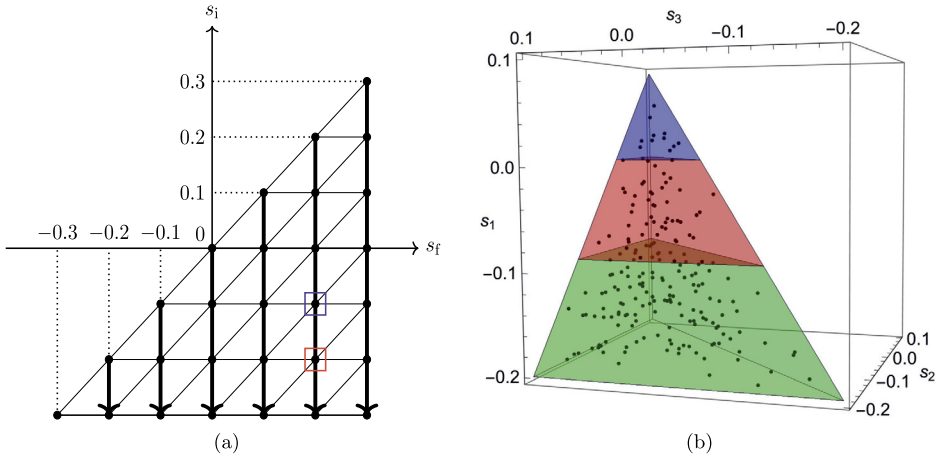


Fig. 4. An example for $h = 0.1$, $N = 3$ and $t = 0.3$ (left: mesh structure, right: decomposition of the domain of integration $\{-0.2 \le s_1 \le s_2 \le s_3 \le 0.1\}$).

$$\tilde{\mathcal{U}}_I(t_j, s_1, \cdots, s_m, t_{k-1}) = \tilde{G}_I(s_m, t_{k-1}) W_s \tilde{G}_I(s_{m-1}, s_m) W_s \cdots W_s \tilde{G}_I(s_1, s_2) W_s \tilde{G}_I(t_j, s_1) \tag{53}$$

where the interpolating function $\tilde{G}_I(\cdot,\cdot)$ satisfies $\tilde{G}_I(t_\ell, t_n) = \tilde{G}_{\ell,n}$, for all $j \le \ell \le n \le k-1$, and the piecewise linear interpolation is adopted in our implementation. To ensure these $\tilde{G}_{\ell,n}$ are available before evaluating (53), we compute the full propagator column by column from left to right in Fig. 4(a). For each of these columns, we compute from top to bottom along the corresponding arrow. In order to better present the reuse of computed bath influence functionals, we consider the following decomposition of the domain of integration:

$$\{\boldsymbol{s} = (s_1, \cdots, s_m) \mid t_j \le s_1 \le \cdots \le s_m \le t_{k-1}\} = \bigcup_{p=j}^{k-2} T^{(m)}_{p,k-1} \tag{54}$$

where

$$T^{(m)}_{p,k-1} = \{(s_1, \cdots, s_m) \mid t_p \le s_1 \le t_{p+1}, s_1 \le s_2 \le \cdots \le s_m \le t_{k-1}\}, \tag{55}$$

which are pairwise disjoint for $p = j, \cdots, k-2$. This decomposition can be visualized using the example in Fig. 4: when we use the scheme (52) to evaluate $\tilde{G}_{-2,2}$ (node in red box in Fig. 4(a)), the domain of integration for $m = 3$ is the simplex $\{(s_1, s_2, s_3) \mid -0.2 \le s_1 \le s_2 \le s_3 \le 0.1\}$ plotted in Fig. 4(b). According to the decomposition (54), this simplex can be split into $T^{(3)}_{0,1}$ (blue tetrahedron), $T^{(3)}_{-1,1}$ (red pentahedron) and $T^{(3)}_{-2,1}$ (green pentahedron). This decomposition allows us to reuse the bath integrand factor $\mathcal{L}^c_b(\boldsymbol{s}, t_{k-1})$ when computing an integral in (52) via

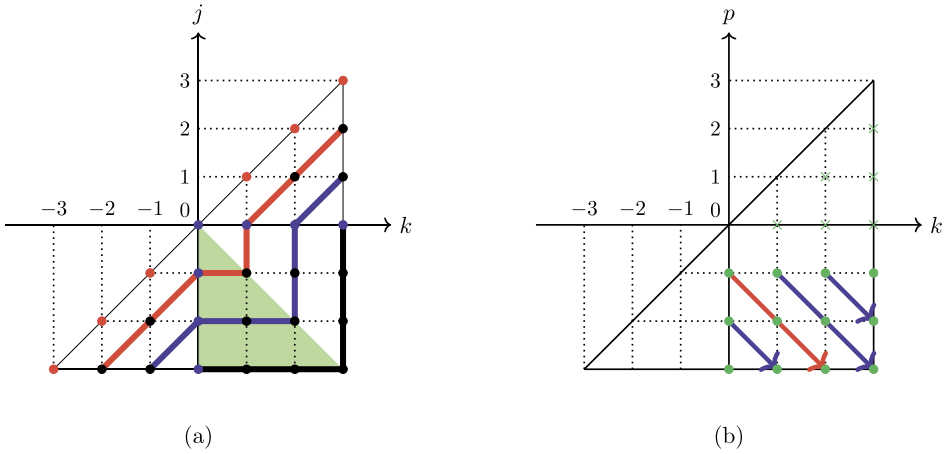
$$\int_{t_j \le \boldsymbol{s} \le t_{k-1}} \mathrm{d}\boldsymbol{s} (-1)^{\#\{\boldsymbol{s}<0\}} W_s \tilde{\mathcal{U}}_I(t_j, \boldsymbol{s}, t_{k-1}) \mathcal{L}^c_b(\boldsymbol{s}, t_{k-1}) = \int_{t_{j+1} \le \boldsymbol{s} \le t_{k-1}} \mathrm{d}\boldsymbol{s} (\text{integrand}) + \int_{\boldsymbol{s} \in T^{(m)}_{j,k-1}} \mathrm{d}\boldsymbol{s} (\text{integrand}) \tag{56}$$

where the value of $\mathcal{L}^c_b(\boldsymbol{s}, t_{k-1})$ in the second integral above has been obtained when calculating $\tilde{G}_{j+1,k}$, while $\mathcal{L}^c_b(\boldsymbol{s}, t_{k-1})$ in the last integral should be newly evaluated. This reuse of bath calculation can also be understood by the same example in Fig. 4: when evaluating $\tilde{G}_{-2,2}$, the values of $\mathcal{L}^c_b(\boldsymbol{s}, 0.1)$ for $\boldsymbol{s} = (s_1, s_2, s_3)$ in $\{-0.1 \le s_1 \le s_2 \le s_3 \le 0.1\}$ (points in blue and red pentahedra) can be reused from $\tilde{G}_{-1,2}$ (node in blue box in Fig. 4(a)), and $\mathcal{L}^c_b(\boldsymbol{s}, 0.1)$ for $\boldsymbol{s} \in T^{(2)}_{-2,1}$ (points in the green pentahedron) are to be calculated newly. However, we remark that such reuse does not apply to the entire integrand as the value of $\tilde{\mathcal{U}}_I$ replies on the t_j, which are different in $\tilde{G}_{-2,2}$ and $\tilde{G}_{-1,2}$.

At this point, we draw time sequences $\mathcal{S}_{p,k-1} := \{\boldsymbol{s}^{(i)}_{p,k-1}\}_{i=1}^{\mathcal{M}_{p,k-1}}$ from $T_{p,k-1} = \bigcup_{\substack{m=1 \\ m \text{ is odd}}}^{\bar{M}} T^{(m)}_{p,k-1}$ and approximate the sum of integrals in (52) using Monte Carlo method. The numerical scheme becomes

$$G_{j,k} = G_{j,k-1} + \mathrm{sgn}(t_{k-1}) h \Bigg[\mathrm{i} H_s G_{j,k-1} + \frac{1}{\sum_{p=j}^{k-2} \mathcal{M}_{p,k-1}} \sum_{p=j}^{k-2} \sum_{i=1}^{\mathcal{M}_{p,k-1}} \frac{1}{\mathbb{P}_{j,k-1}(m^{(i)}_{p,k-1}, \boldsymbol{s}^{(i)}_{p,k-1})} \times \\ \times \mathrm{i}^{m^{(i)}_{p,k-1}+1} (-1)^{\#\{\boldsymbol{s}^{(i)}_{p,k-1}<0\}} W_s \mathcal{U}_I(t_j, \boldsymbol{s}^{(i)}_{p,k-1}, t_{k-1}) \mathcal{L}^c_b(\boldsymbol{s}^{(i)}_{p,k-1}, t_{k-1}) \Bigg]$$

where the function $\mathbb{P}_{j,k-1}(m, \boldsymbol{s})$ gives the probability density of $(m, \boldsymbol{s})$ in $\bigcup_{p=j}^{k-2} T_{p,k-1}$, and the functional $\mathcal{U}_I$ is similarly defined as (53) with all $\tilde{G}_I$ replaced by G_I. The reuse of bath influence functionals stated in (56) is also reflected in the above scheme: when evaluating $G_{j,k}$, $\mathcal{L}^c_b(\boldsymbol{s}^{(i)}_{p,k-1}, t_{k-1})$ for $p = j+1, \cdots, k-2$ have already been obtained when computing $G_{j+1,k}$, and $\mathcal{L}^c_b(\boldsymbol{s}^{(i)}_{p,k-1}, t_{k-1})$ for $p = j$ should be newly calculated. Such implementation also indicates that one should follow a proper order to evolve the scheme, which will be discussed in detail in the next section.



(a) (b)

Fig. 5. An example for $N = 3$ (left: time evolution of inchworm Monte Carlo method, right: sampling and reuse strategy).

To achieve a higher convergence order in time, we now put Heun's method (23) into this framework and the corresponding inchworm Monte Carlo method reads:

$$\begin{aligned} G^*_{j,k} =& G_{j,k-1} + \mathrm{sgn}(t_{k-1})h\Bigg[\mathrm{i}H_s G_{j,k-1} + \frac{1}{\sum_{p=j}^{k-2}\mathcal{M}_{p,k-1}}\sum_{p=j}^{k-2}\sum_{i=1}^{\mathcal{M}_{p,k-1}}\frac{1}{\mathbb{P}_{j,k-1}(m^{(i)}_{p,k-1}, \boldsymbol{s}^{(i)}_{p,k-1})}\times \\ & \times \mathrm{i}^{m^{(i)}_{p,k-1}+1}(-1)^{\#\{\boldsymbol{s}^{(i)}_{p,k-1}<0\}} W_s \mathcal{U}_I(t_j, \boldsymbol{s}^{(i)}_{p,k-1}, t_{k-1})\mathcal{L}^c_b(\boldsymbol{s}^{(i)}_{p,k-1}, t_{k-1})\Bigg], \\ G_{j,k} =& \frac{1}{2}(G_{j,k-1} + G^*_{j,k}) + \frac{1}{2}\mathrm{sgn}(t_k)h\Bigg[\mathrm{i}H_s G^*_{j,k} + \frac{1}{\sum_{p=j}^{k-1}\mathcal{M}_{p,k}}\sum_{p=j}^{k-1}\sum_{i=1}^{\mathcal{M}_{p,k}}\frac{1}{\mathbb{P}_{j,k}(m^{(i)}_{p,k}, \boldsymbol{s}^{(i)}_{p,k})}\times \\ & \times \mathrm{i}^{(m^{(i)}_{p,k}+1)}(-1)^{\#\{\boldsymbol{s}^{(i)}_{p,k}<0\}} W_s \mathcal{U}^*_I(t_j, \boldsymbol{s}^{(i)}_{p,k}, t_k)\mathcal{L}^c_b(\boldsymbol{s}^{(i)}_{p,k}, t_k)\Bigg] \end{aligned} \tag{57}$$

where $\mathcal{U}^*_I$ in the second stage is given by

$$\mathcal{U}^*_I(t_j, s_1, \cdots, s_m, t_{k+1}) = G^*_I(t_m, s_{k+1})W_s G^*_I(s_{m-1}, s_m)W_s \cdots W_s G^*_I(s_1, s_2)W_s G^*_I(t_j, s_1)$$

with

$$G^*_I(t_\ell, t_n) = \begin{cases} G_{\ell,n}, & \text{if } (\ell, n) \neq (j, k+1), \\ G^*_{j,k+1}, & \text{if } (\ell, n) = (j, k+1). \end{cases}$$

In general, the inchworm Monte Carlo method (57) for the integro-differential equation (46) is similar to the scheme (26) for Dyson series, but for the inchworm method, some special care needs to be taken at time $t = 0$, which will be detailed in the next section.

3.2.1. General procedure of the inchworm Monte Carlo method

To apply the numerical scheme (57) accurately and efficiently, we need to take the properties of $G(\cdot,\cdot)$ into consideration, which leads us to the rules below that we should follow during the implementation:

(R1) The evolution of the numerical scheme should begin with the boundary value $G_{j,j} = \mathrm{Id}$ for $j = -N, \ldots, N$, which are denoted by the red dots in Fig. 5(a).

(R2) Due to the discontinuities, when $j = 0$ or $k = 0$ (blue dots in Fig. 4(a)), $G_{j,k}$ is considered to be multiple-valued, and we use $G_{0^\pm,k}$ and $G_{j,0^\pm}$ respectively to represent the approximation of the left and right limits $\lim_{s\to 0^\pm} G(s, t_k)$ and $\lim_{s\to 0^\pm} G(t_j, s)$. By the jump condition (45), we have the relation

$$G_{j,0^+} = O_s G_{j,0^-} \text{ and } G_{0^-,k} = G_{0^+,k}O_s \text{ for } -N \le j \le -1, 1 \le k \le N.$$

In particular, the boundary value on the discontinuities are given by: $G_{0^+,0^+} = G_{0^-,0^-} = \mathrm{Id}$ and $G_{0^-,0^+} = O_s$. Consequently, the interpolation of G_I appearing in the functional $\mathcal{U}_I$ should satisfy

$$\lim_{s\to 0^\pm} G_I(t_j, s) = G_{j,0^\pm}, \qquad \lim_{s\to 0^\pm} G_I(s, t_k) = G_{0^\pm,k},$$
$$\lim_{s\to 0^+}\lim_{\tilde{s}\to 0^+} G_I(\tilde{s}, s) = \lim_{s\to 0^-}\lim_{\tilde{s}\to 0^-} G_I(s, \tilde{s}) = \mathrm{Id}, \qquad \lim_{s\to 0^-}\lim_{\tilde{s}\to 0^+} G_I(s, \tilde{s}) = O_s$$

and the conditions for G^*_I in $\mathcal{U}^*_I$ are similar. This rule is indispensable in our implementation to keep the second-order convergence rate in time of the Heun's method.

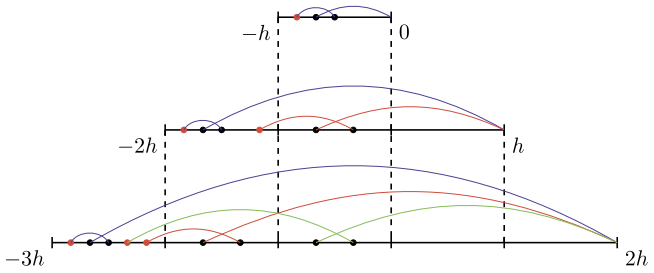


Fig. 6. Calculation reuse of $\mathcal{L}_b^c(s_1, s_2, s_3, t_k)$ along the path $G_{-1,0} \to G_{-2,1} \to G_{-3,2}$.

(R3) We only compute $G_{j,k}$ locating in the green triangle in Fig. 5(a) excluding the origin. Afterwards, the value of $G_{-k,-j}$ is obtained by the conjugate symmetry (44). In addition, the following full propagators are assigned with the same values according to the shift invariance (43):

$$\begin{aligned} &G_{-N,-N+j} = G_{-N+1,-N+j+1} = \cdots = G_{-j-1,-1} = G_{-j,0^-}, \\ &G_{N-j,N} = G_{N-j-1,N-1} = \cdots = G_{1,j+1} = G_{0^+,j} \text{ for } 1 \le j \le N-1. \end{aligned} \tag{58}$$

Algorithm 2 Evolution of inchworm Monte Carlo method

input $\mathcal{S}_{p,k} = \{\boldsymbol{s}_{p,k}^{(i)}\}_{i=1}^{\mathcal{M}_{p,k}} \subset T_{p,k}$ and $L_{p,k} = \{\mathcal{L}_b^c(\boldsymbol{s}_{p,k}^{(i)}, t_k)\}_{i=1}^{\mathcal{M}_{p,k}}$ for $-N \le p \le k-1$ and $0 \le k \le N$
Set $G_{j,j} \leftarrow \mathrm{Id}$ for $j = -N, \cdots, -1, 0^-, 0^+, 1, \cdots, N$ and $G_{0^-,0^+} \leftarrow O_s$ ▷ *Initial condition*
for n from 1 to N **do** ▷ *Time evolution on nth thick segment in Fig. 5(a)*
  Compute $G_{-n,0^-}$ by (57) and set $G_{-n,0^+} \leftarrow O_s G_{-n,0^-}$, $G_{0^\pm,n} \leftarrow G_{-n,0^\mp}$ ▷ *Compute blue dots*
  for ℓ_1 from 1 to n **do** ▷ *Compute inside quadrant IV*
    Compute G_{-n,ℓ_1} by (57) and set $G_{-\ell_1,n} \leftarrow (G_{-n,\ell_1})^\dagger$
  end for
  for ℓ_2 from 1 to $N-n-1$ **do** ▷ *Assign values in quadrant I and III*
    Set $G_{-n-\ell_2,-\ell_2} \leftarrow G_{-n,0^-}$ and $G_{\ell_2,n+\ell_2} \leftarrow G_{0^+,n}$
  end for
end for
return $G_{j,k}$ for $-N \le j \le k \le N$

We are now ready to sketch the implementation of the numerical scheme (57) in Algorithm 2. For the example in Fig. 5(a), the computation of full propagators should be first carried out on the thick red segment, followed by the blue thick segment and finally the black one. Such order of calculation is to guarantee that the values of shorter propagation $G_{n,\ell}$ for $j \le n \le \ell \le k$ are available before computing $G_{j,k}$ as argued previously. On each of these segments, we only evaluate $G_{j,k}$ in the green triangle from left to right, and the rest dots on the same segment can be directly assigned according to (R3). With the evolution of numerical scheme clarified, we now focus on the efficient construction of the input $\mathcal{S}_{p,k}$ and $L_{p,k}$, which will be specified in the next section.

3.2.2. *Time sequence sampling and reuse of bath calculation*

Due to the invariance and symmetry of the full propagators, one only needs to evaluate $G_{j,k}$ in the green triangular area in Fig. 5(a) where the index (j,k) satisfies $0 \le k \le -j$ and $-N \le j \le -1$. Since computing $G_{j,k}$ requires samples in $T_{p,k-1}$ for $j \le p \le k-2$ and $T_{p,k}$ for $j \le p \le k-1$ (see (57)), we need to prepare time sequences $\mathcal{S}_{p,k}$ and calculate $L_{p,k}$ for $-N \le p \le k-1$ and $0 \le k \le N$ according to the splitting (54). These indices (p,k) are denoted by the green nodes (both "×" and "•") in Fig. 5(b). In fact, we can focus only on those "•" nodes since the set of samples $\mathcal{S}_{p,k}$ for (p,k) on "×" nodes can be obtained by shifting the samples in $\mathcal{S}_{p-k,0}$:

$$\boldsymbol{s}_{p,k}^{(i)} = \boldsymbol{s}_{p-k,0}^{(i)} + t_k \mathbf{1} \text{ for } 0 \le p < k \le N$$

where $\mathbf{1}$ is the row vector with all its components being 1. We can then use (15) to directly assign the bath value for these "×" nodes:

$$\mathcal{L}_b^c(\boldsymbol{s}_{p,k}^{(i)}, t_k) = \overline{\mathcal{L}_b^c(\boldsymbol{s}_{p-k,0}^{(i)}, 0)}.$$

Note that this cannot be applied to the green "•" nodes as the corresponding $(\boldsymbol{s}_{p,k}^{(i)}, t_k)$ contains both positive and negative entries and thus the condition of (15) cannot be satisfied.

From here, we will only work with $\mathcal{S}_{p,k}$ with "•" indices and consider the reuse of bath influence functionals for inchworm Monte Carlo method. Since the full propagator is now defined in the two-dimensional half-space, we have multiple paths following which the bath influence functionals can be reused and each of these paths is represented by an arrow in the quadrant IV of Fig. 5(b). For example, the red arrow denotes the reuse of $\mathcal{L}_b^c(s_1, \cdots, s_m, t_k)$ for calculating $G_{-1,0} \to G_{-2,1} \to G_{-3,2}$, which is illustrated in Fig. 6. When $m = 3$, each $\mathcal{L}_b^c(s_1, s_2, s_3, t_k)$ is represented by a linked diagram as defined in (49), where the time sequence (s_1, s_2, s_3) denoted by the three dots is a sample in $T_{p,k}^{(3)}$. Note that in each diagram, the leftmost dot marked in red should always be restricted within $[t_p, t_{p+1}]$ by the definition of $T_{p,k}^{(m)}$. One can easily see that Proposition 3 also applies to $\mathcal{L}_b^c$. Hence, by the stretching invariance, the diagrams with the same color have the same functional value. In addition, we remark that the shifting invariance such as the reuse of red arc in Fig. 2 is no longer considered now since the left end t_p and right end t_k satisfy $t_p < 0 \le t_k$ for all "•" nodes in Fig. 5(b).

Similar to algorithm for Dyson series, we consider the sample space $\hat{T}_{p,k}$ from which we draw new time sequences in each time step (e.g., the blue diagram in $(-h, 0)$, the red diagram in $(-2h, h)$ and the green diagram in $(-3h, 2h)$). To do this, we generalize (28) to two indices:

$$\hat{T}_{p,k}^{(m)} = \left\{ (s_1, \cdots, s_m) \in T_{p,k}^{(m)} \;\middle|\; -h < t_k < h \text{ or } \exists\, s_i \text{ such that } -h < s_i < h \right\}, \tag{59}$$

where $p = -N, \cdots, -1$ and $k = 0, \cdots, N$. The volume of $\hat{T}_{p,k}^{(m)}$ is similarly given by the relation:

$$|\hat{T}_{p,k}^{(m)}| = \begin{cases} |T_{p,k}^{(m)}| - |T_{p+1,k-1}^{(m)}|, & \text{if } -N \le p \le -2, 1 \le k \le N, \\ |T_{p,k}^{(m)}|, & \text{if } p = -1 \text{ or } k = 0, \end{cases} \tag{60}$$

where $|T_{p,k}^{(m)}|$ can be calculated by the definition (55):

$$|T_{p,k}^{(m)}| = \int_{t_p}^{t_{p+1}} \mathrm{d}s_1 \int_{s_1 \le s_2 \le \cdots \le s_m} \mathrm{d}s_2 \cdots \mathrm{d}s_m = \frac{1}{m!}[(t_{k-p})^m - (t_{k-p-1})^m]. \tag{61}$$

In general, our algorithm for inchworm Monte Carlo method is summarized in Algorithm 3. Lines 2–9 build up the time sequences $\mathcal{S}_{p,k}$ and bath influence functionals $L_{p,k}$ along the arrows in Fig. 5(b) following the strategy similar to Algorithm 1 for the Dyson series, and Lines 10–14 construct $\mathcal{S}_{p,k}$ and $L_{p,k}$ with "$\times$" indices. The sampling method for the input $\hat{\mathcal{S}}_{p,k}$ is also similar to that for the Dyson series: the number of samples in $\hat{\mathcal{S}}_{p,k}$ is set as

$$\hat{\mathcal{M}}_{p,k}^{(m)} = \frac{\hat{\mathcal{M}}_0}{\hat{\lambda}} \cdot |\hat{T}_{p,k}^{(m)}| \cdot m!! \mathcal{B}^{\frac{m+1}{2}} \tag{62}$$

where $\hat{\mathcal{M}}_0 = \hat{\mathcal{M}}_{-1,0}^{(1)}$ and $\hat{\lambda} = \mathcal{B}h$. Each sample is again generated according to the uniform distribution $U(\hat{T}_{p,k}^{(m)})$. The formula of the density function $\mathbb{P}_{j,k}(m, \boldsymbol{s})$ used in the numerical scheme is then given by the following proposition:

Proposition 6. *For any $-N \le j < k \le N$ and $m = 1, 3, \cdots, \bar{M}$,*

$$\mathbb{P}_{j,k}(m, \boldsymbol{s}) = \frac{1}{\lambda_{j,k}} \cdot m!! \mathcal{B}^{\frac{m+1}{2}} \tag{63}$$

where

$$\lambda_{j,k} = \sum_{\substack{m'=1 \\ m' \text{ is odd}}}^{\bar{M}} \frac{(t_{k-j})^{m'}}{(m'-1)!!} \cdot \mathcal{B}^{\frac{m'+1}{2}}$$

The proof of this proposition is almost identical to that of Proposition 4 and thus omitted.

Algorithm 3 Efficient implementation of inchworm Monte Carlo method

1: **input** $\hat{\mathcal{S}}_{p,k} = \{\boldsymbol{s}_{p,k}^{(i)}\}_{i=1}^{\hat{\mathcal{M}}_{p,k}} \subset \hat{T}_{p,k}$ for $-N \le p \le -1$ and $0 \le k \le N$
2: **for** n from 1 to N **do** ▷ *Reuse on the arrows covering $(N-n+1)$ "•" in Fig. 5(b)*
3: **for** ℓ from 0 to $N-n$ **do**
4: Compute $\hat{L}_{-1-\ell,n+\ell} = \{\mathcal{L}_b^c(\boldsymbol{s}_{-1-\ell,n+\ell}^{(j)}, t_{n+\ell})\}_{j=1}^{\hat{\mathcal{M}}_{-1-\ell,n+\ell}}$ ▷ *Arrows starting from $p = -1$*
5: Compute $\hat{L}_{-n-\ell,\ell} = \{\mathcal{L}_b^c(\boldsymbol{s}_{-n-\ell,\ell}^{(j)}, t_\ell)\}_{j=1}^{\hat{\mathcal{M}}_{-n-\ell,\ell}}$ ▷ *Arrows starting from $k = 0$*
6: Set $\mathcal{S}_{-1-\ell,n+\ell} \leftarrow \bigcup_{j=0}^{\ell} \mathcal{I}_j(\hat{\mathcal{S}}_{-1,n})$; $\mathcal{S}_{-n-\ell,\ell} \leftarrow \bigcup_{j=0}^{\ell} \mathcal{I}_j(\hat{\mathcal{S}}_{-n,0})$ ▷ *Stretch samples*
7: Set $L_{-1-\ell,n+\ell} \leftarrow \bigcup_{j=0}^{\ell} \hat{L}_{-1,n}$; $L_{-n-\ell,\ell} \leftarrow \bigcup_{j=0}^{\ell} \hat{L}_{-n,0}$ ▷ *Reuse $\hat{L}$ values*
8: **end for**
9: **end for**
10: **for** n from 1 to N **do** ▷ *Assign values of "$\times$"*
11: **for** ℓ from 1 to $N-n+1$ **do**
12: Set $\mathcal{S}_{-n+\ell,\ell} \leftarrow \{\boldsymbol{s}_{-n,0}^{(j)} + t_\ell \cdot \mathbf{1}^{(m_{-n,0}^{(j)})}\}_{j=1}^{\hat{\mathcal{M}}_{-n,0}}$ and $L_{-n+\ell,\ell} \leftarrow \hat{L}_{-n,0}$
13: **end for**
14: **end for**
15: **return** $\mathcal{S}_{p,k}$ and $L_{p,k}$ for $-N \le p \le k-1$ and $0 \le k \le N$

Remark 1. Unlike the method based on Dyson series, low memory cost implementation for inchworm method is not available. The system factor $\mathcal{U}_I^*$ in the numerical scheme (57) now depends on the previously computed full propagators, which prohibits the preparation of all the partial sums like we did in Section 2.5. Consequently, these sums have to be computed sequentially, and all the bath influence functionals have to be stored to gain the efficiency. One possible workaround for long-time simulations is to restrict the memory length like in the iterative QuAPI method [28]. We will leave this to future works.

3.3. Analysis on computational cost

We again examine the computational cost saved after reusing the bath calculations for inchworm Monte Carlo method. By (62), the total number of samples in $\hat{\mathcal{S}}_{p,k}$ with "•" indices in Fig. 5(b) is in general given by

$$\#\{\hat{\boldsymbol{s}}^{(m)}\} = \frac{\hat{\mathcal{M}}_0}{\hat{\lambda}} \cdot \left(\sum_{n=1}^{N} \sum_{\ell=0}^{N-n} |\hat{T}^{(m)}_{-1-\ell,n+\ell}| + |\hat{T}^{(m)}_{-n-\ell,\ell}| \right) \cdot m!! \mathcal{B}^{\frac{m+1}{2}} \tag{64}$$

where $|\hat{T}^{(m)}_{p,k}|$ is summed along the arrows in the quadrant IV. Applying the relation (60) on each arrow yields

$$\begin{aligned} \#\{\hat{\boldsymbol{s}}^{(m)}\} &= \frac{\hat{\mathcal{M}}_0}{\hat{\lambda}} \cdot \left(\sum_{n=1}^{N} |T^{(m)}_{-1-N+n,N}| + |T^{(m)}_{-N,N-n}| \right) \cdot m!! \mathcal{B}^{\frac{m+1}{2}} \\ &= \frac{\hat{\mathcal{M}}_0 \mathcal{B}^{\frac{m+1}{2}}}{\hat{\lambda}(m-1)!!} \cdot \left[(t_{2N})^m + (t_{2N-1})^m - (t_N)^m - (t_{N-1})^m \right]. \end{aligned}$$

On the other hand, similar to (37) for Dyson series, the number of all time sequences, denoted by $\#\{\boldsymbol{s}^{(m)}\}$, is expressed by (64) with the volume $|\hat{T}^{(m)}_{p,k}|$ replaced by $|T^{(m)}_{p,k}|$. Note that the value of $|T^{(m)}_{p,k}|$ only depends on the difference $k-p$ according to (61), we therefore have

$$\begin{aligned} \#\{\boldsymbol{s}^{(m)}\} &= \frac{\hat{\mathcal{M}}_0}{\hat{\lambda}} \cdot \left(\sum_{i=1}^{2N} \sum_{\substack{-N \le p \le -1, \\ 0 \le k \le N, \\ k-p=i}} |T^{(m)}_{p,k}| \right) \cdot m!! \mathcal{B}^{\frac{m+1}{2}} \\ &= \frac{\hat{\mathcal{M}}_0}{\hat{\lambda}} \cdot \left(\sum_{j=1}^{N} j \cdot \frac{(t_j)^m - (t_{j-1}^m)}{m!} + \sum_{j=N+1}^{2N} (2N-j+1) \cdot \frac{(t_j)^m - (t_{j-1}^m)}{m!} \right) \cdot m!! \mathcal{B}^{\frac{m+1}{2}} \\ &= \frac{\hat{\mathcal{M}}_0 \mathcal{B}^{\frac{m+1}{2}}}{\hat{\lambda}(m-1)!!} \cdot \left(\sum_{j=N+1}^{2N} (t_j)^m - \sum_{j=1}^{N-1} (t_j)^m \right). \end{aligned}$$

Thus, for the order-m bath influence functionals, the proportion of the computational cost saved by the reuse is

$$R^{(m)}(N) = 1 - \frac{\#\{\hat{\boldsymbol{s}}^{(m)}\}}{\#\{\boldsymbol{s}^{(m)}\}} = 1 - \frac{(2N)^m + (2N-1)^m - (N)^m - (N-1)^m}{\sum_{j=N+1}^{2N} (j)^m - \sum_{j=1}^{N-1} (j)^m}. \tag{65}$$

For large N, the denominator can again be estimated by Faulhaber's formula (39):

$$\sum_{j=N+1}^{2N} (j)^m - \sum_{j=1}^{N-1} (j)^m = \sum_{j=1}^{2N} (j)^m - 2 \sum_{j=1}^{N-1} (j)^m - N^m \sim \frac{(2N)^{m+1} - 2(N-1)^{m+1}}{m+1} - N^m,$$

yielding the following asymptotic growth of $R^{(m)}$:

$$R^{(m)}(N) \sim 1 - \frac{1 - (\frac{1}{2})^{m+1}}{1 - (\frac{1}{2})^m} \cdot \frac{m+1}{N}, \tag{66}$$

which also converges to 1 at the rate $O(\frac{1}{N})$. It can be seen that this asymptotic value is close to $1-(m+1)/N$ as in (40) for the Dyson series, especially for large m. In particular, one can check that (38) and (65) are equal when $m=1$. This similarity can be verified by the graphs of $R^{(m)}$ in Fig. 7 where the dashed lines are almost identical to those in Fig. 3 for Dyson series, suggesting that inchworm Monte Carlo method can benefit the same reduction in the computational cost of $\mathcal{L}_b^c(s_1, \cdots, s_m)$ after reusing the bath calculations. By further taking the computational complexity of $\mathcal{L}_b^c(s_1, \cdots, s_m)$ into consideration, which is $O(\alpha^m)$ with $\alpha \approx 2.1258$ upon applying the inclusion-exclusion principle [50, Section 3], the overall reduction of the computational cost can again be formulated as (41) with $\mathcal{T}^{(m)}$ denoting the average wall clock time on evaluating $\mathcal{L}_b^c(s_1, \cdots, s_m, s_f)$, which has the following asymptotic behavior:

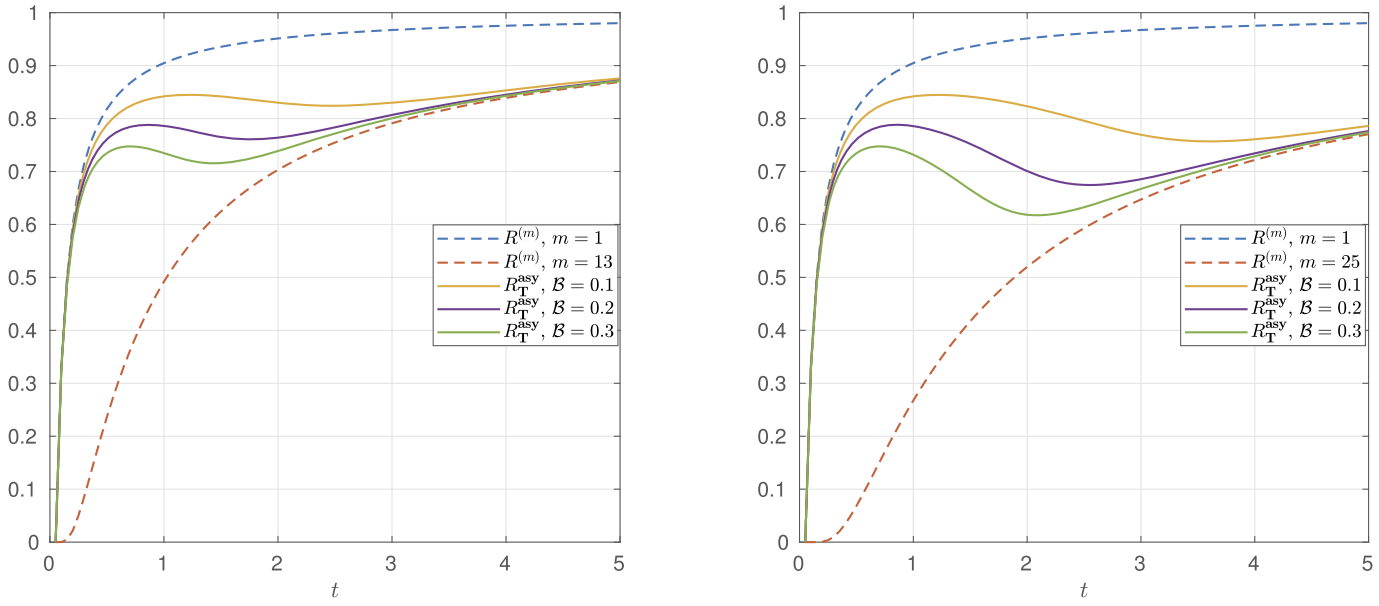


Fig. 7. Graphs of $R^{(m)}$ and $R_{\mathrm{T}}^{\mathrm{asy}}$ for inchworm Monte Carlo method (left: $\bar{M}=13$, right: $\bar{M}=25$).

$$
\begin{aligned}
R_{\mathrm{T}} \sim R_{\mathrm{T}}^{\mathrm{asy}} &= 1 - \frac{\sum_{\substack{m=1 \\ m \text{ is odd}}}^{\bar{M}} \#\{\hat{\boldsymbol{s}}^{(m)}\} \cdot \alpha^m}{\sum_{\substack{m=1 \\ m \text{ is odd}}}^{\bar{M}} \#\{\boldsymbol{s}^{(m)}\} \cdot \alpha^m} \\
&= 1 - \frac{\sum_{m=1}^{\frac{\bar{M}+1}{2}} \frac{(\sqrt{\mathcal{B}}\alpha)^{2m-1}}{(2m-2)!!} \cdot \left[(t_{2N})^{2m-1} + (t_{2N-1})^{2m-1} - (t_N)^{2m-1} - (t_{N-1})^{2m-1}\right]}{\sum_{m=1}^{\frac{\bar{M}+1}{2}} \frac{(\sqrt{\mathcal{B}}\alpha)^{2m-1}}{(2m-2)!!} \cdot \left(\sum_{j=N+1}^{2N} (t_j)^{2m-1} - \sum_{j=1}^{N-1} (t_j)^{2m-1}\right)} \\
&\sim 1 - \frac{(\bar{M}+1)\left(2 - (\frac{1}{2})^{\bar{M}}\right)}{2\left(1 - (\frac{1}{2})^{\bar{M}}\right)} \cdot \frac{1}{N} \text{ as } N \to +\infty.
\end{aligned}
$$

This ratio again converges to $R^{(\bar{M})}$ as shown by the solid lines in Fig. 7.

4. Numerical experiments

In our numerical experiments, we consider the spin-boson model where the system Hamiltonian has the energy difference $\epsilon = 1$ and frequency of the spin flipping $\Delta = 1$. For the bath influence functional, we assume an Ohmic spectral density

$$J(\omega) = \frac{\pi}{2} \sum_{l=1}^{L} \frac{c_l^2}{\omega_l} \delta(\omega - \omega_l)$$

where the number of modes is set as $L = 400$. The coupling intensity c_l and frequency of each harmonic oscillator ω_l above are respectively given by

$$c_l = \omega_l \sqrt{\frac{\xi \omega_c}{L}[1 - \exp(-\omega_{\max}/\omega_c)]}, \quad \omega_l = -\omega_c \ln\left(1 - \frac{l}{L}[1 - \exp(-\omega_{\max}/\omega_c)]\right), \ l = 1, \cdots, L$$

where the maximum frequency is set as $\omega_{\max} = 4\omega_c$. Hence, the two-point correlation (8) is formulated as

$$B(\tau_1, \tau_2) = \sum_{l=1}^{L} \frac{c_l^2}{2\omega_l} \left[\coth\left(\frac{\beta\omega_l}{2}\right) \cos\left(\omega_l \Delta\tau\right) - \mathrm{i} \sin\left(\omega_l \Delta\tau\right)\right].$$

In Fig. 8, we plot the amplitude of the two-point correlation with Kondo parameter $\xi = 0.4$, inverse temperature $\beta = 5$ and primary frequency $\omega_c = 2.5$ as the orange curve. The empirical constant $\mathcal{B}$ appearing in (32) and (62) should then be chosen between $(0, 1.2)$. Larger $\mathcal{B}$ will lead to more time sequences sampled with large m, and thus a higher computational cost. In practice, one may start with a relatively small $\mathcal{B}$ to see whether the variance is small enough. If not, one may then increase $\mathcal{B}$ and repeat the simulation. According to our tests, choosing $\mathcal{B} = 0.2$ provides satisfactory results. We will also consider another numerical example with $\xi = 0.2$ for which the modulus of the two-point bath correlation is given by the blue curve in Fig. 8. The corresponding $\mathcal{B}$ is set to be 0.1. For all our numerical examples in this section, we truncate the series in (22) and (46) by $\bar{M} = 11$.

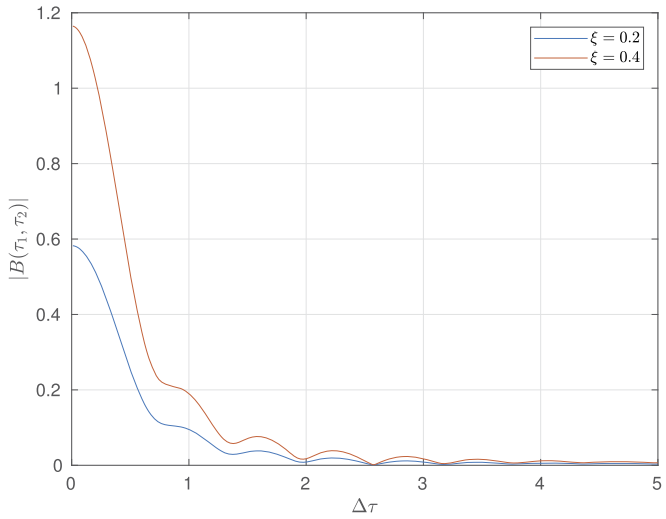


Fig. 8. Two-point correlation functions for different Kondo parameters (orange: $\xi = 0.4$, blue: $\xi = 0.2$).

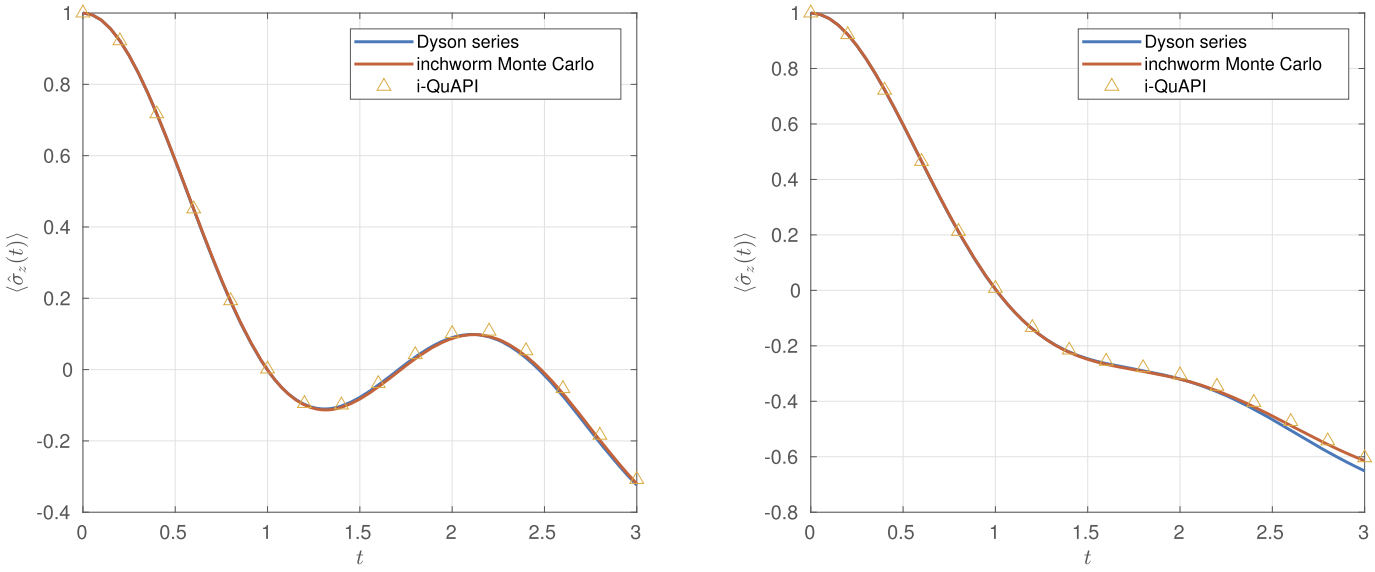


Fig. 9. Evolution of $\langle\hat{\sigma}_z(t)\rangle$ under different settings of the Kondo parameter (left: $\xi = 0.2$, right: $\xi = 0.4$).

4.1. Evolution of observable

To validate our numerical method, we first apply our reuse of bath calculations to both coupling intensities and compare the evolution of the observables to the results of classical methods. The observable of interest is set to be $O = \hat{\sigma}_z \otimes \mathrm{Id}_b$ which only acts on the system, and the initial density matrix $\rho = \rho_s \otimes \rho_b$ is given by

$$\rho_s = |0\rangle\langle 0| \quad \text{and} \quad \rho_b = Z^{-1}\exp(-\beta H_b),$$

where Z is a normalizing factor satisfying $\mathrm{tr}(\rho_b) = 1$. The evolution of observable $\langle\hat{\sigma}_z(t)\rangle$ is then evaluated discretely by

$$\langle\hat{\sigma}_z(nh)\rangle \approx \langle 0| G_{-n,n} |0\rangle \text{ for } n = 0, 1, \cdots, N$$

where $G_{-n,n}$ is computed by either scheme (26) for Dyson series or scheme (57) for inchworm Monte Carlo method.

In our numerical tests, we set the time step to be $h = 0.05$. It is generally believed that the inchworm Monte Carlo method requires less samples than the summation of the Dyson series. Therefore we set the initial number of samples $\hat{\mathcal{M}}_0$ to be 10^5 for the solver of the Dyson equation (22), and set $\hat{\mathcal{M}}_0 = 10^4$ for the inchworm Monte Carlo method. In Fig. 9, we plot the numerical results of observable for both Kondo parameters. The results by iterative QuAPI method [28,29] are also given as the reference solutions. In the left panel, the two curves are hardly distinguishable and both match the reference solution well. In the right panel, however, an obvious difference between two curves can be observed after $t = 2.5$ and the result of the iterative QuAPI method indicates that the inchworm Monte Carlo method gives a better approximation. This is due to the fact that the larger amplitude of $B(\tau_1, \tau_2)$ with $\xi = 0.4$ makes the Dyson series harder to converge with respect to m for long time simulations. As a result, the truncation $\bar{M} = 11$ is no longer sufficient for Dyson series, but still works for the inchworm Monte Carlo method thanks to its faster convergence as mentioned in Section 3.1.2.

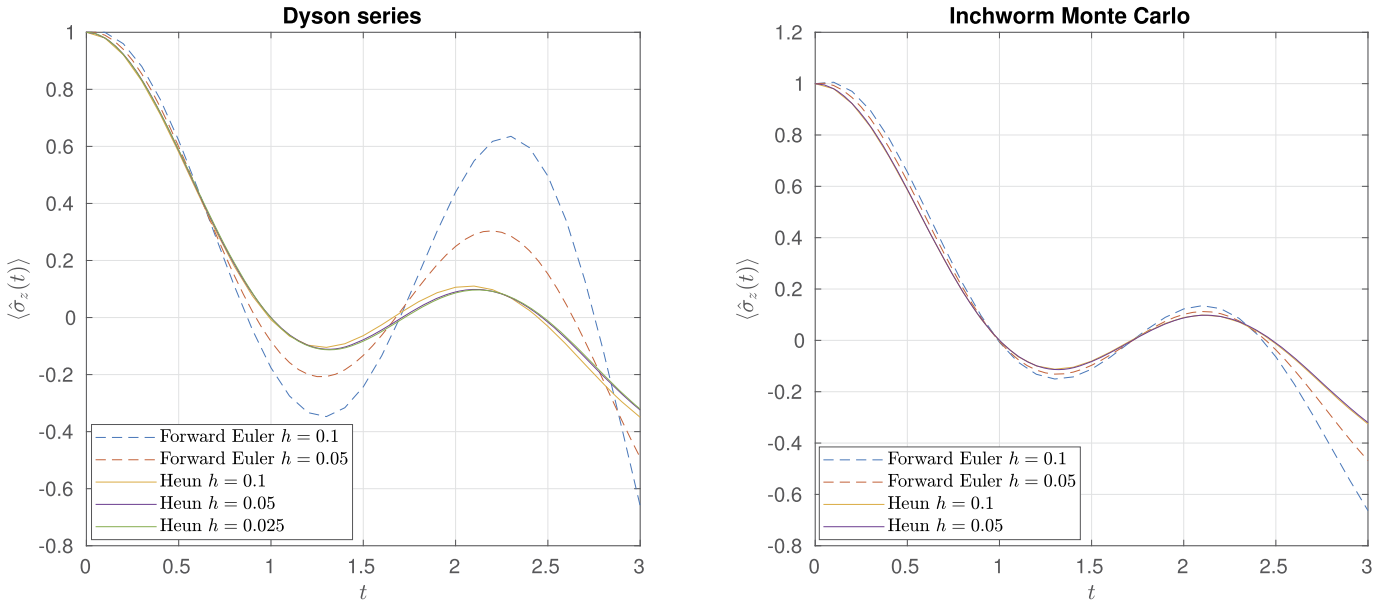


Fig. 10. Evolution of $\langle\hat{\sigma}_z(t)\rangle$ for various time step lengths.

Table 1
Wall clock time (seconds) on evaluating a given $\mathcal{U}^{(0)}(-t, s_1, \cdots, s_m, t)$ and $\mathcal{L}_b(s_1, \cdots, s_m, t)$.

m	1	3	5	7	9	11
$\mathcal{U}^{(0)}$	6.8000e-05	1.1800e-04	1.8000e-04	2.0800e-04	2.3200e-04	3.6500e-04
$\mathcal{L}_b$	1.0100e-04	3.2800e-04	6.7200e-04	0.0011	0.0016	0.0023

4.2. Accuracy test

To verify the accuracy of the numerical discretization by Heun's method used throughout this paper, we plot the results of $\langle\hat{\sigma}_z(t)\rangle$ computed by both algorithms with different time steps in Fig. 10. The parameters of simulations are set to be the same as the left panel of Fig. 9. For Dyson series, the result of $h = 0.05$ is indistinguishable with the result of $h = 0.025$ by naked eyes, while the curve for $h = 0.1$ still shows observable discrepancy with the other two lines. Note that $h = 0.05$ is used for the simulations in Fig. 9, which is now proven to be reliable according to our accuracy test. For the inchworm Monte Carlo method, the convergence is achieved at a coarser grid $h = 0.1$, which is possibly due to the smaller number of terms in the bath influence functional. As a comparison, we also plot the results by first-order Forward Euler scheme (dashed curves), which obviously have not converged at $h = 0.05$. This shows the advantage of Heun's method in terms of the accuracy of time discretization. While the second-order Heun's scheme is sufficient to produce accurate simulations up to $t = 3$ in the current work, it is also worthwhile to consider higher-order or implicit schemes for the integral-differential equations (22) and (46) to achieve better accuracy and stability.

4.3. Efficiency test

We now examine the computational time that can be saved by reusing the bath calculations. The experiments are carried out using MATLAB on AMD Ryzen 7 4800H CPU, and we use the parameters for the orange curve ($\xi = 0.4$) in Fig. 8 for the efficiency tests.

We first compare the wall clock time on evaluating a given system associated $\mathcal{U}^{(0)}$ with that on $\mathcal{L}_b$ appearing in the integrand of Dyson series. As shown in Table 1, the evaluation of $\mathcal{L}_b$ is more expensive than $\mathcal{U}^{(0)}$ in terms of time consumed for all choices of m. As m increases, this difference becomes larger due to the linear complexity of $\mathcal{U}^{(0)}$ and exponential complexity of $\mathcal{L}_b$. Therefore, the computational cost on the bath influence functional dominates the overall evaluation of a given Dyson series. Similar conclusion for the inchworm Monte Carlo method can be drawn by Table 2, where we list the wall clock time of $\mathcal{U}_I$ and $\mathcal{L}_b^c$ in the scheme (57). Here both $\mathcal{L}_b$ and $\mathcal{L}_b^c$ are computed using the fast algorithms based on inclusion-exclusion principle as mentioned previously. Instead of directly summing the linked diagrams in (50), a given $\mathcal{L}_b^c(\boldsymbol{s}, t)$ is evaluated indirectly under such algorithms which relies on the value of $\mathcal{L}_b(\boldsymbol{s}, t)$ as well as $\mathcal{L}_b(\tilde{\boldsymbol{s}}, t)$ for some subsequences $\tilde{\boldsymbol{s}} \subset \boldsymbol{s}$, making $\mathcal{L}_b^c$ in general more expensive than $\mathcal{L}_b$ despite the fact that $\mathcal{L}_b^c$ contains fewer diagrams. We refer the readers to [50] for more details of the algorithm. On the other hand, the computation of $\mathcal{U}_I$ defined by (53) in inchworm method is faster than $\mathcal{U}^{(0)}$ defined by (5) in Dyson series since each matrix G_I in $\mathcal{U}_I$ is obtained by linear interpolation, which is cheaper than $G_s^{(0)}$ in $\mathcal{U}^{(0)}$ where a matrix exponential is to be computed.

In Fig. 11, we plot the theoretical savings in computational time spent on bath computations R_{T} defined by (41) as the yellow solid lines, where the average wall clock time $\mathcal{T}^{(m)}$ for $\mathcal{L}_b$ in Dyson series and $\mathcal{L}_b^c$ in inchworm Monte Carlo method are respectively assigned with the values in Table 1 and 2. The graphs of $R^{(1)}$ and $R^{(11)}$ are also plotted as the reference. As augured in Section 2.5 and 3.3, R_{T} is always bounded by $R^{(1)}$ and $R^{(11)}$.

Meanwhile, we carry out two sets of numerical simulations under both methods with the initial number of samples $\hat{\mathcal{M}}_0 = 100$. In the first set of simulations, we apply the bath calculation reuse and record the total time spent on the bath influence functional up to

Table 2
Wall clock time (seconds) on evaluating a given $\mathcal{U}_I(s_\mathrm{i}, s_1, \cdots, s_m, s_\mathrm{f})$ and $\mathcal{L}_b^c(s_1, \cdots, s_m, s_\mathrm{f})$.

m	1	3	5	7	9	11
$\mathcal{U}_I$	2.8000e-05	5.3000e-05	7.2000e-05	1.2600e-04	1.5900e-04	1.7000e-04
$\mathcal{L}_b^c$	8.8000e-05	4.0200e-04	0.0010	0.0025	0.0053	0.0118

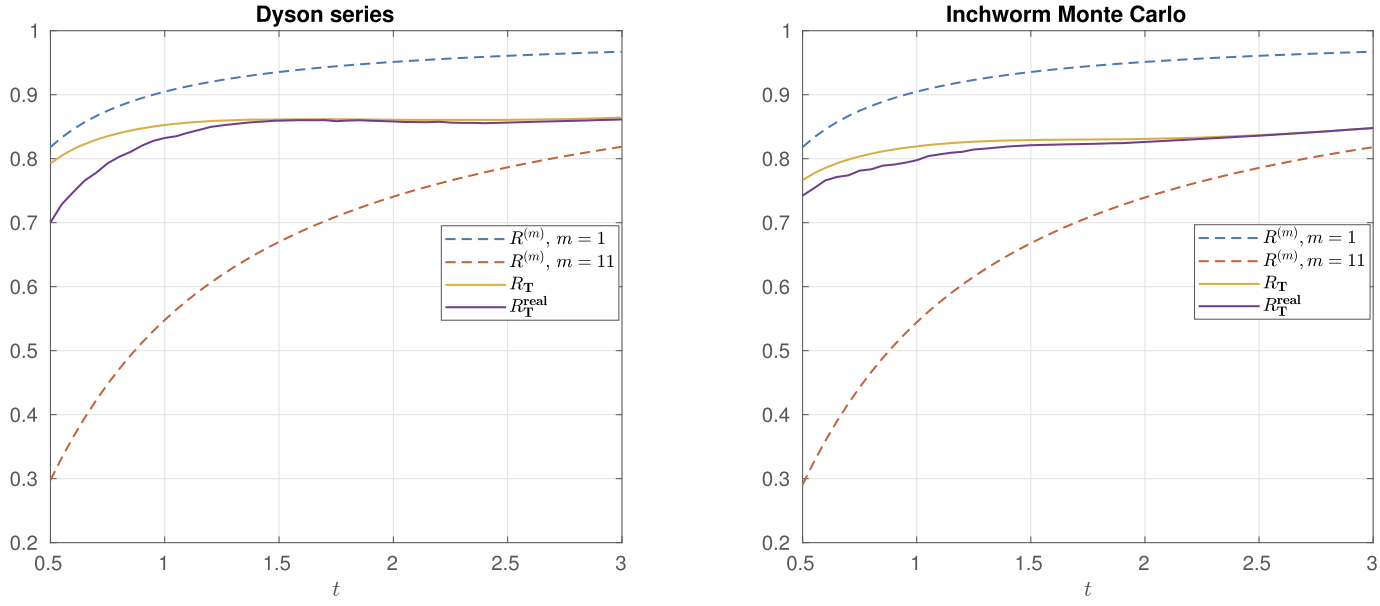


Fig. 11. Overall savings of computational time spent on bath calculations.

nth time step as $\hat{\mathcal{T}}(n)$, while the second set are implemented without reusing calculations and the time consumed on bath is denoted by $\mathcal{T}(n)$. Then we may use the ratio

$$R_\mathrm{T}^\mathrm{real}(n) = 1 - \hat{\mathcal{T}}(n)/\mathcal{T}(n)$$

to measure the overall saving in time in real implementations, which are plotted as the purple solid lines in Fig. 11. Since each evaluation on $\mathcal{L}_b$ or $\mathcal{L}_b^c$ cannot cost exactly the same amount of time, some oscillations can be observed in the purple curves. Nevertheless, $R_\mathrm{T}^\mathrm{real}$ generally matches the theoretical R_T as t grows, and thus we have verified the complexity analysis in Section 2.5 and 3.3. As time further evolves, we may expect the overall saving in time to gradually converge to $R^{(11)}$ (orange dashed lines). Therefore, asymptotically the bath calculation reuse can achieve a total reduce in computational time at around the percentage $1 - \frac{12}{n}$ for both Dyson series and inchworm Monte Carlo method for this example according to (40) and (66).

4.4. Order of convergence

As both numerical methods we have developed are stochastic schemes based on Monte Carlo, it is of interest to study the convergence rate of the standard derivation of the numerical solution with respect to the initial number of samples $\hat{\mathcal{M}}_0$. In this experiment, we fix the time step length as $h = 0.1$ and compute up to $t = 1$. The parameter setting for the two-point correlation is given as $\omega_c = 1$, $\xi = 0.1$ and $\beta = 0.2$ with the empirical constant $\mathcal{B} = 0.3$. We run the same simulation independently for $N_\mathrm{exp} = 1000$ times, and the standard derivation of $G_{-n,n}$ is estimated as

$$\sigma_{\hat{\mathcal{M}}_0}(t_n) = \left(\frac{1}{N_\mathrm{exp}} \sum_{k=1}^{N_\mathrm{exp}} \left\| G_{-n,n}^{[k]} - \mu_{-n,n} \right\|_\mathrm{F}^2 \right)^{1/2}, \quad \text{for } n = 0, 1, \cdots, 20$$

where $\|\cdot\|_\mathrm{F}$ denotes the Frobenius norm. Here $G_{-n,n}^{[k]}$ is the result of kth numerical simulation, and $\mu_{-n,n}$ should be the expectation of $G_{-n,n}$, which in our implementation is replaced by the numerical exact solution $G_{-n,n}$ that is computed based on a large initial number of samples $\hat{\mathcal{M}}_0 = 10^6$ for Dyson series and $\hat{\mathcal{M}}_0 = 10^5$ for inchworm Monte Carlo method. The numerical results are shown in Fig. 12, where the $1/2$ order of convergence for the standard derivation is obvious, indicating that the optimal convergence rate of Monte Carlo method is achieved in both stochastic schemes.

5. Conclusion

We propose fast algorithms by reusing calculations of bath influence functionals to accelerate the summation of Dyson series and inchworm Monte Carlo method in the simulation of system-bath dynamics. For Dyson series, an integro-differential equation is derived, allowing us to solve the evolution of the observables using classic numerical schemes such as Runge-Kutta type methods. The idea of

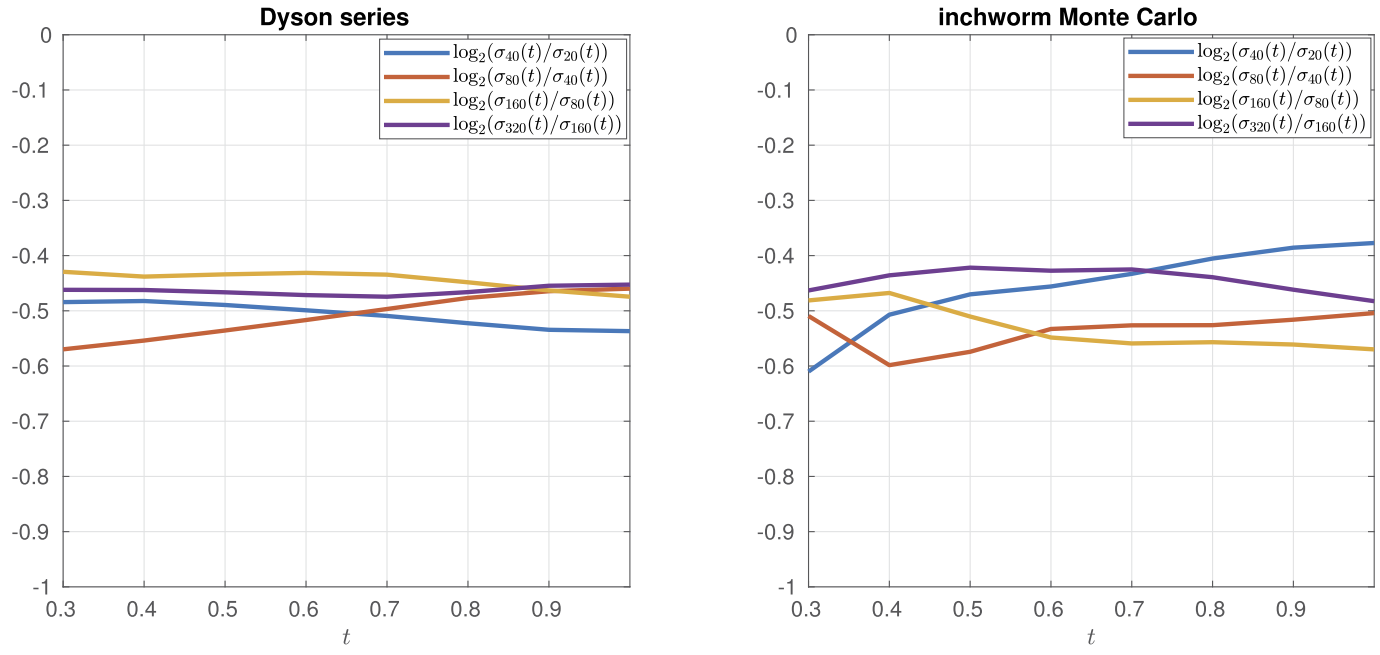


Fig. 12. Evolution of the convergence rate of the standard derivation for the numerical solution.

our fast algorithm is to make use of the invariance of the bath influence functionals so that any bath influence functional computed in the current time step can be reused in all the future time steps. Thanks to the linearity of the governing equation, the reuse algorithm for Dyson series can be implemented at a low memory cost. Such idea is then extended to the inchworm Monte Carlo method which computes the observables via the bivariate full propagator $G(s_i, s_f)$, where the bath influence functionals calculated during the computation $G(s_i, s_f)$ can be reused when computing $G(s_i - \tau, s_f + \tau)$ for any $\tau > 0$. According to our complexity analysis, the computational cost is saved by a factor of N with N being the number of time steps, which makes our algorithms efficient for long time simulations. These theoretical results are further verified by numerical experiments.

While we mainly focus on spin-boson model in this paper, our acceleration strategy can be also applied to general quantum systems interacting with the harmonic bath, where the value of two-point correlation $B(\tau_1, \tau_2)$ only relies on the time difference $\Delta\tau = |\tau_1| - |\tau_2|$ as in (8). We also point out that for certain parameter settings of $B(\tau_1, \tau_2)$, a very small truncation at $\bar{M} = 1$ or $\bar{M} = 3$ may be sufficient for Dyson series and inchworm Monte Carlo method (see such examples in [7, Section 7]). In these cases, computational cost on the system integrand factor is comparable to that on the bath as shown in Table 1 and 2. Therefore, including the system associated functional in the calculation reuse will be an interesting future direction. In addition, as the storage of bath influence functionals is the Achilles' heel of inchworm method in the current framework, further explorations into the memory cost reduction are also worth considering in future works.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Proof of statements

A.1. Proof of (13)

Proof. • If $s_i \le s_f < 0$, we have

$$G_s^{(0)}(s_i, s_f)^\dagger = \left(e^{-i(s_f - s_i)H_s}\right)^\dagger = e^{-i(s_i - s_f)H_s},$$

$$\overline{B(s_i, s_f)} = \overline{B^*(s_f - s_i)} = B^*(s_i - s_f)$$

Since $0 < -s_f \le -s_i$ in this case,

$$G_s^{(0)}(-s_f, -s_i) = e^{-i(s_i - s_f)H_s} = G_s^{(0)}(s_i, s_f)^\dagger,$$

$$B(-s_f, -s_i) = B^*(s_i - s_f) = \overline{B(s_i, s_f)}.$$

• If $0 < s_i \le s_f$, we have $-s_f \le -s_i < 0$ and thus

$$G_s^{(0)}(-s_f, -s_i) = e^{-i(s_f - s_i)H_s} = \left(e^{-i(s_i - s_f)H_s}\right)^\dagger = G_s^{(0)}(s_i, s_f)^\dagger,$$

$$B(-s_f, -s_i) = B^*(s_f - s_i) = \overline{B^*(s_i - s_f)} = \overline{B(s_i, s_f)}.$$

- If $s_{\rm i} < 0 < s_{\rm f}$, we have $-s_{\rm f} < 0 < -s_{\rm i}$ and thus

$$G_s^{(0)}(-s_{\rm f}, -s_{\rm i}) = {\rm e}^{-{\rm i}s_{\rm i}H_s} O_s {\rm e}^{-{\rm i}s_{\rm f}H_s} = \left({\rm e}^{{\rm i}s_{\rm f}H_s} O_s {\rm e}^{{\rm i}s_{\rm i}H_s}\right)^\dagger = G_s^{(0)}(s_{\rm i}, s_{\rm f})^\dagger,$$
$$B(-s_{\rm f}, -s_{\rm i}) = B^*(s_{\rm i} + s_{\rm f}) = \overline{B^*(-(s_{\rm i} + s_{\rm f}))} = \overline{B(s_{\rm i}, s_{\rm f})}.$$

The above analysis excludes the special cases $0 = s_{\rm i} \le s_{\rm f}$ and $s_{\rm i} < s_{\rm f} = 0$ for which the statement for $B(\cdot,\cdot)$ is still true, while for $G_s^{(0)}(\cdot,\cdot)$ in general it is not due to the presence of O_s.

- If $0 = s_{\rm i} < s_{\rm f}$, we have $-s_{\rm f} < 0$ and

$$G_s^{(0)}(-s_{\rm f}, -s_{\rm i}) = G_s^{(0)}(-s_{\rm f}, 0) = O_s {\rm e}^{-{\rm i}s_{\rm f}H_s} = O_s G_s^{(0)}(s_{\rm i}, s_{\rm f})^\dagger,$$
$$B(-s_{\rm f}, -s_{\rm i}) = B(-s_{\rm f}, 0) = B^*(s_{\rm f}) = \overline{B^*(-s_{\rm f})} = \overline{B(s_{\rm i}, s_{\rm f})}.$$

- If $s_{\rm i} < s_{\rm f} = 0$, we have $-s_{\rm i} > 0$ and

$$G_s^{(0)}(-s_{\rm f}, -s_{\rm i}) O_s = G_s^{(0)}(0, -s_{\rm i}) O_s = {\rm e}^{-{\rm i}s_{\rm i}H_s} O_s = G_s^{(0)}(s_{\rm i}, s_{\rm f})^\dagger,$$
$$B(-s_{\rm f}, -s_{\rm i}) = B(0, -s_{\rm i}) = B^*(s_{\rm i}) = \overline{B^*(-s_{\rm i})} = \overline{B(s_{\rm i}, s_{\rm f})}.$$

- If $s_{\rm i} = s_{\rm f} = 0$,

$$G_s^{(0)}(-s_{\rm f}, -s_{\rm i}) = G_s^{(0)}(0, 0) = I = G_s^{(0)}(s_{\rm i}, s_{\rm f})^\dagger,$$
$$B(-s_{\rm f}, -s_{\rm i}) = B(0, 0) = \frac{1}{\pi}\int_0^\infty J(\omega){\rm d}\omega = \overline{B(s_{\rm i}, s_{\rm f})}. \quad \square$$

A.2. Proof of (20) in Lemma 1

Proof. Define $s_{m+1} = t$ and $s'_0 = -t$, we have

$$\begin{aligned}
\mathcal{L}_b(-t, s_1, \cdots, s_m) &= \sum_{\mathfrak{q}' \in \mathcal{Q}(-t, \boldsymbol{s}')} \prod_{(s'_j, s'_k) \in \mathfrak{q}'} B(s'_j, s'_k) \\
&= \sum_{\mathfrak{q}' \in \mathcal{Q}(-t, \boldsymbol{s}')} \prod_{(s'_j, s'_k) \in \mathfrak{q}'} B(-s_{m+1-j}, -s_{m+1-k}) \\
&= \sum_{\mathfrak{q}' \in \mathcal{Q}(-t, \boldsymbol{s}')} \prod_{(s'_j, s'_k) \in \mathfrak{q}'} \overline{B(s_{m+1-k}, s_{m+1-j})} \\
\text{replace } j' = m+1-k, k' = m+1-j \Rightarrow &= \sum_{\mathfrak{q}' \in \mathcal{Q}(-t, -\boldsymbol{s})} \prod_{(s'_{m+1-k'}, s'_{m+1-j'}) \in \mathfrak{q}'} \overline{B(s_{j'}, s_{k'})} \\
&= \sum_{\mathfrak{q}' \in \mathcal{Q}(-t, -\boldsymbol{s})} \prod_{(-s_{k'}, -s_{j'}) \in \mathfrak{q}'} \overline{B(s_{j'}, s_{k'})} \\
&= \sum_{\mathfrak{q} \in \mathcal{Q}(\boldsymbol{s}, t)} \prod_{(s_{j'}, s_{k'}) \in \mathfrak{q}} \overline{B(s_{j'}, s_{k'})} = \overline{\mathcal{L}_b(s_1, \cdots, s_m, t)}. \quad \square
\end{aligned}$$

References

[1] A. Barvinok, in: Random Structures & Algorithms, 1999, pp. 29–61.
[2] M.H. Beck, A. Jackle, G.A. Worth, H.D. Meyer, Phys. Rep. 324 (2000) 1–105.
[3] A. Björklund, B. Gupt, N. Quesada, ACM J. Exp. Algorithmics 24 (2019).
[4] A. Boag, E. Gull, G. Cohen, Phys. Rev. B 98 (2018) 115152.
[5] H. Breuer, F. Petruccione, The Theory of Open Quantum Systems, Oxford University Press, 2007.
[6] Z. Cai, J. Lu, S. Yang, arXiv:2006.07654.
[7] Z. Cai, J. Lu, S. Yang, Commun. Pure Appl. Math. 73 (11) (2020) 2430–2472.
[8] A.O. Caldeira, A.J. Leggett, Phys. A, Stat. Mech. Appl. 121 (3) (1983) 587–616.
[9] A.O. Caldeira, A.J. Leggett, Ann. Phys. (N. Y.) 149 (2) (1983) 374–456.
[10] J. Cerrillo, J. Cao, Phys. Rev. Lett. 112 (2014) 110401.
[11] H.-T. Chen, G. Cohen, D.R. Reichman, J. Chem. Phys. 146 (2017) 054106.
[12] H.-T. Chen, G. Cohen, D.R. Reichman, J. Chem. Phys. 146 (2017) 054105.
[13] A.W. Chin, Á. Rivas, S.F. Huelga, M.B. Plenio, J. Math. Phys. 51 (9) (2010) 092109.
[14] G. Cohen, E. Gull, D.R. Reichman, A.J. Millis, Phys. Rev. Lett. 115 (26) (2015) 266802.
[15] G. Cohen, E. Gull, D.R. Reichman, A.J. Millis, E. Rabani, Phys. Rev. B 87 (2013) 195108.
[16] G. Cohen, E. Rabani, Phys. Rev. B 84 (2011) 075150.
[17] C. Duan, Z. Tang, J. Cao, J. Wu, Phys. Rev. B 95 (21) (2017) 214308.
[18] R. Egger, L. Mühlbacher, C.H. Mak, Phys. Rev. E 61 (2000) 5961–5966.
[19] R.P. Feynman, F.L. Vernon, Ann. Phys. (N. Y.) 24 (1963) 118–173.

[20] E. Gull, A.J. Millis, A.I. Lichtenstein, A.N. Rubtsov, M. Troyer, P. Werner, Rev. Mod. Phys. 83 (2011) 349–404.
[21] L.V. Keldysh, Sov. Phys. JETP 20 (4) (1965) 1018–1026.
[22] A. Kelly, T.E. Markland, J. Chem. Phys. 139 (1) (2013) 014104.
[23] D. Mac Kernan, G. Ciccotti, R. Kapral, J. Chem. Phys. 116 (6) (2002) 2346–2353.
[24] L. Kidon, H. Wang, M. Thoss, E. Rabani, J. Chem. Phys. 149 (10) (2018) 104105.
[25] D.E. Knuth, Math. Comput. 61 (203) (1993) 277–294.
[26] C.H. Mak, Phys. Rev. Lett. 68 (1992) 899–902.
[27] D.E. Makarov, N. Makri, Chem. Phys. Lett. 221 (5–6) (1994) 482–491.
[28] N. Makri, J. Math. Phys. 36 (5) (1995) 2430–2457.
[29] N. Makri, J. Phys. Chem. A 102 (24) (1998) 4414–4427.
[30] N. Makri, J. Chem. Phys. 146 (13) (2017) 134101.
[31] N. Makri, J. Chem. Theory Comput. 16 (7) (2020) 4038–4049.
[32] N. Makri, E. Sim, D.E. Makarov, M. Topaler, Proc. Natl. Acad. Sci. 93 (9) (1996) 3926–3931.
[33] H.-D. Meyer, U. Manthe, L.S. Cederbaum, Chem. Phys. Lett. 165 (1) (1990) 73–78.
[34] H. Mori, Prog. Theor. Exp. Phys. 33 (3) (1965) 423–455.
[35] L. Mühlbacher, E. Rabani, Phys. Rev. Lett. 100 (17) (2008) 176403.
[36] L. Mühlbacher, R. Egger, J. Chem. Phys. 118 (1) (2003) 179–191.
[37] S. Nakajima, Prog. Theor. Phys. 20 (6) (1958) 948–959.
[38] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition, Cambridge University Press, 2010.
[39] J. Prior, A.W. Chin, S.F. Huelga, M.B. Plenio, Phys. Rev. Lett. 105 (2010) 050404.
[40] N.V. Prokof'ev, B.V. Svistunov, Phys. Rev. Lett. 81 (1998) 2514–2517.
[41] R. Rosenbach, J. Cerrillo, S.F. Huelga, J. Cao, M.B. Plenio, New J. Phys. 18 (2) (2016) 023035.
[42] Q. Shi, E. Geva, J. Chem. Phys. 119 (23) (2003) 12063–12076.
[43] P.R. Stein, Discrete Math. 21 (1978) 309–318.
[44] J.T. Stockburger, H. Grabert, Phys. Rev. Lett. 88 (17) (2002) 170407.
[45] J. Strümpfer, K. Schulten, J. Chem. Theory Comput. 8 (8) (2012) 2808–2816.
[46] H. Wang, J. Chem. Phys. 113 (22) (2000) 9948–9956.
[47] H. Wang, M. Thoss, J. Chem. Phys. 119 (3) (2003) 1289–1299.
[48] P. Werner, A. Comanac, L. de' Medici, M. Troyer, A.J. Millis, Phys. Rev. Lett. 97 (2006) 076405.
[49] P. Werner, T. Oka, A.J. Millis, Phys. Rev. B 79 (3) (2009) 035320.
[50] S. Yang, Z. Cai, J. Lu, New J. Phys. 23 (6) (2021) 063049.
[51] M.-L. Zhang, B.J. Ka, E. Geva, J. Chem. Phys. 125 (4) (2006) 044106.
[52] E. Zwanzig, J. Chem. Phys. 33 (5) (1960) 1338–1341.