RESEARCH ARTICLE | MAY 23 2023

QuantumDynamics.jl: A modular approach to simulations of dynamics of open quantum systems

Amartya Bose

J. Chem. Phys. 158, 204113 (2023)
https://doi.org/10.1063/5.0151483

The Journal of Chemical Physics ARTICLE scitation.org/journal/jcp

QuantumDynamics.jl: A modular approach to simulations of dynamics of open quantum systems

Cite as: J. Chem. Phys. 158, 204113 (2023); doi: 10.1063/5.0151483
Submitted: 22 March 2023 • Accepted: 5 May 2023 •
Published Online: 23 May 2023

Amartya Bose[a)]

AFFILIATIONS

Department of Chemical Sciences, Tata Institute of Fundamental Research, Mumbai 400005, India

[a)]Author to whom correspondence should be addressed: amartya.bose@tifr.res.in

ABSTRACT

A simulation of the non-adiabatic dynamics of a quantum system coupled to dissipative environments poses significant challenges. New sophisticated methods are regularly being developed with an eye toward moving to larger systems and more complicated descriptions of solvents. Many of these methods, however, are quite difficult to implement and debug. Furthermore, trying to make the individual algorithms work together through a modular application programming interface can be quite difficult as well. We present a new, open-source software framework, QuantumDynamics.jl, designed to address these challenges. It provides implementations of a variety of perturbative and non-perturbative methods for simulating the dynamics of these systems. Most prominently, QuantumDynamics.jl supports hierarchical equations of motion and methods based on path integrals. An effort has been made to ensure maximum compatibility of the interface between the various methods. Additionally, QuantumDynamics.jl, being built on a high-level programming language, brings a host of modern features to explorations of systems, such as the usage of Jupyter notebooks and high level plotting, the possibility of leveraging high-performance machine learning libraries for further development. Thus, while the built-in methods can be used as end-points in themselves, the package provides an integrated platform for experimentation, exploration, and method development.

I. INTRODUCTION

Understanding the evolution of a system over time is at the heart of chemistry and physics.[1–8] While many systems can indeed be treated classically, there are several important problems where the quantum mechanical mechanism of tunneling becomes inescapable. Some of the most ubiquitous of these are charge transfer problems, excitation energy transfer processes, and spin dynamics. Additionally, in all these cases, the dynamics may be severely modulated by the existence of solvent degrees of freedom, which exist at a given temperature. The necessity of simulating the system's quantum mechanical behavior while accounting for the environment and solvents accurately proves to be significantly challenging.

Various approaches exist to tackle this problem. On the end of approximate approaches are the perturbative Bloch–Redfield Master Equations[9,10] (BRME) and methods based on empirical Lindbladians. However, these are uncontrolled approximations with no good error bounds. Therefore, it becomes important to be able to obtain the exact dynamics of these systems. Various path integral-based techniques such as the quasi-adiabatic propagator path integral[11–13] (QuAPI) family of methods and the hierarchy equations of motion[14–18] (HEOM) family of methods exist, which, at greater costs, can simulate the full non-Markovian dynamics of a quantum system coupled with dissipative media using the Feynman–Vernon influence functional.[19] Over the years, methods of unparalleled sophistication have been built on both of these frameworks, which reduce the computational costs of simulations.[20–36]

Despite the existence of a multitude of rigorous methods, software support for quantum dynamics is relatively sparse. The situation becomes especially stark when put in comparison to the plethora of alternatives, both open-source and proprietary, that exist for electronic structure theory.[37–44] The lack of easily available implementation of the latest methods prevents their widespread adoption. In addition to preventing people from being able to apply these novel ideas to a variety of problems, this has the inadvertent disadvantage of preventing critical comparison and evaluation of the different methods. In terms of providing access to multiple

state-of-the-art algorithms for dynamics in a single package, i-PI[45] is exemplary, providing flexible implementations of various methods based on imaginary time path integrals and approximate quantum dynamics using ring-polymers. However, it does not support approaches for simulating non-adiabatic processes. Recent work is trying to explore methods based on HEOM to deal with such cases in an exact manner.[46]

Among the exact methods for simulating processes that can be decomposed in terms of a quantum system interacting with a thermal bath, HEOM has a fair number of implementations.[47–50] QuTiP,[51] which supports a plethora of approximate methods for simulating open quantum systems, has an implementation of HEOM. A C++/Python software called Libra[52] and a new Julia package called NQCDynamics.jl[53] have been developed primarily for using classical trajectory-based methods for simulating non-adiabatic quantum dynamics. However, these classical trajectory-based methods are based on *ad hoc* approximations and are not numerically exact. Consequently, they can often suffer from uncontrolled errors. When it comes to numerically "exact" simulation of these systems and supporting a variety of state-of-the-art real time path integral-based methods in a modular fashion, there is a severe dearth of software. This has been a significant impediment to the approachability, adoption, and further development of these powerful methods.

Most computational codes have historically been written in C, C++, or Fortran. While performant, these languages are low-level, and their use significantly adds to the code complexity and raises the bar for others contributing to the frameworks. Of late, Python is being used for writing scientific code, with the most performance intensive parts written in C or C++. The prime examples of programs and packages using this "two-language" infrastructure are PySCF,[40] Psi4,[39] i-PI,[45] etc. A relatively new language called Julia,[54] with promise in terms of balancing performance with ease of use, has been gaining popularity in the scientific community. It features a just-in-time (JIT) compilation scheme that solves the two-language problem, where the application programming interface (API) exposes features to a high-level language but the performance-critical parts are coded in a different low-level language. The JIT compilation allows the code to run fast once the initial compilation to machine code is done. When the runtime significantly outstrips the time required for the compilation of the code, the performance of JIT compiled code compares favorably with a traditionally compiled language such as C++ or Fortran. Most scientific applications and codes fall into this category. It, consequently, becomes easy to have scientific packages written completely in Julia without sacrificing performance. There has been an explosion of packages for computational chemistry in Julia in the recent past.[53,55–59]

We introduce a new open-source software package for the Julia language called QuantumDynamics.jl[98] to provide easy access to the state-of-the-art tools for rigorous simulation of non-adiabatic systems to the community. An implementation in a high-performance, high-level language is convenient for widespread adoption and easy development in the future. Although it supports some approximate methods, the primary focus of QuantumDynamics.jl is on methods for numerically exact simulations of non-adiabatic problems. The design aims at providing atomic concepts that help maximize the reuse of code between a diverse set of path integral-based methods. Online documentation has already been provided. It will continue to be maintained, updated, and improved upon as the package changes. This paper is organized as follows: In Sec. II, we discuss the methods supported in QuantumDynamics.jl and the structure of the package. We demonstrate the usage of the package in Sec. III through representative examples of the methods. While code snippets have been provided in this paper, the full examples are in the examples folder of the repository. Some concluding remarks are provided in Sec. IV.

II. METHODS SUPPORTED AND STRUCTURE OF THE CODE

A. Methods supported

The main focus of QuantumDynamics.jl is the simulation of the dynamics of open quantum systems with non-adiabatic processes. These are characterized by a relatively small dimensional quantum system, described by a Hamiltonian, $\hat{H}_0$, interacting with N_{env} large thermal environments,

$$\hat{H} = \hat{H}_0 + \sum_b^{N_{\text{env}}} \hat{H}_{\text{env}}^{(b)}, \tag{1}$$

$$\hat{H}_{\text{env}}^{(b)} = \sum_j \frac{p_{j,b}^2}{2m_{j,b}} + \frac{1}{2} m_{j,b}\omega_{j,b}^2 \left(x_{j,b} - \frac{c_{j,b}\hat{s}_b}{m_{j,b}\omega_{j,b}^2} \right)^2, \tag{2}$$

where $\omega_{j,b}$ and $c_{j,b}$ are the frequency and coupling of the jth mode of the bth environment. The interaction between the system and the bth environment is described by $\hat{H}_{\text{env}}^{(b)}$ and happens through the system operator $\hat{s}_b$. In general, environments are atomistically defined. However, under the Gaussian response limit, it is possible to map the effects of the atomistic environment onto a bath of harmonic oscillators[60–63] through the energy gap auto-correlation function and its spectral density,

$$J_b(\omega) = \frac{\pi}{2} \sum_j \frac{c_{j,b}^2}{m_{j,b}\omega_{j,b}} \delta(\omega - \omega_{j,b}). \tag{3}$$

The famous spin-boson model is a specialization of Eq. (1) for the case of $\hat{H}_0 = \epsilon\sigma_z - \hbar\Omega\sigma_x$, where ϵ is the asymmetry between the two states and Ω is the coupling strength. The spin-boson models typically have a single harmonic bath as an environment. Two of the most common model spectral densities are

$$J_{\text{ExpCutoff}}(\omega, n) = \frac{2\pi}{\Delta s^2} \hbar\xi \frac{\omega^n}{\omega_c^{n-1}} \exp\left(-\frac{\omega}{\omega_c}\right) \tag{4}$$

and

$$J_{\text{DrudeLorentz}}(\omega) = \frac{2\lambda}{\Delta s^2} \frac{\gamma\omega}{\omega^2 + \gamma^2}. \tag{5}$$

Here, Δs is the separation between the system states. Depending on the value of n, $J_{\text{ExpCutoff}}$ represents an Ohmic spectral density ($n = 1$), a super-Ohmic spectral density ($n > 1$), or a sub-Ohmic spectral density ($n < 1$). This family of spectral densities is specified in terms of the dimensionless Kondo parameter ξ, and the cutoff frequency ω_c. The Drude–Lorentz spectral density is another Ohmic spectral density but with a Lorentzian cutoff. It is typically specified using reorganization energy λ and the characteristic bath time scale γ.

There are a variety of approaches for simulating the dynamics of these systems, ranging from completely empirical to numerically exact. The implementations are often challenging and difficult to bring to a common interface. Because of the typically strong system-environment couplings, perturbative methods of calculating dynamics are often not very accurate. However, they might still provide useful starting points for understanding the dynamics. A broad set of these exact and approximate methods is supported in QuantumDynamics.jl. They can be roughly categorized as

1. empirical approaches,
2. hierarchical equations of motion,
3. path integral approaches.

It is often difficult to maintain the consistency of the interface across these different classes of approaches. However, within every category, consistency has been ensured. QuantumDynamics.jl does not support the rich gamut of classical trajectory-based methods. Consequently, notable in its omission is the ubiquitous surface hopping method.[64–66] The NQCDynamics.jl package[53] implements surface hopping both in its fewest switching form and in connection to ring-polymer molecular dynamics. It also implements various other classical trajectory-based approaches in a modular manner.

Within the group of empirical approaches, QuantumDynamics.jl supports the propagation of both Hermitian and non-Hermitian systems. It also supports more rigorous approaches based on master equations such as the BRME and Lindblad master equations. HEOM[14,15,17,18] is implemented in its "scaled" form.[33] While many other improvements and extensions of HEOM exist in the literature, they have not yet been implemented in QuantumDynamics.jl. These will be incorporated into future versions as and when required.

The largest class of methods supported by QuantumDynamics.jl is the path integral approach. In addition to the original QuAPI,[11–13] blip decomposition of path integrals[20,21] (BSPI), the tensor network path integral[25] implementation of the time-evolving matrix product operator[31] (TEMPO) approach, and the pairwise-connected tensor network path integral[22] (PC-TNPI) method are supported. Quantum–classical path integral[67,68] (QCPI) using solvent-driven reference propagators [69] in the harmonic backreaction[70] framework has been implemented using the same interface. As elaborated in Sec. II B, the code has been designed in a way that QCPI could be used with different "backends" corresponding to QuAPI or TEMPO. Crucial to this generalization of the backends is the identification that using the forward–backward system propagator as the building block for these methods is more flexible than building it on the Hamiltonian. The bare forward–backward propagator and the one augmented by the bath influence are the common objects that connect all of these methods.

The idea of dynamical maps has been shown to be effective in understanding the non-Markovian evolution of systems.[71] The transfer tensor method[71] (TTM) allows the construction of transfer tensors from dynamical maps, which for open quantum systems are the forward–backward propagators augmented by the bath influence, $\mathcal{E}(t) = \mathrm{Tr}_{\mathrm{bath}}(\exp(-i\mathcal{L}t/\hbar))$, where $\mathcal{L}$ is the Liouvillean corresponding to the system–bath. These transfer tensors can be further used to propagate the reduced density matrix of the system beyond the memory length. This reduces the complexity of simulating the time-evolution beyond memory length to multiplying matrices of the size of the system and removes all storage requirements. TTM in QuantumDynamics.jl can take advantage of the forward–backward augmented propagators obtained from other path integral methods such as QuAPI, TEMPO, PC-TNPI, and blips.

The small matrix decomposition of path integrals[27,28] (SMatPI) is a rigorous QuAPI-based method that achieves a similar objective but with more efficient implementations for extended memory length[30] and support for the simulation of dynamics under the influence of external fields.[72] It has been noted by Makri[28] that while TTM employs time-translational invariance, leading to the generation of spurious memory, SMatPI, through a rigorous derivation based on QuAPI, lifts this limitation. QuantumDynamics.jl enables the use of tensor network-based methods such as TEMPO with TTM, which allows the inclusion of the possible spurious memory generated without a significant increase in computational complexity.

All these methods, with the exception of TTM, have been implemented in such a manner so that they can simulate the dynamics of these systems in the presence of external time-dependent fields. One of the potential applications of such time-dependent fields is the simulation of dynamics in the presence of light described, in a semiclassical manner.

B. Code structure

QuantumDynamics.jl, being a Julia package, can be used on any operating system and platform supported by the programming language. It has recently been registered with the Julia package registry. Thus, the installation procedure is relatively simple. After Julia has been setup, there are two ways to install QuantumDynamics.jl. The first way involves Julia's package manager's read-eval-print loop (REPL) interface:

```
julia> ]
pkg> add QuantumDynamics
```

The alternate is to use the `Pkg` module in Julia:

```
import Pkg
Pkg.add("QuantumDynamics")
```

All the dependencies will be automatically installed. Julia comes with implementations of OpenBlas built-in by default. However, depending on the architecture, it may be preferable to install and use Intel's Math Kernel Library (MKL), which can be installed as an additional package, MKL.jl. If MKL is used, it should be loaded before QuantumDynamics.jl in the source code.

In QuantumDynamics.jl, an attempt has been made to provide as flexible and consistent an application programming interface (API) as possible across the gamut of supported methods. This consistency is crucial in ensuring a successful mix-and-match of various approaches. However, this is an extremely challenging task given the different requirements and restrictions of various methods. In this section, we discuss some of the important design choices present in this package.

Each method has its own module. The empirical methods are completely grouped in the `Bare` module. Bloch–Redfield Master

Equation[9,10] and HEOM[18] are supported in the `Bloch-Redfield` and `HEOM` modules, respectively. The path integral methods are more varied and have been afforded their own individual modules, viz., `QuAPI`,[11,12] `Blip`,[20] `TEMPO`,[31] `PCTNPI`,[22] etc. All the path integral methods, with the exception of quantum–classical path integral,[67,68] build on top of a time-series of forward–backward propagators corresponding to the bare or isolated system. Because of this decision, it becomes possible for QCPI to use any of the base path integral methods as the engine to simulate the dynamics. For every sampled phase space point of the solvent, the QCPI routine provides the underlying path integral routine with a sequence of solvent-driven reference propagators[69] and obtains as an output the reduced density matrices after incorporation of the backreaction in the harmonic approximation.[70] The toggle of whether the full memory needs to be incorporated or just the quantum memory from the back reaction is necessary is determined by the boolean parameter, `reference_propagator`. If `reference_propagator` is false, which is the default behavior, then the full influence functional is incorporated, otherwise, only the quantum memory is incorporated. For any method, the function for simulating the dynamics of a reduced density matrix is called `propagates`. Individual methods often have convergence parameters that differ wildly from each other. All such parameters are grouped into method-specific argument types, all derived from `Utilities.ExtraArgs`.

QCPI requires the definition of a solvent, which is treated by classical trajectories. This facility is provided by the abstract struct `Solvents.Solvent`, which can be inherited from different types of solvents. Currently, only a discrete harmonic bath is provided. There is scope for providing wrappers around emerging Julia libraries for doing molecular dynamics with more detailed solvents. Associated with each solvent is a description of the corresponding phase space and an iterator, which generates phase space points that are distributed according to the thermal Boltzmann distribution.

Many of the empirical methods and HEOM require the solution of differential equations, which is done numerically using the DifferentialEquations.jl[73] package. It implements a variety of methods for solving differential equations. The details that control the differential equation solver, such as the method of simulation, relative error, and absolute error, are controlled through the structure, `Utilities.DiffEqArgs`. In QuantumDynamics.jl, the default method of solution is an adaptive Runge–Kutta approach of order 5(4),[74] though other methods can be easily used by suitably changing the `Utilities.DiffEqArgs` are passed to the method. The methods based on tensor networks are built on the open-source ITensor[75,76] library.

For the specification of the bath spectral densities, QuantumDynamics.jl provides a `SpectralDensities` module. Currently, we support `ExponentialCutoff` for Eq. (4) and `DrudeLorentz` for Eq. (5). Facilities for reading tabulated spectral densities obtained as Fourier transforms of numerically simulated bath response functions are also provided through `SpectralDensityTable`. Utility functions are provided for reading the tabulated data for both $J(\omega)$ and $J(\omega)/\omega$.

Finally, TTM[71] builds on propagators from the initial time, $t = 0$, to the final time. Thus, in addition to providing routines for propagating a reduced density matrix, the various sub-modules for path integral also provide `build_augmented_propagator` functions that calculate the time-series of propagators, including the solvent effects, using the corresponding full path methods. As detailed in the numerical examples in Sec. III, these functions make it possible to use TTM to propagate a system whose augmented propagators have been calculated using some path integral method.

The full documentation of the package also shows other examples along with a detailed description of the various arguments and parameters supported by these methods. It will remain updated as the package continues to evolve and implement other methods.

III. NUMERICAL EXAMPLES

A. Empirical approaches to open quantum systems

1. Isolated Hermitian and non-Hermitian systems

The simplest case of propagation happens to be an isolated system. The dynamics is Markovian. QuantumDynamics.jl provides an interface for simulating this dynamics both for Hermitian and non-Hermitian systems defined by a Hamiltonian, $\hat{H}$. The equation of motion for the density matrix,

$$i\hbar\partial_t\rho(t) = \hat{H}\rho(t) - \rho(t)\hat{H}^\dagger, \tag{6}$$

works for both types of systems.

Consider two degenerate states that are described by the Hamiltonian,

$$\hat{H} = \begin{pmatrix} 0.0 & -1.0 \\ -1.0 & 0.0 \end{pmatrix}. \tag{7}$$

This is a Hermitian Hamiltonian. In addition, consider a non-Hermitian Hamiltonian where the two states are lossy with different rates,

$$\hat{H}_{\rm nh} = \begin{pmatrix} -0.1i & -1.0 \\ -1.0 & -0.5i \end{pmatrix}. \tag{8}$$

In either case, the Hamiltonian can be defined as a 2×2 complex matrix or by using the convenience function `Utilities.create_tls_hamiltonian`. Currently, QuantumDynamics.jl also provides another convenience function for creating a periodic or aperiodic nearest-neighbor Hamiltonian, `Utilities.create_nn_hamiltonian`.

The dynamics of a system under these two Hamiltonians starting with a density matrix of

$$\rho(0) = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 0.0 \end{pmatrix} \tag{9}$$

is shown in Figs. 1(a) and 1(b). When a time-dependent external field, $V(t) = 12\cos(10t)$, is coupled with the operator $\hat{\sigma}_z$, the dynamics changes substantially. The dynamics under the external field for the Hermitian and non-Hermitian systems are shown in Figs. 1(c) and 1(d), respectively.

The code snippet for simulating the dynamics of the non-Hermitian system in the presence of an external field is as follows:

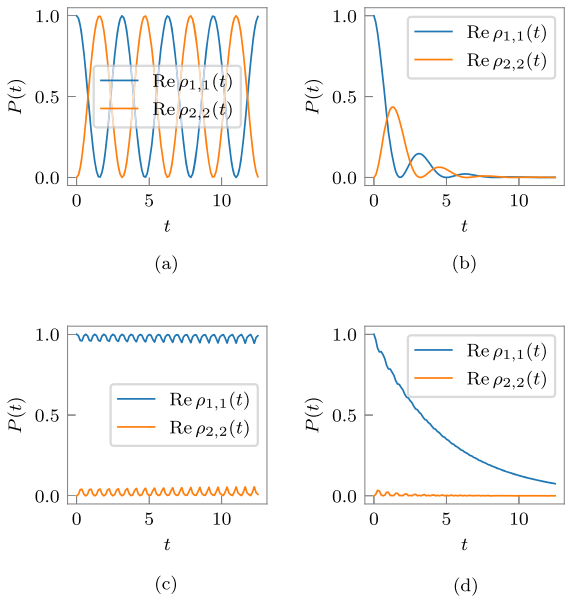


FIG. 1. Dynamics of the different elements of the density matrix. (a) Hermitian system, $\hat{H}$. (b) Non-Hermitian system, $\hat{H}_{\text{nh}}$. (c) Hermitian system, $\hat{H} + V(t)$. (d) Non-Hermitian system, $\hat{H}_{\text{nh}} + V(t)$.

```
using QuantumDynamics

# define the Hamiltonian
H = [-0.1im -1.0; -1.0 -0.5im]
V(t) = 12 * cos(10.0 * t)
EF = Utilities.ExternalField(V, [1.0+0.0im 0.0; 0.0
↪  -1.0])

# define the initial condition, the time step and the
↪  number of steps of simulation
ρ0 = [1.0+0.0im 0.0; 0.0 0.0]
dt = 0.125
ntimes = 100

# simulate the dynamics using the propagate method
times, ρs = Bare.propagate(; Hamiltonian=H, ρ0, dt,
↪  ntimes, external_fields=[EF])
```

The other cases are also similar. Notice that the Hamiltonian is defined as a simple 2×2 matrix. The design moves away from defining class hierarchies for these fundamental objects because that creates barriers when using different hardware. For example, with the current design, implementing the same algorithm on a graphics processing unit (GPU) should be as simple as using the array abstractions in a Julia library like CUDA.jl.[77] The external field, `Utilities.ExternalField`, is a `struct` with a simple function of time and the system operator that couples to the field. The same `Bare.propagate` function works for Hermitian or non-Hermitian systems with or without external fields.

2. Lindblad master equation

Consider a system interacting with a variety of environmental degrees of freedom as in Eq. (1). An empirical approach to incorporating the effects of these environments on the dynamics of the reduced density matrix (RDM) of the quantum systems is through the use of the Lindblad master equation,

$$\frac{\mathrm{d}\rho(t)}{\mathrm{d}t} = -\frac{i}{\hbar}\left[\hat{H}_0, \rho(t)\right] + \sum_j \left(L_j \rho(t) L_j^\dagger - \frac{1}{2}\left\{L_j^\dagger L_j, \rho(t)\right\}\right), \quad (10)$$

where $\hat{H}_0$ is the Hamiltonian of the system and $\rho(t)$ is the time-evolved system RDM. The impact of the environment is empirically modeled through the so-called Lindblad "jump" operators, L_j. In the absence of any jump operators, one recovers the von Neumann equation for the propagation of a density matrix. The idea of Lindbladian stems from the assumption of rapid decay of the bath correlations. The dynamics of a system described by a Lindbladian are no longer unitary but trace-preserving and completely positive for all initial conditions. Different processes require different types of jump operators. A couple of examples are demonstrated here.

For mapping a spin-boson problem onto a system described with the Lindblad master equation, we use a jump operator proportional to σ_z. The strength of the system–bath coupling in a spin-boson parameter is related to this proportionality constant. Consider a system Hamiltonian given by $\hat{H}_0 = -\sigma_x$ and a localized initial condition. The code to simulate the dynamics using QuantumDynamics.jl is

```
using QuantumDynamics

# define the Hamiltonian, the initial RDM, and the
↪  simulation details
H = Matrix{ComplexF64}([
    0.0 -1.0
    -1.0 0.0
])
ρ0 = [1.0+0.0im 0.0; 0.0 0.0]
dt = 0.125
ntimes = 100

# define the Lindbladian jump operator
L = [1.0+0.0im 0.0; 0.0 -1.0]

# call propagate with a list of jump operators to be
↪  applied
times, ρs = Bare.propagate(; Hamiltonian=H, ρ0, dt,
↪  ntimes, L=[L])
```

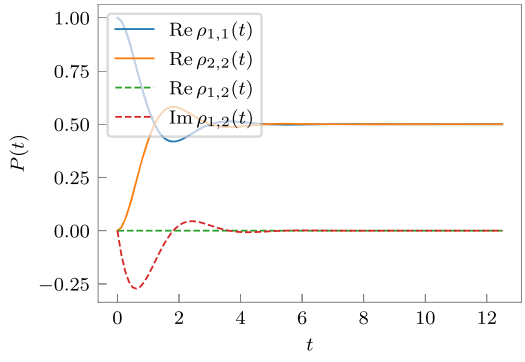


FIG. 2. Simulation of dynamics corresponding to a typical spin-boson parameter.

The resultant dynamics is shown in Fig. 2. One can notice the features reminiscent of typical spin-boson parameters.[11,12] Also note that the only change from the simulations of the isolated Hermitian and non-Hermitian systems is the new argument, L, containing a vector of jump operators L_j that is being passed in.

Now consider a more involved example. We want to model an excitation transport between a dimer of molecules while accounting for the possibility of spontaneous emission, which will bring one molecule to the ground state without exciting the other. This possibility does not allow for modeling the problem in the so-called first excitation subspace. In the full Hilbert space, the system Hamiltonian is taken to be

$$\hat{H}_0 = 20.0|ee\rangle\langle ee| + 10.0|ge\rangle\langle ge| + 10.0|eg\rangle\langle eg| - 1.0|eg\rangle\langle ge| - 1.0|ge\rangle\langle eg|. \quad (11)$$

The simulation starts with an initial condition of $|ge\rangle\langle ge|$. For the effects of the molecular vibrations moving the energies of the excited and the ground states, we use jump operators proportional to $\sigma_z \otimes \mathbb{I}$ and $\mathbb{I} \otimes \sigma_z$. To capture the spontaneous decay process, we introduce jump operators proportional to $\sigma_m \otimes \mathbb{I}$ and $\mathbb{I} \otimes \sigma_m$. The code for simulating this system is as follows:

```
using QuantumDynamics

# define Hamiltonian, initial RDM and simulation parameters
H = Matrix{ComplexF64}([
    20.0 0.0 0.0 0.0
    0.0 10.0 -1.0 0.0
    0.0 -1.0 10.0 0.0
    0.0 0.0 0.0 0.0
])

 ρ0 = Matrix{ComplexF64}([
     0.0 0.0 0.0 0.0
     0.0 1.0 0.0 0.0
     0.0 0.0 0.0 0.0
     0.0 0.0 0.0 0.0
 ])
 dt = 0.125
 ntimes = 100

 # define the jump operators corresponding to the decohering effects of the individual Born-Oppenheimer surfaces.
 # bo is the coupling strength
 σz = bo * Matrix{ComplexF64}([
     1.0 0.0
     0.0 -1.0
 ])
 id = Matrix{ComplexF64}([
     1.0 0.0
     0.0 1.0
 ])
 L1 = kron(σz, id)
 L2 = kron(id, σz)

 # define the jump operators corresponding to spontaneous emission
 # se is the coupling strength
 σm = se * Matrix{ComplexF64}([
     0.0 0.0
     1.0 0.0
 ])
 L3 = kron(σm, id)
 L4 = kron(id, σm)

 times, ρs = Bare.propagate(; Hamiltonian=H, ρ0, dt, ntimes, L=[L1, L2, L3, L4])
```

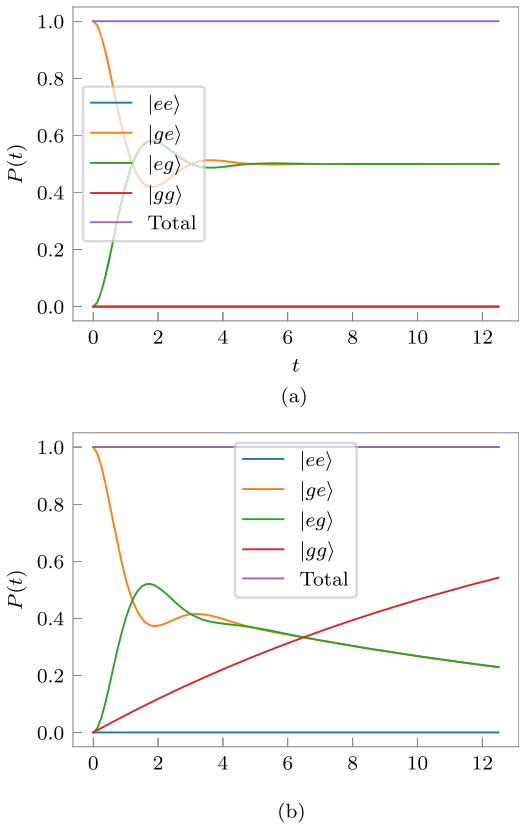


FIG. 3. Simulation of dynamics corresponding to an excitation transfer in a dimer with and without spontaneous emission. (a) Dynamics for `bo` = 0.7071 and `se` = 0.0. (b) Dynamics for `bo` = 0.7071 and `se` = 0.25.

where the proportionality constants have been given as `bo` and `se` for the molecular vibrations and the spontaneous emission processes, respectively. Figure 3 demonstrates the dynamics obtained using this code when spontaneous emission is switched off and on. We see the expected conservation of the number of excitations when spontaneous emission is turned off and a gradual buildup of the population in the ground state in the presence of spontaneous emission.

B. Perturbative and non-perturbative dynamics of open quantum systems

While QuantumDynamics.jl supports empirical methods as described in Sec. III A, the primary focus is on rigorous methods of simulation of open quantum systems. Now, we turn our attention to the more numerically involved methods. For these examples, we will specify the detailed characteristics of the harmonic bath using spectral densities.

1. Bloch–Redfield master equation

The Bloch–Redfield master equation[9,10] (BRME) is one of the simplest and most versatile approaches to perturbatively simulating the dynamics of open quantum systems. It includes a perturbative description of the system-environment interaction, with the environment being described under the Born approximation. Finally, an additional approximation of Markovian dynamics is invoked to obtain BRME. While the combination of approximations involved often makes the method unsuitable for strongly coupled solvents, it is still useful for understanding the very rough timescales of dynamics. Combining BRME with ideas of the polaron transform is successful in extending its applicability to strongly coupled non-perturbative solvents.[78–82]

For the system-solvent Hamiltonian, Eq. (1), under the Born approximation and Markovian limit of the environment, BRME can be expressed as an equation of motion for the reduced density matrix in the eigen-basis of the system Hamiltonian, $\hat{H}_0$,

$$\frac{\mathrm{d}\rho_{ab}}{\mathrm{d}t} = -i\omega_{ab}\rho_{ab}(t) + \sum_{cd} R_{abcd}\rho_{cd}(t), \tag{12}$$

where R_{abcd} is the Redfield tensor that captures the impact of the solvent on the system in a perturbative manner,

$$\begin{aligned} R_{abcd} = -\frac{1}{2}\sum_{k=1}^{N_{\mathrm{env}}} \Big(& \delta_{bd}\sum_n \langle a|\hat{s}_k|n\rangle\langle n|\hat{s}_k|c\rangle J_k(\omega_c - \omega_n) \\ & - \langle a|\hat{s}_k|c\rangle\langle d|\hat{s}_k|b\rangle J_k(\omega_c - \omega_a) \\ & + \delta_{ac}\sum_n \langle d|\hat{s}_k|n\rangle\langle n|\hat{s}_k|b\rangle J_k(\omega_d - \omega_n) \\ & - \langle a|\hat{s}_k|c\rangle\langle d|\hat{s}_k|b\rangle J_k(\omega_d - \omega_b) \Big). \end{aligned} \tag{13}$$

A particular example of the results of BRME for a spin-boson system and its comparison with exact quantum dynamical calculations using QuAPI is shown in Fig. 4. The code for simulating the BRME equations using QuantumDynamics.jl for this particular case is as follows:

```
using QuantumDynamics

# define the system Hamiltonian
H = Matrix{ComplexF64}([
    0.0 -1.0
    -1.0 0.0
])

# specify the spectral density describing the bath and the
↪  inverse temperature
Jw = SpectralDensities.ExponentialCutoff(; ξ=0.1, ωc=7.5)
β = 5.0

ρ0 = Matrix{ComplexF64}([
    1.0 0.0
    0.0 0.0
])
dt = 0.25
ntimes = 100
time, ρs = BlochRedfield.propagate(; Hamiltonian=H,
↪  Jw=[Jw], β, ρ0, dt, ntimes, sys_ops=[[1.0+0.0im 0.0;
↪  0.0 -1.0]])
```

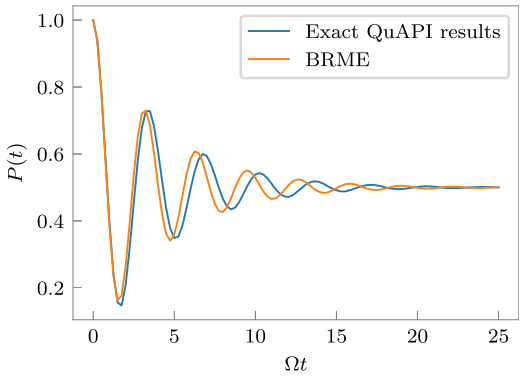


FIG. 4. Comparison between BRME simulation and numerically exact QuAPI calculations. A discussion of QuAPI is given later in Sec. III B 3.

2. Hierarchical equations of motion

The hierarchical equations of motion (HEOM)[14,17,18,83] are one of the two foundational, numerically exact, non-perturbative methods for simulating the dynamics of an open quantum system interacting with a harmonic bath. While originally formulated primarily for the Drude–Lorentz spectral density, recent work has made it possible to use this method with more general spectral densities.[84–87] Other developments have improved the numerical stability of HEOM at lower temperatures.[88] QuantumDynamics.jl supports the scaled version of HEOM[33] for the Drude–Lorentz spectral density. The more advanced approaches required to handle other spectral densities will be incorporated in later versions of the package.

The general problem that HEOM solves is Eq. (1). However, for HEOM, the system–environment interaction Hamiltonian is not exactly Eq. (2). It is given by

$$\hat{H}_{\mathrm{env}}^{(b)} = \sum_j \frac{p_{j,b}^2}{2m_{j,b}} + \frac{1}{2}m_{j,b}\omega_{j,b}^2 x_{j,b}^2 - c_{j,b}\hat{s}_b x_{j,b}. \tag{14}$$

Notice that the difference with Eq. (2) is that here the square is not completed. For the implementation of HEOM in QuantumDynamics.jl, the baths need to be characterized by spectral densities having the Drude–Lorentz form,

$$J_b(\omega) = \frac{2\lambda_b}{\Delta s_b^2}\frac{\gamma_b\omega}{\omega^2 + \gamma_b^2}. \tag{15}$$

The separation between the system states is Δs_b. For problems involving exciton transport, the spectral density is specified using $\Delta s_b = 1$. For application of HEOM, the correlation functions corresponding to the spectral densities are written in a sum over poles form,[33]

$$C_b(t) = \sum_{m=0}^{\infty} c_{bm}\exp(-\nu_{bm}t), \tag{16}$$

where $\nu_{b0} = \gamma_b$ is the Drude decay constant and $\nu_{bm\geq 1} = 2m\pi/\beta$ are the Matsubara frequencies. The coefficients, c_{bm}, are given by

$$c_{b0} = \gamma_b \frac{\lambda_b}{\Delta s_b^2}\left(\cot\left(\frac{\beta\hbar\gamma_b}{2}\right) - i\right), \tag{17}$$

$$c_{bm\geq 1} = \frac{4\lambda_b\gamma_b}{\beta\hbar\Delta s_b^2}\left(\frac{\nu_{bm}}{\nu_{bm}^2 - \gamma_b^2}\right). \tag{18}$$

For such a system, the primary expression for HEOM is given as

$$\begin{aligned}\frac{d\rho_{\mathbf{n}}}{dt} = &-\frac{i}{\hbar}\left[\hat{H}_0, \rho_{\mathbf{n}}\right] + \sum_{b=1}^{N_{\text{env}}}\sum_{m=0}^{M} n_{bm}\nu_{bm}\rho_{\mathbf{n}} \\ &- \sum_{b=1}^{N_{\text{env}}}\left(\frac{2\lambda_b}{\beta\hbar^2\gamma_b} - \sum_{m=0}^{M}\frac{c_{bm}}{\hbar\nu_{bm}}\right)\left[\hat{s}_b, \left[\hat{s}_b, \rho_{\mathbf{n}}\right]\right] - i\sum_{b=1}^{N_{\text{env}}}\left[\hat{s}_b, \sum_{m=0}^{M}\rho_{\mathbf{n}_{bm}^+}\right] \\ &- \frac{i}{\hbar}\sum_{b=1}^{N_{\text{env}}}\sum_{m=0}^{M} n_{bm}\left(c_{bm}\hat{s}_b\rho_{\mathbf{n}_{bm}^-} - c_{bm}^*\rho_{\mathbf{n}_{bm}^-}\hat{s}_b\right),\end{aligned} \tag{19}$$

where $\rho_{\mathbf{n}}$ represents the generalized density operators—when $\mathbf{n} = 0, 0, 0, \ldots$, it is the reduced density operator; for all other $\mathbf{n}$, it is an auxiliary density operator. The subscript vectors, $\mathbf{n}$, are of length $N_{\text{env}}K(M+1)$, where K is the depth of the hierarchy. Each density matrix is assigned a depth of $L = \sum_{b=1}^{N_{\text{env}}}\sum_{m=0}^{M} n_{bm}$. The term in the second line of Eq. (19) is the correction term in the Ishizaki–Tanimura scheme of truncating the Matsubara terms by treating $m > M$ using a Markovian approximation.[15,16]

The scaled version of HEOM[33] rescales the auxiliary density operators in a manner that allows truncation of the hierarchy at a lower value of K,

$$\tilde{\rho}_{\mathbf{n}} = \left(\prod_b\prod_m n_{bm}!|c_{bm}|^{n_{bm}}\right)^{-\frac{1}{2}}\rho_{\mathbf{n}}, \tag{20}$$

which changes Eq. (19) to

$$\begin{aligned}\frac{d\tilde{\rho}_{\mathbf{n}}}{dt} = &-\frac{i}{\hbar}\left[\hat{H}_0, \tilde{\rho}_{\mathbf{n}}\right] + \sum_{b=1}^{N_{\text{env}}}\sum_{m=0}^{M} n_{bm}\nu_{bm}\tilde{\rho}_{\mathbf{n}} \\ &- \sum_{b=1}^{N_{\text{env}}}\left(\frac{2\lambda_b}{\beta\hbar^2\gamma_b} - \sum_{m=0}^{M}\frac{c_{bm}}{\hbar\nu_{bm}}\right)\left[\hat{s}_b, \left[\hat{s}_b, \tilde{\rho}_{\mathbf{n}}\right]\right] \\ &- i\sum_{b=1}^{N_{\text{env}}}\left[\hat{s}_b, \sum_{m=0}^{M}\sqrt{(n_{bm}+1)|c_{bm}|}\tilde{\rho}_{\mathbf{n}_{bm}^+}\right] \\ &- \frac{i}{\hbar}\sum_{b=1}^{N_{\text{env}}}\sum_{m=0}^{M}\sqrt{\frac{n_{bm}}{|c_{bm}|}}\left(c_{bm}\hat{s}_b\tilde{\rho}_{\mathbf{n}_{bm}^-} - c_{bm}^*\tilde{\rho}_{\mathbf{n}_{bm}^-}\hat{s}_b\right).\end{aligned} \tag{21}$$

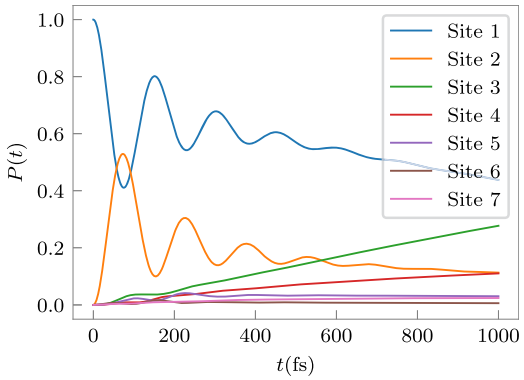
In this new version, Eq. (21), the number of levels of hierarchy required for convergence decreases significantly in comparison to the original unscaled HEOM, Eq. (19). This is the version that is used by default in QuantumDynamics.jl. To use the unscaled version of HEOM, one needs to set `scaled` to `false` while calling the `HEOM.propagate` function.

As a demonstration of the HEOM module in the code, we simulate the dynamics of the chromophoric excitation in the famous seven-state model for the Fenna–Matthews–Olson (FMO) complex[89,90] at $T = 77$ *and* 300 K. The code snippet for this part is quite self-explanatory:

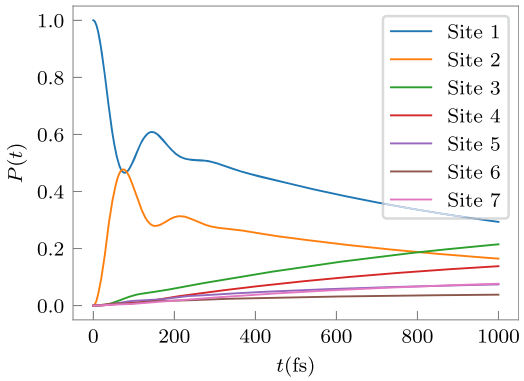
```
using QuantumDynamics

# unit conversion
invcm2au = 4.55633e-6
au2fs = 0.02418884254

function FMO(num_modes, Lmax, β)
    # set up the system Hamiltonian
    H = Matrix{ComplexF64}([
        12410 -87.7   5.5  -5.9   6.7 -13.7  -9.9
        -87.7 12530  30.8   8.2   0.7  11.8   4.3
          5.5  30.8 12210 -53.5  -2.2  -9.6   6.0
         -5.9   8.2 -53.5 12320 -70.7 -17.0 -63.3
          6.7   0.7  -2.2 -70.7 12480  81.1  -1.3
        -13.7  11.8  -9.6 -17.0  81.1 12630  39.7
        -9.9    4.3   6.0 -63.3  -1.3  39.7 12440
    ]) * invcm2au
    nsteps = 500                    # number of steps of
    ↪ simulations
    dt = 1000 / au2fs / nsteps      # dt for simulation

    # initial density matrix with excitation localized on
    ↪ BChl unit 1
    ρ0 = Matrix{ComplexF64}(zeros(7, 7))
    ρ0[1, 1] = 1

    # create spectral densities on each of the BChl sites
    λs = [35.0, 35.0, 35.0, 35.0, 35.0, 35.0, 35.0] *
    ↪ invcm2au
    γs = 1 ./ ([50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0]
    ↪ / au2fs)
    Jw = Vector{SpectralDensities.DrudeLorentz}()
    sys_ops = Vector{Matrix{ComplexF64}}()
    for (j, (λ, γ)) in enumerate(zip(λs, γs))
        push!(Jw, SpectralDensities.DrudeLorentz(; λ, γ,
        ↪ Δs=1.0))

        op = zeros(7, 7)
        op[j, j] = 1.0
        push!(sys_ops, op)
    end

    # simulate the time evolution of ρ0 using HEOM
    t, ρ = HEOM.propagate(; Hamiltonian=H, ρ0=ρ0, Jw, β,
    ↪ ntimes=nsteps, dt, sys_ops, num_modes, Lmax)
    t .*= au2fs
    t, ρ
end
```

Here, we use the `propagate` function under the `HEOM` submodule. It takes a list of spectral densities, `Jw`, along with the corresponding system operators that couple to a particular bath, `sys_ops`. The number of Matsubara modes that are required to converge the results is `num_modes`, and `Lmax` is the number of auxiliary density operators considered in the calculation. At both temperatures, well converged results were obtained with `num_modes` = 2 and `Lmax` = 3. The dynamics obtained using this code is shown in Fig. 5, which matches the original results reported by Ishizaki and Fleming.[89]

As a final example of HEOM, let us consider the case of spontaneous emission that was empirically modeled using the Lindblad Master equation, Fig. 3, in Sec. III A 2. Spontaneous emission happens because of the presence of an environment or bath that is able



(a) $T = 77\,\mathrm{K}$



(b) $T = 300\,\mathrm{K}$

FIG. 5. Excitation population dynamics in the seven-state FMO model introduced by Ishizaki and Fleming.[89]

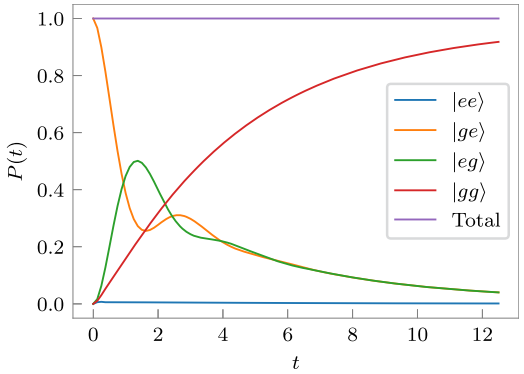


FIG. 6. Dynamics of an excitonic dimer with multiple non-commuting baths.

to couple the molecular excited state to the molecular ground state, thereby reducing the excited state lifetime to some finite value. Once again, the ground and excited states will include the corresponding vibrations and the changes in energy that they bring about. The resultant dynamics is shown in Fig. 6. The bath that enables a spontaneous excitation or relaxation of the molecular eigenstate acts through the system $\hat{\sigma}_x$ operator, whereas the baths representing the vibrational motion on the Born–Oppenheimer surfaces act through $\hat{\sigma}_z$. HEOM is able to handle both a "diagonal" and an "off-diagonal" bath on the same footing without an increase in computational complexity,

```
# define Hamiltonian, initial RDM and simulation parameters
H = Matrix{ComplexF64}([
    20.0 0.0 0.0 0.0
    0.0 10.0 -1.0 0.0
    0.0 -1.0 10.0 0.0
    0.0 0.0 0.0 0.0
])
ρ0 = Matrix{ComplexF64}([
    0.0 0.0 0.0 0.0
    0.0 1.0 0.0 0.0
    0.0 0.0 0.0 0.0
    0.0 0.0 0.0 0.0
])

dt = 0.125
ntimes = 100

# define the spectral density for the vibrational bath
Jw = Vector{SpectralDensities.DrudeLorentz}()
svec = Vector{Matrix{ComplexF64}}()
σz = Matrix{ComplexF64}([
    1.0 0.0
    0.0 -1.0
])
id = Matrix{ComplexF64}([
    1.0 0.0
    0.0 1.0
])
jw1 = SpectralDensities.DrudeLorentz(; λ=bo, γ=5.0, Δs=1.0)
bo1 = kron(σz, id)
bo2 = kron(id, σz)

# define the bath that cause a change in the excitation state of a monomer
σx = Matrix{ComplexF64}([
    0.0 1.0
    1.0 0.0
])
jw3 = SpectralDensities.DrudeLorentz(; λ=se, γ=5.0, Δs=1.0)
se3 = kron(σx, id) + kron(id, σx)

times, ρs = HEOM.propagate(; Hamiltonian=H, ρ0=ρ0, Jw=[jw1, jw1, jw3], β, ntimes, dt, sys_ops=[bo1, bo2, se3], num_modes, Lmax)
```

3. Path integral methods

Path integral approaches form the other numerically exact family of computational methods for simulating the dynamics of open quantum systems as described by Eqs. (1) and (2). The primary restriction in the implementations here is the fact that the system–bath coupling happens through an operator $\hat{s}_b$, which has to be diagonal. While baths coupled through off-diagonal operators can also be simulated at higher costs, the current implementation does not support them. Since the original papers,[11–13] significant developments[20–23,25,27–31,72] have led to the proliferation of methods based on the foundations of path integrals with Feynman–Vernon influence functionals.[19]

Starting from an initial state given as a direct product of the system's reduced density matrix and the thermal distribution of the

environment, the dynamics of the system after N time-steps can be expressed as a path integral,

$$\langle s_N^+|\rho(N\Delta t)|s_N^-\rangle = \sum_{s_0^\pm}\sum_{s_1^\pm}\cdots\sum_{s_{N-1}^\pm}\langle s_N^+|\hat{U}|s_{N-1}^+\rangle\langle s_{N-1}^+|\hat{U}|s_{N-2}^+\rangle\ldots \times\langle s_1^+|\hat{U}|s_0^+\rangle\langle s_0^+|\rho(0)|s_0^-\rangle\langle s_0^-|\hat{U}^\dagger|s_1^-\rangle\ldots \times\langle s_{N-1}^-|\hat{U}|s_N^-\rangle F[\{s_j^\pm\}], \quad (22)$$

where

$$F[\{s_j^\pm\}] = \exp\left(-\frac{1}{\hbar}\sum_{k=0}^{N}(s_k^+ - s_k^-)\sum_{k'=0}^{k}(\eta_{kk'}s_{k'}^+ - \eta_{kk'}^* s_{k'}^-)\right). \quad (23)$$

Here, $\hat{U}$ is the bare system propagator, and F is the Feynman–Vernon influence functional corresponding to the forward–backward path $s_j^\pm$. The influence functional for a system coupled to multiple environments is given as a product of the influence functionals corresponding to the individual environments. The cost of simulations does not increase as long as all the operators in the environments commute with each other. The bath response function is discretized into η-coefficients.[11] The non-Markovian nature of the dynamics is brought about by the dependence of the influence functional on the full path of the system. However, in condensed phases, the memory decays with the time difference between the interacting points. Thus, after a full-memory simulation of L time steps, which is a convergence parameter, one can use an iterative algorithm to propagate the reduced density matrix further out in time. The summand of the right-hand side of Eq. (22) can be thought of as a tensor indexed by the forward–backward system paths, called the path amplitude tensor.

Various approaches have been used to reduce the computational complexity of the problem, which naïvely grows as $\mathcal{O}(d^{2L})$ for the original QuAPI algorithm, where d is the system dimensionality and L is the memory length. Other approaches attempt to decrease these exponentially growing computational and storage requirements. The blip decomposition[20,21] of the path integral uses the fact that the influence functional, Eq. (23), depends on the value of $\Delta s = s^+ - s^-$ for the latter point. That means that for all the paths with no time-point where $\Delta s = 0$, the influence functional is one. Thus, this set of paths can be summed up in a Markovian manner. In fact, any segment of the path that consists solely of points with $\Delta s = 0$, or "sojourns," can be summed up through iterative matrix-vector multiplications, thereby reducing the effective number of paths that need to be considered. This blip decomposition is especially lucrative for slow and strongly coupled solvents.

Recently, tensor networks have been used in a variety of ways to reduce the complexity of these path integral calculations. Most prominent of these is the time-evolved matrix product operators approach[31] (TEMPO), which uses a matrix product state to give a compact representation of the path amplitude tensor utilizing the decaying correlation between indices with large separation. Under the tensor network path integral[25] (TNPI) implementation of the TEMPO algorithm, it has been shown that the influence functional for multiple baths can be analytically represented in the form of an optimal matrix product operator.

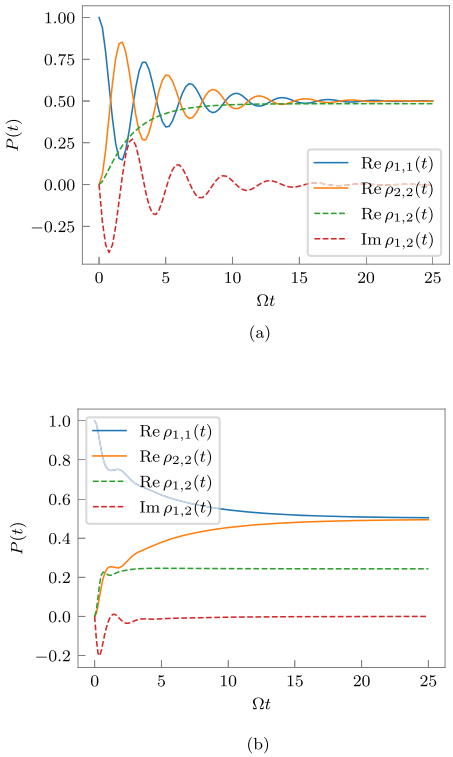


FIG. 7. Example of dynamics using QuAPI-related methods simulated using QuantumDynamics.jl. (a) $\xi = 0.1$, $\omega_c = 7.5\Omega$, $\hbar\Omega\beta = 5$, $L = 6$, and $\Delta t = 0.25$. The parameters from Ref. 11. Run with `method = QuAPI.propagate`. (b) $\xi = 2.0$, $\omega_c = \Omega$, $\hbar\Omega\beta = 1$, $L = 150$, and $\Delta t = 0.125$. The parameters from Refs. 72 and 91. Run with `method = TEMPO.propagate`.

Additionally, PC-TNPI is a new tensor network that has been designed to manifestly capture the symmetries present in the influence functional.[22]

There are four basic modules of path integral simulations that are supported—`QuAPI` implementing ideas in Refs. 11 and 12, `Blip` implementing Ref. 20, `TEMPO` implementing Ref. 31, and `PCTNPI` implementing Ref. 22. In principle iterative propagation of reduced density matrices beyond the memory time is possible in all of these methods; however, based on our experience TEMPO gives the greatest ability to access long memory lengths and large systems. Thus, iterative propagation is implemented only in `TEMPO` and in the base `QuAPI`. All the modules support the creation of augmented propagators, which are the effective propagators of the system in the presence of the solvent.

First, we demonstrate the QuantumDynamics.jl code both for the most fundamental path integral method, QuAPI,[11,12] and for TEMPO.[31] Consider the symmetric system: $\hat{H}_0 = -\hbar\Omega\sigma_x$ coupled with a bath of harmonic oscillators characterized by an Ohmic spectral density where ξ is the dimensionless Kondo parameter and ω_c is the cutoff frequency. The simulation with any of the methods will have the following outline:

```
# define the system Hamiltonian
H = Matrix{ComplexF64}([
    0.0 -1.0
    -1.0 0.0
])

# calculate the sequence of bare propagators with a given
↪  time-step
barefbU = Propagators.calculate_bare_propagators(;
↪  Hamiltonian=H, dt, ntimes)

# define the initial condition and the spectral density
ρ0 = [1.0+0.0im 0.0; 0.0 0.0]
Jw = SpectralDensities.ExponentialCutoff(; ξ, ωc)

# use a given method to propagate the initial state in
↪  presence of the environment at an inverse temperature,
↪  β.
# method is one of QuAPI.propagate or TEMPO.propagate
t, ρs = method(; fbU=barefbU, Jw=[Jw], β, ρ0, dt, ntimes,
↪  L, svec)
```

The `method` is currently one of `QuAPI.propagate` or `TEMPO.propagate`. These propagate methods take custom extra arguments that specify how to tune the algorithms in specific ways to improve performance.

As an illustration, we demonstrate two different parameters using base QuAPI [Fig. 7(a)] and using TEMPO [Fig. 7(b)]. For the example simulated using QuAPI, we use a parameter that was introduced in Ref. 11. For the example that we simulated using TEMPO, we chose a parameter that was originally simulated using a quantum–classical path integral up to a short time[91] and more recently using SMatPI until equilibration.[72] In this case, the bath is localized around the initial system state.

For problems where the iterative portion of the dynamics is significantly longer than the full-memory portion, the cost of the iteration, which is proportional to the number of paths, adds up. One way of solving this is to use TTM[71] to reduce the cost to that of a "convolution" of these transfer tensors and the augmented propagators. TTM uses the other base path integral methods to generate the propagators for some number of time-steps and then uses them to calculate the propagators further out in time. We demonstrate the use of TTM using the strong excitation energy transfer (EET) dimer from Ref. 92. The structure of a code using TTM is shown below:

```
# define the EET dimer Hamiltonian
H = Matrix{ComplexF64}([
    50 100.0
    100.0 -50
] .* invcm2au)
# using a time-step, dt, calculate the bare
↪  forward-backward propagators for ntimes time-steps
fbU = Propagators.calculate_bare_propagators(;
↪  Hamiltonian=H, ntimes, dt)

# define the spectral densities acting on the different
↪  monomers
J1 = SpectralDensities.DrudeLorentz(; λ=λ * invcm2au, γ=γ
↪  * invcm2au, Δs=1.0)
J2 = SpectralDensities.DrudeLorentz(; λ=λ * invcm2au, γ=γ
↪  * invcm2au, Δs=1.0)
svec = [1.0 0.0; 0.0 1.0]
β = 1052.0

ρ0 = [1.0+0.0im 0.0; 0.0 0.0]
# TTM.propagate takes a particular path_integral_routine
↪  and the corresponding extraargs.
times, ρs = TTM.propagate(; fbU=fbU, ρ0=ρ0, Jw=[J1, J2],
↪  β, ntimes, dt, svec, rmax=rmax,
↪  extraargs=TEMPO.TEMPOArgs(; cutoff=1e-13,
↪  maxdim=10000), path_integral_routine =
↪  TEMPO.build_augmented_propagator, verbose=true)
```

These calculations were done with full memory simulations of 75 steps with a time-step of $\Delta t = 4.84$ fs. The spectral density used is the Drude–Lorentz spectral density, Eq. (5), with $\gamma = 53.08\ \mathrm{cm}^{-1}$. The results are shown for two different reorganization energies, λ, in Fig. 8.

Finally, the last major method supported by QuantumDynamics.jl is QCPI[67,68] with reference propagators[69] and harmonic backreaction.[70] The incorporation of classical trajectories not only allows for larger time-steps but also reduces the effective memory that needs to be accounted for through path integrals by incorporating the classical part of the memory completely.[69] A simulation that only incorporates the classical part of the memory is called the ensemble average classical path (EACP) simulation. With reference propagators, one can do this simulation in a Markovian manner. Currently, the support is only for a harmonic bath, although the

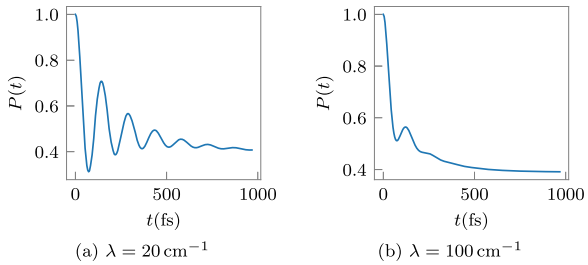


FIG. 8. Simulation of an excitation energy transfer dimer with parameters obtained from Ref. 92. $\epsilon = 100\ \mathrm{cm}^{-1}$, $\Omega = -100\ \mathrm{cm}^{-1}$.

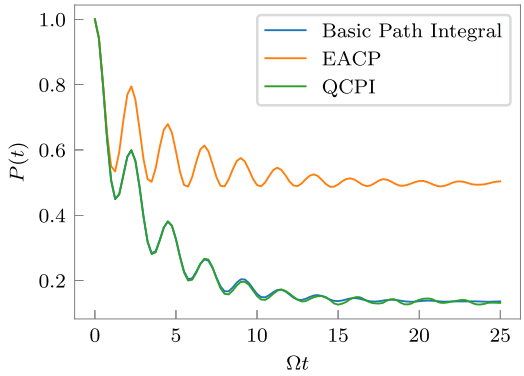


FIG. 9. Comparison between QuAPI and QCPI runs for the parameters shown in the code. 10 000 initial conditions were used for the EACP and QCPI calculations.

infrastructure is built in such a manner that it is trivial to extend it to include anharmonic solvents in the reference propagators either by solving the equations of motion using DifferentialEquations.jl,[73] or by coupling it with a molecular dynamics framework such as Molly.jl and the Atomic Simulation Engine[93] (ASE). Due to the modular nature of QuantumDynamics.jl, these different classical trajectory backends will work in a plug-and-play manner.

Below is a code snippet that does both the EACP calculation and a full QCPI calculation on a sample spin-boson parameter,

```
# specify the system Hamiltonian
H0 = Matrix{ComplexF64}([
    1.0 -1.0
    -1.0 -1.0
])

# specify the spectral density and the inverse temperature
Jw = SpectralDensities.ExponentialCutoff(; ξ=0.1, ωc=7.5)
β = 5.0

ρ0 = Matrix{ComplexF64}([
    1.0 0.0
    0.0 0.0
])

dt = 0.25
ntimes = 100

# discretize the spectral density and create a harmonic
↪  bath solvent
# for an atomistic solvent, here we would use the actual
↪  description based on an appropriate force field or ab
↪  initio DFT calculation
ω, c = SpectralDensities.discretize(Jw, 100)
hb = Solvents.HarmonicBath(β, ω, c, [1.0, -1.0],
↪  num_points)

# calculate EACP dynamics
EACP_fbU =
↪  Propagators.calculate_average_reference_propagators(;
↪  Hamiltonian=H0, solvent=hb, classical_dt=dt / 100,
↪  dt, ntimes)
times_EACP, ρs_EACP = Utilities.apply_propagator(;
↪  propagators=EACP_fbU, ρ0, ntimes, dt)

# simulate QCPI
times_QCPI, ρs_QCPI = QCPI.propagate(; Hamiltonian=H0,
↪  Jw, solvent=hb, ρ0, classical_dt=dt / 100, dt,
↪  ntimes, kmax=3, extraargs=QuAPI.QuAPIArgs(),
↪  path_integral_routine=QuAPI.propagate)
```

QuantumDynamics.jl does not enforce any parallelization over the Monte Carlo runs, binning, and calculation of error statistics. That is left to the end user to implement in a manner suited to the problem being studied. It will be quite simple to spread `QCPI.propagate` calls over multiple nodes and aggregate across them using a message-passing interface. The dynamics obtained with 10 000 initial conditions are demonstrated in Fig. 9.

IV. CONCLUSION

In this paper, we have introduced a new package called QuantumDynamics.jl for simulations of non-adiabatic processes. The Julia programming language has been emerging as a promising candidate for modern high-level, high-performance scientific computing, with a growing base of packages for computational chemistry and physics. Being written in Julia allows QuantumDynamics.jl to take advantage of packages such as DifferentialEquations.jl for solving differential equations. This also allows us to avoid the "two-language" problem, where the performance critical parts need to be implemented in some lower-level high-performance language.

Simulating the dynamics of quantum systems interacting with their environments is often very difficult if done in a numerically exact manner. The exact methods are quite involved from a theoretical perspective while being challenging to implement in code. They are built on top of a variety of deep insights into the structure and dynamics of these systems. Very few open-source packages exist that aim to make these methods accessible to non-specialists while providing a platform for specialists that encourages exploration and further theoretical development. Inspired by the objectives behind PySCF,[40] QuantumDynamics.jl was designed to address this particular problem. It joins the recently growing ranks of computational packages for chemistry in the Julia programming language.[53,55,56]

QuantumDynamics.jl already supports a variety of methods. On the empirical and perturbative ends, methods such as the propagation of non-Hermitian Hamiltonians, the Lindblad master equation, and the perturbative Bloch–Redfield master equation are all built on top of the backend provided in DifferentialEquations.jl. The BRME can later be extended using polaron and variational polaron transformed approaches to increase the applicability of the perturbative ideas. In terms of numerically exact approaches, both HEOM-based and QuAPI-based methods are supported. In HEOM, we have already implemented the unscaled and scaled versions. The use of matrix product states and other approaches to generalizing them to account for non-Drude–Lorentz spectral densities will be implemented in the near future.

The largest set of methods implemented in QuantumDynamics.jl fall into the category of path integral- or QuAPI-based approaches. The base methods of QuAPI, blip decomposition, TEMPO, and PC-TNPI are all supported. QuAPI and TEMPO support the propagation of density matrices, while blips and PC-TNPI are currently only capable of producing augmented forward–backward propagators. While this does not hamper the usability of these methods in conjunction with TTM, this deficiency will be remedied in a future version. Probably the single most useful sub-module of the path integral methods is TEMPO. Given its ability to handle comparatively large systems with long memories, it is exceptionally powerful. The TNPI-based implementation allows the use of multiple baths in an optimal manner. The compatibility of all of these methods with TTM is a very useful feature of QuantumDynamics.jl.

The goal is to provide the community with a platform that is fit for exploration and method development, in addition to a repository of methods that can be directly used for accurate simulations of quantum dynamics. There are many other developments that are yet to be incorporated in QuantumDynamics.jl. A notable example is the recently developed multisite decomposition of the tensor network path integral[23] (MS-TNPI), which combines ideas from time-dependent density matrix renormalization group[5,6,94–96] with the Feynman–Vernon influence functional in order to make simulations of extended open quantum systems feasible.[3,97] While we will introduce some methods such as MS-TNPI[23] in the near future and continue to develop into this package, we hope that QuantumDynamics.jl becomes a toolbox for the community with others actively using and developing it as well.

AUTHOR DECLARATIONS

Conflict of Interest

The author has no conflicts to disclose.

Author Contributions

Amartya Bose: Conceptualization (equal); Data curation (equal); Formal analysis (equal); Funding acquisition (equal); Investigation (equal); Methodology (equal); Resources (equal); Software (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

APPENDIX: COMPARISON OF METHODS

To demonstrate the accuracy of the implementations and the realms of validity of the various methods, we apply the perturbative and non-perturbative methods to a common set of spin-boson parameters, where the asymmetric system is described by

$$\hat{H}_0 = \epsilon\hat{\sigma}_z - \hbar\Omega\hat{\sigma}_x, \tag{A1}$$

where $\hat{\sigma}_z$ and $\hat{\sigma}_x$ are the Pauli spin matrices, $\epsilon = 1$ and $\Omega = 1$, respectively. While BRME and the path integral methods can handle arbitrary spectral densities, the HEOM implementation is currently unable to handle anything but the Ohmic spectral density with a Drude–Lorentz cutoff. Therefore, to keep the comparison fair, we will use a Drude–Lorentz spectral density as defined in Eq. (5). Both HEOM and the path integral-based methods are numerically exact.

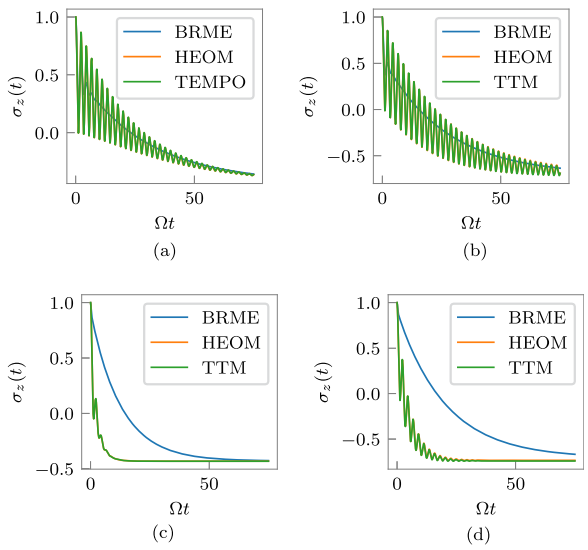


FIG. 10. Comparison between different methods for the Drude–Lorentz spectral density. (a) $\hbar\Omega\beta = 0.5$ and $\lambda = 0.2$. (b) $\hbar\Omega\beta = 5$ and $\lambda = 0.2$. (c) $\hbar\Omega\beta = 0.5$ and $\lambda = 1$. (d) $\hbar\Omega\beta = 5$ and $\lambda = 1$.

Therefore, under convergence, they should give identical results. They may have different costs depending on the exact parameters. However, BRME is an approximation and, consequently, is not guaranteed to give the correct results for all parameters.

In these examples, we will evaluate the accuracy of dynamics and the long-time limit of $\langle\hat{\sigma}_z(t)\rangle$ as obtained by BRME, and the costs associated with the numerically exact methods. The correctness of the implementations of the exact methods will be validated by showing that both HEOM and path integrals give identical results under convergence. For these tests, we will use TEMPO with TTM as a representative path integral method. In the online documentation, we demonstrate the equivalence of all the path integral methods in terms of their converged results. The costs associated with them obviously vary and the more recent methods make simulations of larger systems with longer memory more cost effective.

Figure 10 shows the comparison between the different methods of simulation at different temperatures and reorganization energies for the Drude–Lorentz spectral densities. BRME is able to get the long-time value of $\langle\hat{\sigma}_z(t)\rangle$ at high temperatures and low reorganization energies. Generally, for spin-boson problems, the perturbation theory-based BRME does not work well by itself. In these examples, it seems that BRME washes away the oscillatory features of the dynamics. Relatively recent work has shown that variational polaron transfers coupled with BRME are a significantly better approximation.[79]

HEOM and path integrals are both exact methods that converge to the same result. The cost of the calculations and the parameters at which they converge are very different. For example, HEOM requires fewer Matsubara modes at higher temperatures and, thus, the computations become simpler. In both the high-temperature examples, the HEOM results converged with a single extra Matsubara mode, whereas for the low temperatures, convergence was achieved at ≈8 Matsubara modes. The number of levels of hierarchy that need to be retained is related to the non-Markovian memory that is incorporated. In contrast, the computations with path integral methods involve parameters that are very different. The major parameters are the number of steps in the non-Markovian memory and the size of the time step. For these calculations, a memory span of $L\Delta t\Omega = 2.5$ yielded convergent results. Various other methods of filtering the path list are, of course, used. The fact that both of these methods give the same results provides an independent check of the correctness of both codes.

REFERENCES

[1] S. Maity and U. Kleinekathöfer, "Recent progress in atomistic modeling of light-harvesting complexes: A mini review," Photosynth. Res. **156**, 147 (2022).
[2] J. Cao, R. J. Cogdell, D. F. Coker, H.-G. Duan, J. Hauer, U. Kleinekathöfer, TT. L. C. Jansen, T. Mančal, R. J. D. Miller, J. P. Ogilvie, V. I. Prokhorenko, T. Renger, H.-S. Tan, R. Tempelaar, M. Thorwart, E. Thyrhaug, S. Westenhoff, and D. Zigmantas, "Quantum biology revisited," Sci. Adv. **6**, eaaz4888 (2020).
[3] A. Bose and P. L. Walters, "Tensor network path integral study of dynamics in B850 LH2 ring with atomistically derived vibrations," J. Chem. Theory Comput. **18**, 4095–4108 (2022).
[4] C. Olbrich, J. Strümpfer, K. Schulten, and U. Kleinekathöfer, "Theory and simulation of the environmental effects on FMO electronic transitions," J. Phys. Chem. Lett. **2**, 1771–1776 (2011).

[5]S. Paeckel, T. Köhler, A. Swoboda, S. R. Manmana, U. Schollwöck, and C. Hubig, "Time-evolution methods for matrix-product states," Ann. Phys. **411**, 167998 (2019).
[6]S. R. White and A. E. Feiguin, "Real-time evolution using the density matrix renormalization group," Phys. Rev. Lett. **93**, 076401 (2004).
[7]M. V. Rakov and M. Weyrauch, "Spin-$\frac{1}{2}$ XXZ Heisenberg chain in a longitudinal magnetic field," Phys. Rev. B **100**, 134434 (2019).
[8]T. Macrì, L. Lepori, G. Pagano, M. Lewenstein, and L. Barbiero, "Bound state dynamics in the long-range spin-$\frac{1}{2}$ XXZ model," Phys. Rev. B **104**, 214309 (2021).
[9]F. Bloch, "Generalized theory of relaxation," Phys. Rev. **105**, 1206–1222 (1957).
[10]A. G. Redfield, "On the theory of relaxation processes," IBM J. Res. Dev. **1**, 19–31 (1957).
[11]N. Makri and D. E. Makarov, "Tensor propagator for iterative quantum time evolution of reduced density matrices. I. Theory," J. Chem. Phys. **102**, 4600–4610 (1995).
[12]N. Makri and D. E. Makarov, "Tensor propagator for iterative quantum time evolution of reduced density matrices. II. Numerical methodology," J. Chem. Phys. **102**, 4611–4618 (1995).
[13]N. Makri, "Numerical path integral techniques for long time dynamics of quantum dissipative systems," J. Math. Phys. **36**, 2430–2457 (1995).
[14]Y. Tanimura and R. Kubo, "Time evolution of a quantum system in contact with a nearly Gaussian–Markoffian noise bath," J. Phys. Soc. Jpn. **58**, 101–114 (1989).
[15]A. Ishizaki and Y. Tanimura, "Quantum dynamics of system strongly coupled to low-temperature colored noise bath: Reduced hierarchy equations approach," J. Phys. Soc. Jpn. **74**, 3131–3134 (2005).
[16]Y. Tanimura, "Stochastic Liouville, Langevin, Fokker–Planck, and master equation approaches to quantum dissipative systems," J. Phys. Soc. Jpn. **75**, 082001 (2006).
[17]Y. Tanimura, "Reduced hierarchical equations of motion in real and imaginary time: Correlated initial states and thermodynamic quantities," J. Chem. Phys. **141**, 044114 (2014).
[18]Y. Tanimura, "Numerically 'exact' approach to open quantum dynamics: The hierarchical equations of motion (HEOM)," J. Chem. Phys. **153**, 020901 (2020).
[19]R. P. Feynman and F. L. Vernon, "The theory of a general quantum system interacting with a linear dissipative system," Ann. Phys. **24**, 118–173 (1963).
[20]N. Makri, "Blip decomposition of the path integral: Exponential acceleration of real-time calculations on quantum dissipative systems," J. Chem. Phys. **141**, 134117 (2014).
[21]N. Makri, "Iterative blip-summed path integral for quantum dynamics in strongly dissipative environments," J. Chem. Phys. **146**, 134101 (2017).
[22]A. Bose, "Pairwise connected tensor network representation of path integrals," Phys. Rev. B **105**, 024309 (2022).
[23]A. Bose and P. L. Walters, "A multisite decomposition of the tensor network path integrals," J. Chem. Phys. **156**, 024101 (2022).
[24]N. Makri, "Modular path integral methodology for real-time quantum dynamics," J. Chem. Phys. **149**, 214108 (2018).
[25]A. Bose and P. L. Walters, "A tensor network representation of path integrals: Implementation and analysis," arXiv:2106.12523 (2021).
[26]M. R. Jørgensen and F. A. Pollock, "Exploiting the causal tensor network structure of quantum processes to efficiently simulate non-Markovian path integrals," Phys. Rev. Lett. **123**, 240602 (2019).
[27]N. Makri, "Small matrix disentanglement of the path integral: Overcoming the exponential tensor scaling with memory length," J. Chem. Phys. **152**, 041104 (2020).
[28]N. Makri, "Small matrix path integral for system-bath dynamics," J. Chem. Theory Comput. **16**, 4038–4049 (2020).
[29]N. Makri, "Small matrix modular path integral: Iterative quantum dynamics in space and time," Phys. Chem. Chem. Phys. **23**, 12537–12540 (2021).
[30]N. Makri, "Small matrix path integral for driven dissipative dynamics," J. Phys. Chem. A **125**, 10500–10506 (2021).
[31]A. Strathearn, P. Kirton, D. Kilda, J. Keeling, and B. W. Lovett, "Efficient non-Markovian quantum dynamics using time-evolving matrix product operators," Nat. Commun. **9**, 3322 (2018).
[32]J. Hu, M. Luo, F. Jiang, R.-X. Xu, and Y. Yan, "Padé spectrum decompositions of quantum distribution functions and optimal hierarchical equations of motion construction for quantum open systems," J. Chem. Phys. **134**, 244106 (2011).
[33]Q. Shi, L. Chen, G. Nan, R.-X. Xu, and Y. Yan, "Efficient hierarchical Liouville space propagator to quantum dissipative dynamics," J. Chem. Phys. **130**, 084105 (2009).
[34]Q. Shi, Y. Xu, Y. Yan, and M. Xu, "Efficient propagation of the hierarchical equations of motion using the matrix product state method," J. Chem. Phys. **148**, 174102 (2018).
[35]Y. Yan, M. Xu, T. Li, and Q. Shi, "Efficient propagation of the hierarchical equations of motion using the Tucker and hierarchical Tucker tensors," J. Chem. Phys. **154**, 194104 (2021).
[36]T. Ikeda and G. D. Scholes, "Generalization of the hierarchical equations of motion theory for efficient calculations with arbitrary correlation functions," J. Chem. Phys. **152**, 204101 (2020).
[37]E. Aprà, E. J. Bylaska, W. A. de Jong, N. Govind, K. Kowalski, T. P. Straatsma, M. Valiev, H. J. J. van Dam, Y. Alexeev, J. Anchell, V. Anisimov, F. W. Aquino, R. Atta-Fynn, J. Autschbach, N. P. Bauman, J. C. Becca, D. E. Bernholdt, K. Bhaskaran-Nair, S. Bogatko, P. Borowski, J. Boschen, J. Brabec, A. Bruner, E. Cauët, Y. Chen, G. N. Chuev, C. J. Cramer, J. Daily, M. J. O. Deegan, T. H. Dunning, M. Dupuis, K. G. Dyall, G. I. Fann, S. A. Fischer, A. Fonari, H. Früchtl, L. Gagliardi, J. Garza, N. Gawande, S. Ghosh, K. Glaesemann, A. W. Götz, J. Hammond, V. Helms, E. D. Hermes, K. Hirao, S. Hirata, M. Jacquelin, L. Jensen, B. G. Johnson, H. Jónsson, R. A. Kendall, M. Klemm, R. Kobayashi, V. Konkov, S. Krishnamoorthy, M. Krishnan, Z. Lin, R. D. Lins, R. J. Littlefield, A. J. Logsdail, K. Lopata, W. Ma, A. V. Marenich, J. Martin del Campo, D. Mejia-Rodriguez, J. E. Moore, J. M. Mullin, T. Nakajima, D. R. Nascimento, J. A. Nichols, P. J. Nichols, J. Nieplocha, A. Otero-de-la-Roza, B. Palmer, A. Panyala, T. Pirojsirikul, B. Peng, R. Peverati, J. Pittner, L. Pollack, R. M. Richard, P. Sadayappan, G. C. Schatz, W. A. Shelton, D. W. Silverstein, D. M. A. Smith, T. A. Soares, D. Song, M. Swart, H. L. Taylor, G. S. Thomas, V. Tipparaju, D. G. Truhlar, K. Tsemekhman, T. Van Voorhis, Á. Vázquez-Mayagoitia, P. Verma, O. Villa, A. Vishnu, K. D. Vogiatzis, D. Wang, J. H. Weare, M. J. Williamson, T. L. Windus, K. Woliński, A. T. Wong, Q. Wu, C. Yang, Q. Yu, M. Zacharias, Z. Zhang, Y. Zhao, and R. J. Harrison, "NWChem: Past, present, and future," J. Chem. Phys. **152**, 184102 (2020).
[38]M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, G. A. Petersson, H. Nakatsuji, X. Li, M. Caricato, A. V. Marenich, J. Bloino, B. G. Janesko, R. Gomperts, B. Mennucci, H. P. Hratchian, J. V. Ortiz, A. F. Izmaylov, J. L. Sonnenberg, Williams, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V. G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. J. Bearpark, J. J. Heyd, E. N. Brothers, K. N. Kudin, V. N. Staroverov, T. A. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. P. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, J. M. Millam, M. Klene, C. Adamo, R. Cammi, J. W. Ochterski, R. L. Martin, K. Morokuma, O. Farkas, J. B. Foresman, and D. J. Fox, GAUSSIAN 16 Rev. C.01, 2016.
[39]D. G. A. Smith, L. A. Burns, A. C. Simmonett, R. M. Parrish, M. C. Schieber, R. Galvelis, P. Kraus, H. Kruse, R. Di Remigio, A. Alenaizan, A. M. James, S. Lehtola, J. P. Misiewicz, M. Scheurer, R. A. Shaw, J. B. Schriber, Y. Xie, Z. L. Glick, D. A. Sirianni, J. S. O'Brien, J. M. Waldrop, A. Kumar, E. G. Hohenstein, B. P. Pritchard, B. R. Brooks, H. F. Schaefer, A. Y. Sokolov, K. Patkowski, A. E. DePrince, U. Bozkaya, R. A. King, F. A. Evangelista, J. M. Turney, T. D. Crawford, and C. D. Sherrill, "PSI4 1.4: Open-source software for high-throughput quantum chemistry," J. Chem. Phys. **152**, 184108 (2020).
[40]Q. Sun, T. C. Berkelbach, N. S. Blunt, G. H. Booth, S. Guo, Z. Li, J. Liu, J. D. McClain, E. R. Sayfutyarova, S. Sharma, S. Wouters, and G. K.-L. Chan, "PySCF: The Python-based simulations of chemistry framework," WIREs Comput. Mol. Sci. **8**, e1340 (2018).
[41]S. G. Balasubramani, G. P. Chen, S. Coriani, M. Diedenhofen, M. S. Frank, Y. J. Franzke, F. Furche, R. Grotjahn, M. E. Harding, C. Hättig, A. Hellweg, B. Helmich-Paris, C. Holzer, U. Huniar, M. Kaupp, A. Marefat Khah, S. Karbalaei Khani, T. Müller, F. Mack, B. D. Nguyen, S. M. Parker, E. Perlt, D. Rappoport, K. Reiter, S. Roy, M. Rückert, G. Schmitz, M. Sierka, E. Tapavicza, D. P. Tew, C. van Wüllen, V.

K. Voora, F. Weigend, A. Wodyński, and J. M. Yu, "TURBOMOLE: Modular program suite for *ab initio* quantum-chemical and condensed-matter simulations," J. Chem. Phys. **152**, 184107 (2020).
[42]T. D. Kühne, M. Iannuzzi, M. Del Ben, V. V. Rybkin, P. Seewald, F. Stein, T. Laino, R. Z. Khaliullin, O. Schütt, F. Schiffmann, D. Golze, J. Wilhelm, S. Chulkov, M. H. Bani-Hashemian, V. Weber, U. Borštnik, M. Taillefumier, A. S. Jakobovits, A. Lazzaro, H. Pabst, T. Müller, R. Schade, M. Guidon, S. Andermatt, N. Holmberg, G. K. Schenter, A. Hehn, A. Bussy, F. Belleflamme, G. Tabacchi, A. Glöß, M. Lass, I. Bethune, C. J. Mundy, C. Plessl, M. Watkins, J. Vandevondele, M. Krack, and J. Hutter, "CP2K: An electronic structure and molecular dynamics software package—Quickstep: Efficient and accurate electronic structure calculations," J. Chem. Phys. **152**, 194103 (2020).
[43]P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, and R. M. Wentzcovitch, "QUANTUM ESPRESSO: A modular and open-source software project for quantum simulations of materials," J. Phys.: Condens. Matter **21**, 395502 (2009).
[44]P. Giannozzi, O. Andreussi, T. Brumme, O. Bunau, M. Buongiorno Nardelli, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, M. Cococcioni, N. Colonna, I. Carnimeo, A. Dal Corso, S. de Gironcoli, P. Delugas, R. A. DiStasio, A. Ferretti, A. Floris, G. Fratesi, G. Fugallo, R. Gebauer, U. Gerstmann, F. Giustino, T. Gorni, J. Jia, M. Kawamura, H.-Y. Ko, A. Kokalj, E. Küçükbenli, M. Lazzeri, M. Marsili, N. Marzari, F. Mauri, N. L. Nguyen, H.-V. Nguyen, A. Otero-de-la-Roza, L. Paulatto, S. Poncé, D. Rocca, R. Sabatini, B. Santra, M. Schlipf, A. P. Seitsonen, A. Smogunov, I. Timrov, T. Thonhauser, P. Umari, N. Vast, X. Wu, and S. Baroni, "Advanced capabilities for materials modelling with QUANTUM ESPRESSO," J. Phys.: Condens. Matter **29**, 465901 (2017).
[45]V. Kapil, M. Rossi, O. Marsalek, R. Petraglia, Y. Litman, T. Spura, B. Cheng, A. Cuzzocrea, R. H. Meißner, D. M. Wilkins, B. A. Helfrecht, P. Juda, S. P. Bienvenue, W. Fang, J. Kessler, I. Poltavsky, S. Vandenbrande, J. Wieme, C. Corminboeuf, T. D. Kühne, D. E. Manolopoulos, T. E. Markland, J. O. Richardson, A. Tkatchenko, G. A. Tribello, V. Van Speybroeck, and M. Ceriotti, "i-PI 2.0: A universal force engine for advanced molecular simulations," Comput. Phys. Commun. **236**, 214–223 (2019).
[46]T. Ikeda and Y. Tanimura, "Low-temperature quantum Fokker–Planck and Smoluchowski equations and their extension to multistate systems," J. Chem. Theory Comput. **15**, 2517–2534 (2019).
[47]C. Kreisbeck and T. Kramer, "Exciton dynamics lab for light-harvesting complexes (GPU-HEOM)," See nanohub.org for electronic tool, https://10.4231/D3RB6W248, 2013.
[48]J. Strümpfer and K. Schulten, "Open quantum dynamics calculations with the hierarchy equations of motion on parallel computers," J. Chem. Theory Comput. **8**, 2808–2816 (2012).
[49]M. Tsuchimoto and Y. Tanimura, "Spins dynamics in a dissipative environment: Hierarchal equations of motion approach using a graphics processing unit (GPU)," J. Chem. Theory Comput. **11**, 3859–3865 (2015).
[50]S. Temen, A. Jain, and A. V. Akimov, "Hierarchical equations of motion in the Libra software package," Int. J. Quantum Chem. **120**, e26373 (2020).
[51]J. R. Johansson, P. D. Nation, and F. Nori, "QuTiP 2: A Python framework for the dynamics of open quantum systems," Comput. Phys. Commun. **184**, 1234–1240 (2013).
[52]A. V. Akimov, "Libra: An open-source 'methodology discovery' library for quantum and classical dynamics simulations," J. Comput. Chem. **37**, 1626–1649 (2016).
[53]J. Gardner, O. A. Douglas-Gallardo, W. G. Stark, J. Westermayr, S. M. Janke, S. Habershon, and R. J. Maurer, "NQCDynamics.jl: A Julia package for nonadiabatic quantum classical molecular dynamics in the condensed phase," J. Chem. Phys. **156**, 174801 (2022).
[54]J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," SIAM Rev. **59**, 65–98 (2017).
[55]G. J. R. Aroeira, M. M. Davis, J. M. Turney, and H. F. Schaefer, "Fermi.jl: A modern design for quantum chemistry," J. Chem. Theory Comput. **18**, 677–686 (2022).
[56]M. F. Herbst, A. Levitt, and E. Cancès, "DFTK: A Julian approach for simulating electrons in solids," in *Proceedings of the JuliaCon Conferences* (2021), Vol. 3, p. 69.
[57]S. Krämer, D. Plankensteiner, L. Ostermann, and H. Ritsch, "QuantumOptics.jl: A Julia framework for simulating open quantum systems," Comput. Phys. Commun. **227**, 109–116 (2018).
[58]D. Poole, J. L. Galvez Vallejo, and M. S. Gordon, "A new kid on the block: Application of Julia to Hartree–Fock calculations," J. Chem. Theory Comput. **16**, 5006–5013 (2020).
[59]D. Poole, J. L. Galvez Vallejo, and M. S. Gordon, "A task-based approach to parallel restricted Hartree–Fock calculations," J. Chem. Theory Comput. **18**, 2144–2161 (2022).
[60]N. Makri, "The linear response approximation and its lowest order corrections: An influence functional approach," J. Phys. Chem. B **103**, 2823–2829 (1999).
[61]T. C. Allen, P. L. Walters, and N. Makri, "Direct computation of influence functional coefficients from numerical correlation functions," J. Chem. Theory Comput. **12**, 4169–4177 (2016).
[62]P. L. Walters, T. C. Allen, and N. Makri, "Direct determination of discrete harmonic bath parameters from molecular dynamics simulations," J. Comput. Chem. **38**, 110–115 (2017).
[63]A. Bose, "Zero-cost corrections to influence functional coefficients from bath response functions," J. Chem. Phys. **157**, 054107 (2022).
[64]J. C. Tully, "Molecular dynamics with electronic transitions," J. Chem. Phys. **93**, 1061–1071 (1990).
[65]J. C. Tully, "Perspective: Nonadiabatic dynamics theory," J. Chem. Phys. **137**, 22A301 (2012).
[66]L. Wang, A. Akimov, and O. V. Prezhdo, "Recent progress in surface hopping: 2011–2015," J. Phys. Chem. Lett. **7**, 2100–2112 (2016).
[67]R. Lambert and N. Makri, "Quantum-classical path integral. I. Classical memory and weak quantum nonlocality," J. Chem. Phys. **137**, 22A552 (2012).
[68]R. Lambert and N. Makri, "Quantum-classical path integral. II. Numerical methodology," J. Chem. Phys. **137**, 22A553 (2012).
[69]T. Banerjee and N. Makri, "Quantum-classical path integral with self-consistent solvent-driven reference propagators," J. Phys. Chem. B **117**, 13357–13366 (2013).
[70]F. Wang and N. Makri, "Quantum-classical path integral with a harmonic treatment of the back-reaction," J. Chem. Phys. **150**, 184102 (2019).
[71]J. Cerrillo and J. Cao, "Non-Markovian dynamical maps: Numerical processing of open quantum trajectories," Phys. Rev. Lett. **112**, 110401 (2014).
[72]N. Makri, "Small matrix path integral with extended memory," J. Chem. Theory Comput. **17**, 1–6 (2021).
[73]C. Rackauckas and Q. Nie, "Differentialequations.jl—A performant and feature-rich ecosystem for solving differential equations in Julia," J. Open Res. Softw. **5**, 15 (2017).
[74]Ch. Tsitouras, "Runge–Kutta pairs of order 5(4) satisfying only the first column simplifying assumption," Comput. Math. Appl. **62**, 770–775 (2011).
[75]M. Fishman, S. White, and E. Stoudenmire, "The ITensor software library for tensor network calculations," SciPost Phys. Codebases **4**, 4 (2022).
[76]M. Fishman, S. White, and E. Stoudenmire, "Codebase release 0.3 for ITensor," SciPost Phys. Codebases **2022**, 4-r0.3.
[77]T. Besard, C. Foket, and B. De Sutter, "Effective extensible programming: Unleashing Julia on GPUs," IEEE Trans. Parallel Distrib. Syst. **30**, 827 (2018).
[78]R. Silbey and R. A. Harris, "Variational calculation of the dynamics of a two level system interacting with a bath," J. Chem. Phys. **80**, 2615–2617 (1984).
[79]D. Xu and J. Cao, "Non-canonical distribution and non-equilibrium transport beyond weak system-bath coupling regime: A polaron transformation approach," Front. Phys. **11**, 110308 (2016).
[80]D. Xu, C. Wang, Y. Zhao, and J. Cao, "Polaron effects on the performance of light-harvesting systems: A quantum heat engine perspective," New J. Phys. **18**, 023003 (2016).
[81]S. J. Jang, "Partially polaron-transformed quantum master equation for exciton and charge transport dynamics," J. Chem. Phys. **157**, 104107 (2022).
[82]C. K. Lee, J. Moix, and J. Cao, "Accuracy of second order perturbation theory in the polaron and variational polaron frames," J. Chem. Phys. **136**, 204120 (2012).

[83]Y. Tanimura and P. G. Wolynes, "Quantum and classical Fokker–Planck equations for a Gaussian–Markovian noise bath," Phys. Rev. A **43**, 4131–4142 (1991).
[84]C. Duan, Q. Wang, Z. Tang, and J. Wu, "The study of an extended hierarchy equation of motion in the spin-boson model: The cutoff function of the sub-Ohmic spectral density," J. Chem. Phys. **147**, 164112 (2017).
[85]B. Popescu, H. Rahman, and U. Kleinekathöfer, "Using the Chebychev expansion in quantum transport calculations," J. Chem. Phys. **142**, 154103 (2015).
[86]H. Tian and G. Chen, "Application of hierarchical equations of motion (HEOM) to time dependent quantum transport at zero and finite temperatures," Eur. Phys. J. B **86**, 411 (2013).
[87]H. Liu, L. Zhu, S. Bai, and Q. Shi, "Reduced quantum dynamics with arbitrary bath spectral densities: Hierarchical equations of motion based on several different bath decomposition schemes," J. Chem. Phys. **140**, 134106 (2014).
[88]I. S. Dunn, R. Tempelaar, and D. R. Reichman, "Removing instabilities in the hierarchical equations of motion: Exact and approximate projection approaches," J. Chem. Phys. **150**, 184109 (2019).
[89]A. Ishizaki and G. R. Fleming, "Theoretical examination of quantum coherence in a photosynthetic system at physiological temperature," Proc. Natl. Acad. Sci. U. S. A. **106**, 17255–17260 (2009).
[90]J. Adolphs and T. Renger, "How proteins trigger excitation energy transfer in the FMO complex of green sulfur bacteria," Biophys. J. **91**, 2778–2797 (2006).
[91]P. L. Walters and N. Makri, "Iterative quantum-classical path integral with dynamically consistent state hopping," J. Chem. Phys. **144**, 044108 (2016).
[92]A. Ishizaki and G. R. Fleming, "Unified treatment of quantum coherent and incoherent hopping dynamics in electronic energy transfer: Reduced hierarchy equation approach," J. Chem. Phys. **130**, 234111 (2009).
[93]A. Hjorth Larsen, J. Jørgen Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. Bjerre Jensen, J. Kermode, J. R. Kitchin, E. Leonhard Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. Bergmann Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, and K. W. Jacobsen, "The atomic simulation environment—A Python library for working with atoms," J. Phys.: Condens. Matter **29**, 273002 (2017).
[94]U. Schollwöck, "The density-matrix renormalization group," Rev. Mod. Phys. **77**, 259–315 (2005).
[95]U. Schollwöck, "The density-matrix renormalization group in the age of matrix product states," Ann. Phys. **326**, 96–192 (2011).
[96]U. Schollwöck, "The density-matrix renormalization group: A short introduction," Philos. Trans. R. Soc., A **369**, 2643–2661 (2011).
[97]A. Bose and P. L. Walters, "Effect of temperature gradient on quantum transport," Phys. Chem. Chem. Phys. **24**, 22431 (2022).
[98]Dataset: A. Bose and A. Marshall, (2023). "Quantum dynamics," Github. https://github.com/amartyabose/QuantumDynamics.jl