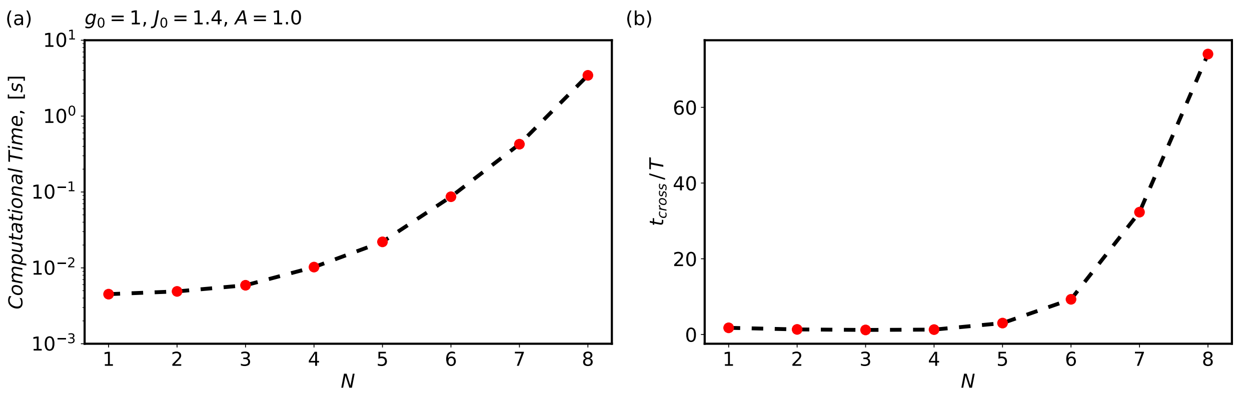

Figure 8: Numerical study of the performance of Floquet basis and `sesolve` depending on the dimension of the system N. (a) Computational time needed to reach the $t_{\rm cross}$ depending on N. (b) Normalised time $t_{\rm cross}/T$ at which Floquet basis and `sesolve` require similar computational time versus N. Discrete data points are shown as red dots, while dashed lines are included to facilitate visualization of the overall trend.

direct `sesolve()` method. Figure 8(a) shows the computational time required to evolve the system until that crossover time.

In addition to the Floquet basis transformation and the `fsesolve()` solver mentioned before, QuTiP includes the solver `fmmesolve()` which is capable of analyzing time-periodic Hamiltonians which are affected by a dissipative bath, i.e., quasi-time-periodic Hamiltonians. We refer the reader to the QuTiP documentation for examples using this solver. When applying it, the user must proceed with caution since a generalized Floquet method can give unphysical results if the dissipation rate is too large compared to the pure eigenenergies of the periodic Hamiltonian. In a future work, we plan to discuss the intricacies of generalized Floquet methods and this solver in particular. Moreover, another solver `flimesolve()`, which also uses Floquet theory to approach open quantum systems, is currently being developed [56].

3.2.11. `smesolve`*: Stochastic master equation solver*

When modelling an open quantum system, classical stochastic noise can be used to simulate a large range of phenomena. For example, classical noise can be used as a random term in the Hamiltonian as a means to simulate a classical environment randomly changing some system property in each run of an experiment. Another example is the Monte Carlo solver discussed earlier, where classical randomness is used to simulate the random chance of a quantum jump occurring in a dissipative Lindblad process.

In the `smesolve()` solver that we will discuss in this section, noise appears because of a continuous measurement. The solver allows us to generate the trajectory evolution of a quantum system conditioned on a noisy measurement record. Historically, this type of solver was used by the quantum optics community to model homodyne (single quadrature) and heterodyne (two-quadrature) detection of light emitted from a cavity. However, the solver is quite general, and can in principle be also applied to other types of problems.

To demonstrate its use we will focus on the standard example of a stochastic master equation describing an optical cavity whose output is subject to homodyne detection. The cavity obeys the general stochastic master equation,

$$d\rho(t) = -i[H, \rho(t)]\, dt + \mathcal{D}[a]\rho(t)\, dt + \mathcal{H}[a]\rho\, dW(t) \tag{34}$$

with $\mathcal{D}[a]\rho = a\rho a^\dagger - \frac{1}{2}a^\dagger a\rho - \frac{1}{2}\rho a^\dagger a$ being the Lindblad dissipator and $H = \Delta a^\dagger a$ the Hamiltonian, which together capture the deterministic part of the system's evolution. The term $\mathcal{H}[a]\rho = a\rho + \rho a^\dagger - \mathrm{tr}\left[a\rho + \rho a^\dagger\right]$ represents the stochastic part which captures the conditioning of a trajectory through continuous monitoring of the operator a. Here, $dW(t)$ is the increment of a Wiener process obeying $\mathbb{E}[dW] = 0$ and $\mathbb{E}[dW^2] = dt$.

This equation can be easily implemented in `smesolve()` with

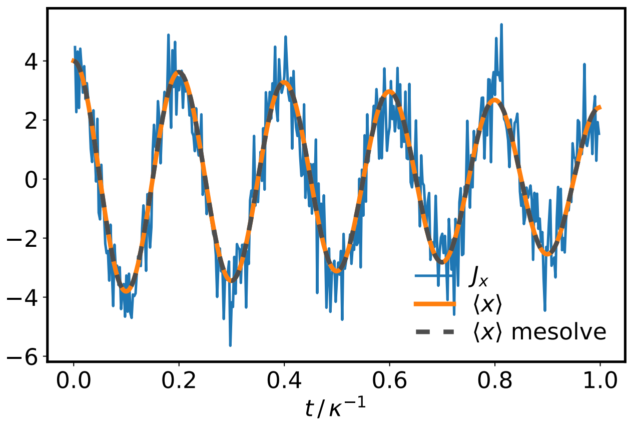

Figure 9: An example of solving the stochastic master equation for a dissipative cavity, with decay rate κ and Hamiltonian $H = \Delta a^\dagger a$, undergoing homodyne monitoring of the output field. Here we plot the averaged homodyne current $J_x = \langle x\rangle + dW/dt$, the average system behavior $\langle x\rangle$ for 50 trajectories, and the average result for $\langle x\rangle$ predicted by `mesolve` for the same model (without resolving conditioned trajectories). In this example the conditioned trajectories are only weakly affected by the noise, while the homodyne current remains noisy. Parameters used were $\Delta = 10\pi\kappa$, and initial condition of a coherent state with displacement $\alpha = 2$.

```
stoc_solution = qt.smesolve(
    H, rho_0, times, c_ops=[], sc_ops=[np.sqrt(kappa) * a], e_ops=[x],
    ntraj=num_traj, options={"dt": 0.00125, "store_measurement": True}
)
```

We have chosen here an initial coherent state, and collect expectation values of the operator $x = a + a^\dagger$. The parameter `sc_ops` indicates which operators are being monitored, and `stoc_solution.expect` returns a list of the averaged results, while `stoc_solution.measurement` returns a list of lists of individual measurement results of the operator $J_x(t) = \langle x\rangle + dW/dt$. The optional `c_ops` can be a standard list of collapse operators describing additional, unmonitored, baths. The results of this simulation are shown in Fig. 9. Interestingly, in this example the conditioned system trajectories are only weakly affected by the noise, making it a useful example for tests of this solver.

3.2.12. `HEOMSolver`*: Hierarchical Equations of Motion*

While the other solvers in QuTiP largely rely on either perturbative approximations and/or assumptions about the Markovianity of the environment a system is coupled to, the hierarchical equations of motion (HEOM) are a numerically exact method [22, 57, 58] to solve the dynamics of an open quantum system, under a minimal set of assumptions. It originated in the field of physical chemistry, where it was used to solve problems related to electronic energy transport in photosynthetic light-harvesting. It has recently found utility in a broad range of other fields, from quantum information to quantum electronics, and it is now used as a benchmark for developing other methods [59]. In QuTiP we provide a solver for both bosonic and fermionic environments, which in an upcoming release will support arbitrary spectral densities and correlation functions via a built-in fitting procedure.

The minimal assumptions that the HEOM method relies on are that the bath is Gaussian, initially in an equilibrium thermal state, and that the bath operator which couples to the system is linear. Using the Feynman-Vernon influence functional, one can show that, under these assumptions, the influence of the environment is fully characterized by its second order correlation function. For a bosonic environment, this correlation function can be expressed as (see [60] for an explanation of the fermionic bath case, and [22, 61] for applications)

$$C(t) = \int_0^\infty d\omega\, \frac{J(\omega)}{\pi} \left(\coth\left(\frac{\beta\omega}{2}\right) \cos(\omega t) - i \sin(\omega t) \right) . \tag{35}$$

The derivation of the HEOM is also based on the Feynman-Vernon influence functional. The HEOM relies on the assumption that the correlation function can be written as a sum of decaying exponentials like

$$C(t) = C_R(t) + C_I(t) \qquad \text{with} \tag{36a}$$

$$C_R(t) = \sum_{k=1}^{N_R} c_k^R \exp\bigl(-\gamma_k^R t\bigr) \qquad \text{and} \tag{36b}$$

$$C_I(t) = \sum_{k=1}^{N_I} c_k^I \exp\bigl(-\gamma_k^I t\bigr) \tag{36c}$$

(though some variants of the technique have generalized this assumption [62]). By taking repeated derivatives of the influence functional, in conjunction with the assumption (36), one arrives at the coupled HEOM differential equations

$$\begin{aligned}\dot{\rho}^n(t) = -i[H_S, \rho^n(t)] - \sum_{j=R,I} \sum_{k=1}^{N_j} n_{jk} \gamma_k^j \rho^n(t) - i \sum_{k=1}^{N_R} c_k^R n_{Rk} \{Q, \rho^{n_{Rk}^-}(t)\} \\ + \sum_{k=1}^{N_I} c_k^I n_{Ik} \{Q, \rho^{n_{Ik}^-}(t)\} - i \sum_{j=R,I} \sum_{k=1}^{N_j} [Q, \rho^{n_{jk}^+}(t)]\,,\end{aligned} \tag{37}$$

where $n = (n_{R1}, n_{R2}, \ldots, n_{RN_R}, n_{I1}, n_{I2}, \ldots, n_{IN_I})$ is a multi-index label of non-negative integers n_{jk}, and n_{jk}^- (n_{jk}^+) denotes the multi-index with the selected entry reduced (increased) by one. Further, Q is the generic system operator which couples to the bath. In practice, the *a priori* infinite hierarchy is truncated to $n_{jk} \leq N_c$, for a suitably chosen cutoff N_c. While the label $n = (0, 0, \ldots, 0)$ corresponds to the system density matrix, operators $\rho^n(t)$ with $n \neq (0, 0, \ldots, 0)$ are referred to as auxiliary density operators (ADOs), and encode correlations between system and bath.

To choose the value of the cutoff N_c, one typically starts with a small value which is then increased step by step until convergence is found. Heuristic arguments indicate that a lower bound is given by [63]

$$N_c \gtrsim \frac{\omega_S}{\min_{k,j} \mathrm{Re}[\gamma_k^j]}, \tag{38}$$

where ω_S is the largest system frequency and Re denotes the real part. In future versions of QuTiP, we are planning to implement more efficient cutoff mechanisms, where ADOs are kept or discarded according to an importance criterion [64–66].

The implementation of the HEOM in QuTiP is explained in greater detail in [22]. Following the release of QuTiP v5, it is currently being enhanced to be more compatible with the other solvers (`mesolve()` and `brmesolve()`) with a generic environment class that allows us to quickly compute the power spectrum. The logic of this new environment class, and its functionality, are described in Fig. (11) and Tables (7), (8) and (9).

Basic example. — To demonstrate how to use the HEOM solver in practice, let us consider the evolution of a qubit in a thermal bosonic environment with the Hamiltonian

$$H = \frac{\omega_0}{2}\sigma_z + \frac{\Delta}{2}\sigma_x + \sum_k w_k a_k^\dagger a_k + \sum_k g_k \sigma_z (a_k + a_k^\dagger)\,. \tag{39}$$

In the continuum limit, one can describe the couplings through the spectral density

$$J(\omega) = \pi \sum_k |g_k|^2 \delta(\omega - \omega_k)\,. \tag{40}$$

Typically, bosonic HEOM solvers use either the overdamped Drude-Lorentz spectral density or the underdamped Brownian motion spectral density. Let us for example consider the underdamped spectral density

$$J(\omega) = \frac{\lambda^2 \Gamma \omega}{(\omega_c^2 - \omega^2)^2 + \Gamma^2 \omega^2} \,. \tag{41}$$

We begin by initializing a bath, based upon the parameters in the spectral density:

```
env = UnderDampedEnvironment(lam=lam, gamma=gamma, T=T, w0=w0) #Create environment
bath = env.approximate("matsubara",Nk=5) #Approximate environment with Matsubara series
```

The `env` object contains the exact information about the bosonic bath, while the `bath` object is an approximated version of `env` that is specifically designed for the HEOMSolver, which requires as input an exponential decomposition of the bath correlation functions as described above. For this particular spectral density, Eq. 36 can be realized via the Matsubara or Padé decompositions. These decompositions rely on a truncation parameter N_k that introduces an approximation to the true bath dynamics by truncating the Matsubara or Padé decompositions into a finite number of exponentials. The QuTiP implementation makes it easy to see the impact of the approximation, as it is straightforward to compute the exact and approximated quantities that describe the bath (correlation function, spectral density and power spectrum)

```
env = UnderDampedEnvironment(lam=lam, gamma=gamma, T=T, w0=w0) #Exact environment
bath = env.approximate("matsubara",Nk=5) #Approximate environment
C = env.correlation_function(t) #Exact environment correlation functions
C2 = bath.correlation_function(t) #Approximate environment correlation functions
```

Similar notation is used for both the power spectrum and spectral density. The power spectrum can be used as quick gateway to compare with the other available solvers. For example, we can solve the HEOM equations by using the bath object (in a tuple, alongside which system operator `Q` it couples to), the system Hamiltonian and the `max_depth` parameter that specifies the cutoff N_c of the hierarchy equations, with the `HEOMSolver` as follows:

```
# -- HEOM --
solver = HEOMSolver(Hsys, (bath, Q), max_depth=9)
result_h = solver.run(rho0, t)
```

Simultaneously, we can compare this to a Bloch-Redfield solution also using the bath object properties,

```
# -- BLOCH-REDFIELD --
a_ops = [[Q, env]]
resultBR = brmesolve(Hsys, rho0, t, a_ops=a_ops, sec_cutoff=-1)
```

See Fig. 10 for a comparison of the results.

Ohmic bath with exponential cutoff. — The QuTiP implementation of the HEOM allows for the simulation of more general spectral densities: the user can create an arbitrary `BosonicEnvironment` from the spectral density, correlation function or power spectrum. As the HEOM requires a decaying exponential representation of the correlation functions, we can either fit the spectral density with one with a known analytical decomposition of its correlation function, or the correlation functions themselves, as explained in [22, 67]. Let us consider the simulation of a Ohmic bath, whose spectral density is given by

$$J(\omega) = \alpha \omega \exp\left(-|\omega|/\omega_c\right) \,. \tag{42}$$

Because this spectral density is frequently used in the literature, a special class, `OhmicEnvironment`, has been added for convenience. As mentioned, in order to use the HEOM solver, we must first choose between fitting either the correlation function or the spectral density. The code snippet below shows how easy it is to set up simulations using these different approaches:

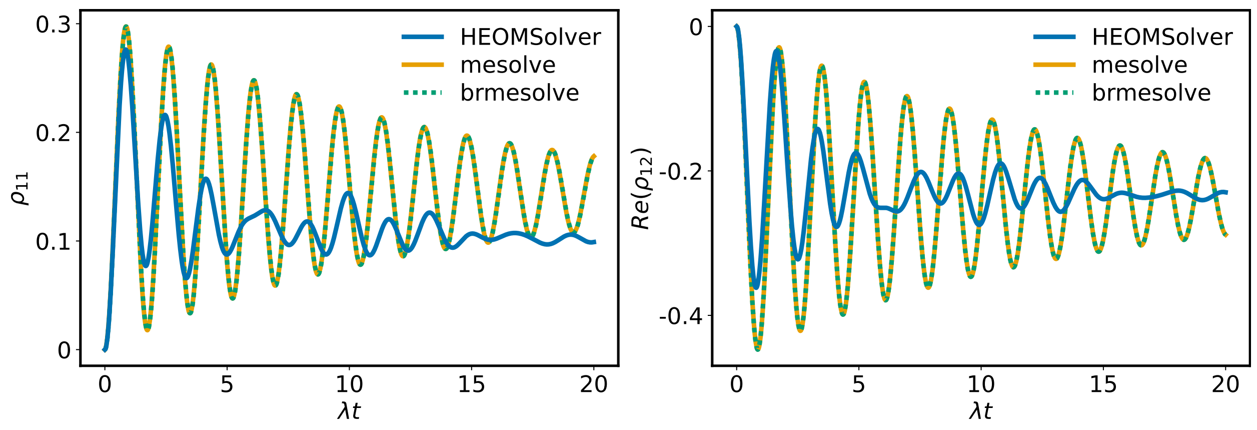

Figure 10: For the example of a standard spin-boson problem, we compare the output of the different master equation solvers available in QuTiP, namely, the Lindblad master equation solver (`mesolve()`), the Bloch-Redfield equation solver (`brmesolve()`) and the hierarchical equations of motion solver (`HEOMSolver`) for a spin coupled to a bosonic bath described by an underdamped Brownian motion spectral density. Left, (a), shows the evolution of the qubit population for the different approaches. Notice that both `brmesolve` and `mesolve` coincide, but they differ substantially from the HEOM result in this deeply non-Markovian regime. The right figure, (b), shows the dynamics of the coherence values. The parameters used to generate these figures are: $\lambda = 0.5\,\Delta$, $\Gamma = 0.1\,\Delta$, $T = 0.5\,\Delta$, $N_k = 5$, $N_c = 6$ and $\omega_0 = \frac{3\Delta}{2}$.

```
# -- FITTING PREDEFINED OHMIC CLASS--
w = np.linspace(0, 100, 2000) #Fitting range in frequency-domain
env_fs, _ = oh.approximate("sd",wlist=w, Nk=3, Nmax=3) #Fit spectral density


t = np.linspace(0, 10, 1000) #Fitting range in time-domain
env_fc, _ = oh.approximate("cf",tlist=t, Ni_max=5, Nr_max=4,
                           target_rmse=None, maxfev=int(1e9)) #Fit correlation functions
```

The first output of the approximation by fitting is a `BosonicEnvironment` object, while the second output provides the fit information. This is particularly useful when dealing with non-standard baths, where one often needs more exponents to accurately describe the bath even when dealing with high temperatures, as the fit information helps one decide how many exponents to take into consideration.

The bath obtained from the fitting can be passed to the solver to quickly obtain its dynamics

```
# -- SOLVING DYNAMICS --
tlist = np.linspace(0, 10, 1000)
HEOM_corr_fit = HEOMSolver(Hsys, (env_fc, Q), max_depth=5)
```

Figure 12 shows an example of fitting the Ohmic spectral density with exponential cut-off with a set of underdamped Brownian motion spectral densities (sometimes called the Meier-Tannor fitting approach).

Zero temperature and the localization-delocalization phase transition. — One of the benefits of including these fitting routines is to be able to simulate situations where the structure of the spectral density has non-trivial effects. An example of such a situation is the localization-delocalization transition in the spin-boson model [68–71]. When the temperature of the bath is $T = 0$, one expects the steady state of the system to be delocalized $\langle \sigma_z(t \to \infty) \rangle = 0$. However, when the coupling to the bath is increased the steady state goes from a completely delocalized state to what appears to be a localized state for long-times $\langle \sigma_z(t \to \infty) \rangle \neq 0$. To demonstrate this, we follow [70] and choose an Ohmic spectral density with a polynomial cutoff of the form

$$J(\omega) = \frac{\pi \alpha \omega}{2(1 + (\frac{w}{wc})^2)}\,. \tag{43}$$

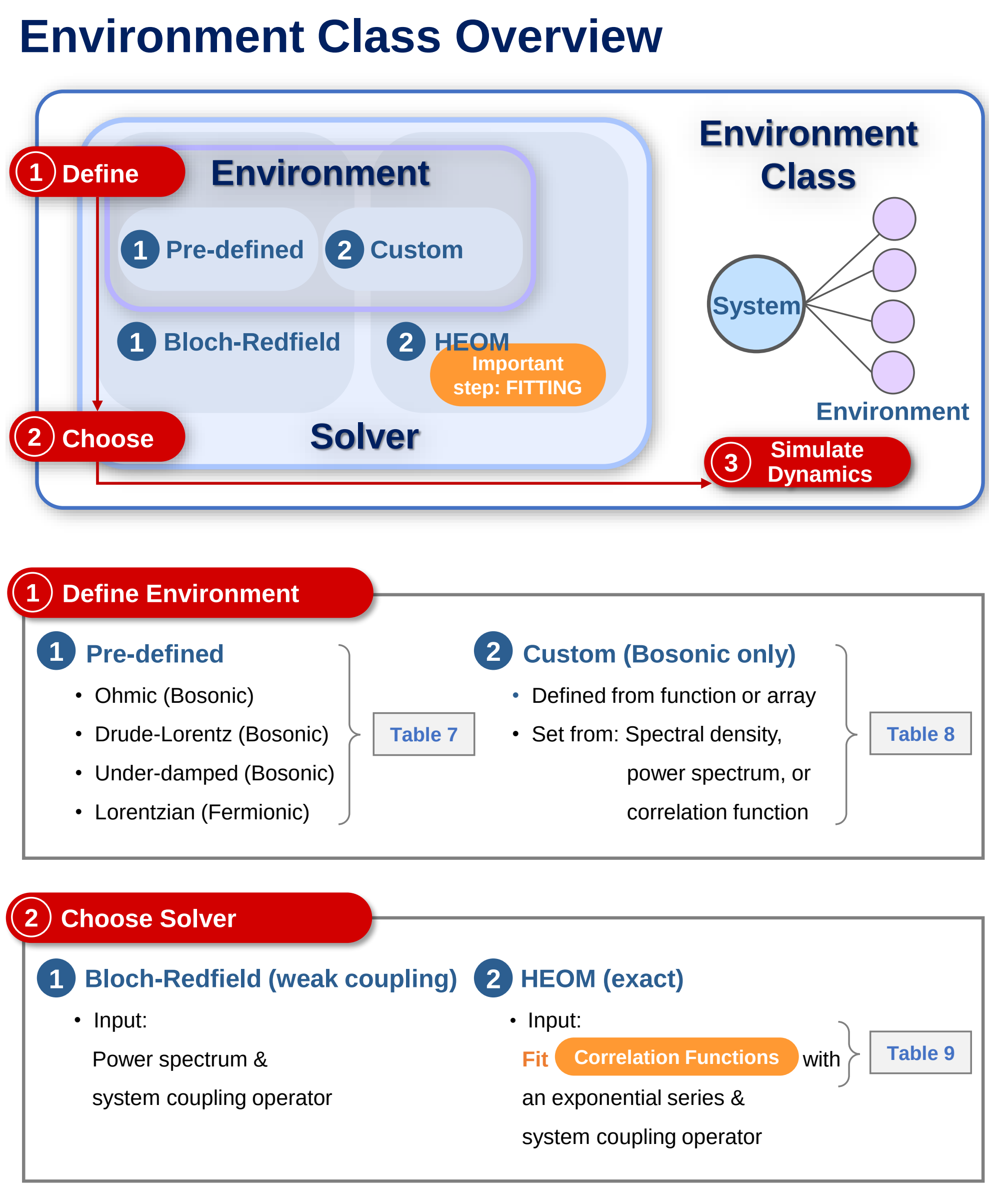

Figure 11: An overview of the environment class, which is now supported by the HEOM and Bloch-Redfield solvers. Details on how predefined and custom environments can be made is provided in tables (7) and (8), and the fitting methods that can be used to approximate the bath correlation function for use in the HEOM solver are described in table (9)

Predefined environment type	Spectral density Function	Python Function	Padé or Matsubara Available	Bosonic or Fermionic
Ohmic	$J(\omega) = \alpha(\omega^s/\omega_c^{s-1})\exp[-\omega/\omega_c]$	`OhmicEnvironment(T, alpha, wc, s)`	✗	Bosonic
Drude-Lorentz	$J(\omega) = 2\lambda\gamma\omega/(\gamma^2+\omega^2)$	`DrudeLorentzEnvironment(T, lam, gamma)`	✓	Bosonic
Under-Damped	$J(\omega) = \lambda^2\Gamma\omega/[(\omega_0^2-\omega^2)^2+\Gamma^2\omega^2]$	`UnderDampedEnvironment(T, lam, Gamma, w0)`	✓	Bosonic
Lorentzian	$J(\omega) = \gamma W^2/[(\omega-\omega_0)^2+W^2]$	`LorentzianEnvironment(T, mu, gamma, W, w0)`	✓	Fermionic

Table 7: Predefined environments: Table of predefined environments as given by their spectral density, and the internal QuTiP function to create them. The fourth column indicates whether or not a Padé or Matsubara series expansion of the bath correlation functions is available. All baths require an initial temperature (which can be zero, though this invalidates the series expansion for the correlation functions, and fitting should be used instead). For the Fermionic Lorentzian bath we also need to provide a chemical potential (defined via the `mu` parameter).

Due to this effect being present at $T = 0$, we fit the correlation function rather than the spectral density. Obtaining the correlation function is straightforward once we create a `BosonicEnvironment`.

```python
def J(w, alpha=1):
    """ The Ohmic bath spectral density as a function
    of w (and the bath parameters). """
    return (np.pi/2) * w * alpha * 1 / (1+(w/wc)**2)**2

env = BosonicEnvironment.from_spectral_density(lambda w: J(w, alpha), wMax=50, T=T)
```

For this example, let us consider the same system Hamiltonian as before, with $\omega_0 = 0$. Figure 13 shows how by increasing the coupling to the bath, the system goes from a localized state to a delocalized state. This is the so called spin-boson localization-delocalization phase transition [68].

Apart from the method used above, the QuTiP environment class provides several different ways to decompose arbitrary environments into a decaying exponential form as in Eq. (36), as needed for simulation with the HEOMSolver. Table 9 provides a summary of the methods available. For details about the methods, we refer to the documentation or Refs. [72, 73]. At present, most of these methods only work for bosonic environments; however, fermionic environments will be supported in the near future.

3.2.13. Visualization of solver results

QuTiP comes with a range of functions to visualize the results returned by its solvers. In addition to automated functions for plotting expectation values with Matplotlib, QuTiP provides utility functions to calculate important and commonly used representations of quantum states, including the Bloch-sphere for two-level systems and pseudo-probability functions, such as the Wigner and Husimi functions, for harmonic systems like cavities.

In this section, we present a few examples to illustrate the utility and features of these plotting functions. Many more examples can be found in the tutorial notebooks described in Appendix B.

For both learners and researchers, using the Bloch sphere to visualize the overlap of a qubit state with the Pauli matrices in three-dimensional space can be useful to help understanding how the state evolves, particularly when combined with QuTiP's animation features. In QuTiP, a Bloch sphere is instantiated with `b = Bloch()`. Points can be added to it using coordinates, as in `b.add_points([1/np.sqrt(3), 1/np.sqrt(3), 1/np.sqrt(3)])`, and the sphere plotted with `b.render()`. Vectors can be added in a similar way, using `b.add_vectors([0, 1, 0])`.

More commonly, one wants to visualize the output of solvers, which is provided in the form of state vectors or density operators. These objects can be added to a sphere using the method `b.add_states(state)`. The tutorial notebook `0004_qubit-dynamics`, briefly summarized in Appendix B, demonstrates this feature using the example of a driven qubit, both with and without noise, and the result is shown in Fig. 14.

Defined from:	**Python Function**	**Important optional arguments**
Spectral density	`BosonicEnvironment .from_spectral_density(func)`	`T`: temperature of the bath. `wlist`: array of frequencies (if array-based). `wMax`: maximum frequency used in Fourier-transform conversions.
Power Spectrum	`BosonicEnvironment .from_power_spectrum(func)`	As above
Correlation function	`BosonicEnvironment .from_correlation_function(func)`	`T`: temperature of the bath. `tlist`: array of times (if array-based). `tMax`: maximum time used in Fourier-transform conversions.
Correlation function from exponential series (bosonic or fermionic)	`ExponentialBosonicEnvironment (ck_real, ck_imag, vk_real,vk_imag)` `ExponentialFermionicEnvironment (ck_plus, vk_plus, ck_minus, vk_minus)`	`T`: temperature of the bath. `combine`: Boolean, indicates whether common-frequency compression is used (bosonic case only). `mu`: Chemical potential (fermionic case only).

Table 8: Custom environments: Table of functions for defining custom environments from either their spectral density, power spectrum or correlation functions (only custom bosonic environments are currently supported in this way, though fermionic ones can be done manually as shown in the final row in the table). The parameter `func` can be a Python function or a list/array of points (requires a corresponding `tlist` or `wlist` optional parameter to be provided). Both `wMax` and `tMax` should be chosen such that the function or array used is negligible for higher values of t or ω.

Pseudo-probability functions are also commonly used to visualize data arising from continous variable systems. In QuTiP, such systems are truncated on finite dimensional Fock spaces, but these pseudo-probability functions can still be used, given sufficient truncation. The Wigner function, for example, helps visualize the probability of the position and momentum quadratures of cavities, and famously contains negative probabilities for non-classical states.

We can demonstrate the visualization of Wigner functions straightforwardly with another common example; a cavity prepared in a "Schrödinger cat" state, i.e., a superposition $\psi = \frac{1}{N}\left(|\alpha_1\rangle + |\alpha_2\rangle\right)$ of two coherent states (where N is a normalization factor). The Wigner function, shown in Fig. 15, clearly shows the expected negative values for such a highly non-classical state. The non-classicality is less apparent in the Husimi-Q function, also shown in Fig. 15, which is non-negative by definition.

New in QuTiP v5 is a suite of tools to automate the animation of many of these commonly used functions. These can be explored in the tutorial [74], and they include customized methods for the:

- Wigner function (`anim_wigner` and `anim_wigner_sphere`),
- Hinton plots (`anim_hinton`),
- sphere plots (`anim_sphereplot`),
- histograms (`anim_matrix_histogram`),
- Fock state distributions (`anim_fock_distribution`),
- spin distributions (`anim_spin_distribution`),
- Qubism plots (for ploting the states of many qudits, `anim_qubism`), and
- Schmidt plots (for plotting matrix elements of a quantum state, `anim_schmidt`).

In addition, there is a new option (`qutip.settings.colorblind_safe`) to choose plotting colors from a pallette of colorblind safe colors.

Method	Arbitrary Functions	Allows Constraints	No need of Extra Input	Optimization Convergence	Sampling Insensitive	Arbitrary Temperatures	Works on Noisy data	Recommended when...
NLSQ (ps,sd,cf)	✓	✓	✗	✗	✗	✓	✓	You have an idea about which exponents should be included [67].
AAA	✓	✗	✓	✓	✗	✓	✓	You need high-accuracy in the steady-state and the spectral density is not too structured.
Prony	✓	✗	✓	✓	✗	✓	✗	The correlation function is noiseless and long lived.
Matsubara	✗	✗	✓	✓	✓	✗	✗	Doing high temperature simulations using the specific spectral densities it is available for.
Pade	✗	✗	✓	✓	✓	✗	✗	The same cases as the Matsubara method. This is recommended over Matsubara whenever available.
ESPIRA	✓	✗	✓	✗	✗	✓	✓	Looking for a general-purpose method.
ESPRIT	✓	✗	✓	✓	✗	✓	✓	The correlation function is long lived.

Table 9: Once an environment is created (see Tables (7) and (8)), its correlations can then be approximated as an exponential series in a variety of ways. This is done by the class method `.approximate("type")` where `"type"` is a string defining the method desired, as listed in the first column of this table. They take a variety of optional arguments to define fitting range or bounds, depending on the method used. Non-linear least squares fitting (NLSQ), the method used in this section, might be performed on the spectral density, the power spectrum, or the correlation function, as specified by the type strings "ps", "sd", and "cf". "Arbitrary Functions" stands for the ability of approximating an arbitrary user-defined environment. "Allows Constraints" stands for the ability to limit the range of values the parameters in Eq. 36, which is often useful when working with HEOM as it may improve convergence. "No need of Extra Input" refers to the input that the algorithm needs to work properly: we consider a set of sampling points as basic input, and other pieces of information required as extra input. "Optimization Convergence" refers to the fact that the approximation does not get stuck in local minima preventing convergence, in some cases failing to provide any approximation. "Sampling Insensitive" refers to the fact that changing the sampling points does not change the approximation. "Arbitrary Temperatures" refers to the method converging at arbitrarily low temperatures. "Works on Noisy data" applies when the user defined spectral density comes from data that is noisy.

3.3. Additional features in QuTiP v5

In addition to the new data layer and solver features described earlier, there are other new features in QuTiP v5, and some of the previously existing features have received important updates. Below, we describe in detail three such new or updated features: excitation number restricted states, which are crucial for the simulation of large composite systems, and now have improved back-end support through a new dimensions class, the option of using the JAX auto-differentiation functionality with the new JAX data layer, and support for the Message Passing Interface (MPI) in the parallelization of various solvers, which enables the easy use of super-computing resources.

3.3.1. Excitation number restricted states

When modeling many interacting quantum systems, QuTiP does not by default apply any approximation apart from truncating the individual systems' Hilbert spaces. For example, when modelling the discrete Fock-space representation of a quantum harmonic oscillator, we normally truncate the states at some finite number of excitations. This is clearly demonstrated when defining the commonly used Jaynes-Cummings

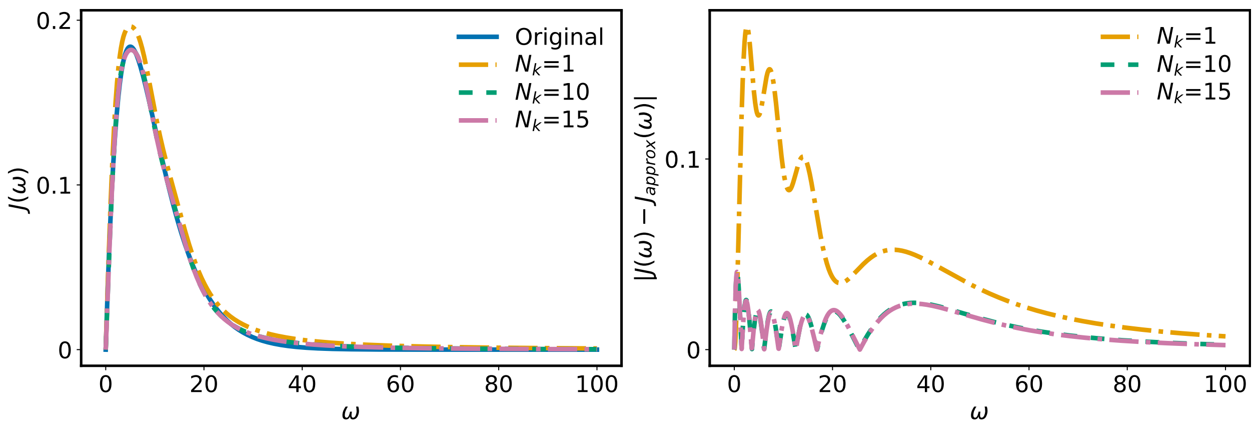

Figure 12: Approximation of the spectral density of an Ohmic Bath via fitting with three underdamped Brownian motion spectral density functions. Importantly, when fitting spectral densities for the HEOM method, we must still expand each spectral density's contribution to the total correlation functions in terms of exponentials. To gain insight of the effectiveness of both the fit and the correlation function expansion, we can then use that expansion, along with the temperature information of the bath, to reconstruct a total effective spectral density, which is what we show in this plot. In the figure, we can see that the number N_k of exponents kept per underdamped mode has an effect on the approximation of the resulting effective bath spectral density. Ultimately one should aim to find the minimum number of exponents that make up a good approximation of the spectral density. The parameters of the original Ohmic bath are $\lambda = 0.1$, $\omega_c = 5$, and $T = 1$.

model from quantum optics, which describes a single two-level atom interacting with a single-mode cavity under the rotating wave approximation:

```
N_cut = 2 #Number of Fock states to keep

psi0 = qt.basis(2, 0) & qt.basis(N_cut, 0) #initial state (qubit excitated)
sz = qt.sigmaz() & qt.qeye(N_cut)
sm = qt.sigmam() & qt.qeye(N_cut)
a = qt.qeye(2) & qt.destroy(N_cut)

H_JC = (
    0.5 * eps * sz + omega_c * a.dag() * a +
    g * (a * sm.dag() + a.dag() * sm)
) #Jaynes-Cummings Hamiltonian
```

In this case the Hilbert space of the cavity is truncated at two Fock states, so the total number of states is four. However, the Hamiltonian conserves the total number of excitations in the coupled system. With an initial condition containing at most a single excitation, the double-excitation state, where the atom is excited and a photon is in the cavity, is therefore decoupled from the dynamics. Using excitation number restricted (ENR) states, we can truncate the total Hilbert space to exclude the double-excitation state:

```
N_exc = 1 #Number of total excitations in truncated system
dims = [2, N_cut] #Original system dimensions (before ENR truncation)

psi0 = qt.enr_fock(dims, N_exc, [0, 0]) #Initial state in ENR space
sm, a = qt.enr_destroy(dims, N_exc) #operators in ENR space
sz = 2 * sm.dag() * sm - 1

H_enr = (
    0.5 * eps * sz + omega_c * a.dag() * a +
    g * (sm.dag() * a + a.dag() * sm)
) #Jaynes-Cummings Hamiltonian
```

Here, `N_exc=1` is the maximum number of excitations we wish to consider across the whole Hilbert space. The function `enr_destroy(dims, N_exc)` returns a list of annihilation operators for each subsystem in `dims` which only act on a reduced space including states with up to that total excitation number. In this example,

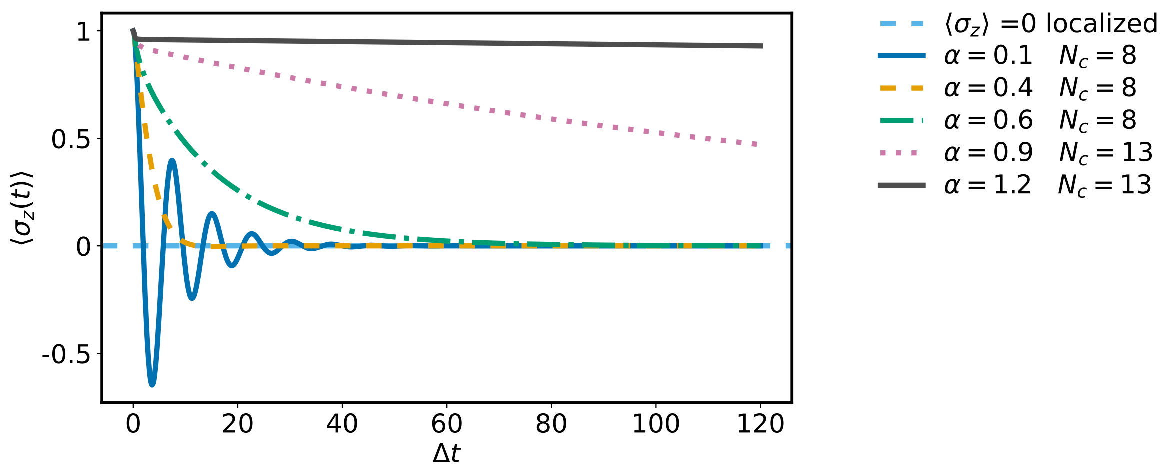

Figure 13: This figure shows the spin-boson localization-delocalization phase transition. For the simulation, we used $\omega_0 = 0$, $\omega_c = 10\,\Delta$ and $T = 0$. The correlation function was fit with two exponents for the real part and one for the imaginary part.

the restricted Hilbert space is spanned by $|0,0\rangle$, $|0,1\rangle$ and $|1,0\rangle$. The first annihilation operator can be thought of as the operator $|0,0\rangle\langle 1,0|$, and the second one as $|0,0\rangle\langle 0,1|$. With these constructions, we can then recreate the full Hamiltonian and dynamics of the Jaynes-Cummings model, omitting the unimportant double occupation state.

The power of this approach lies in situations with many subsystems, where one only needs to consider a limited number of excitations. One of the core QuTiP notebooks demonstrates this well, with a large chain of coupled Jaynes-Cummings models [75]. But it can also be used as a powerful truncation tool in situations where the Hamiltonian is not necessarily excitation-number conserving (see [76]).

It is important to note that since ENR states essentially compress the normal tensor structure of states and operators onto one single reduced Hilbert space, annihilation and creation operators of different subsystems no longer commute. Hence, care must be taken when representing operators on the ENR state space. Typically, when constructing Hamiltonians, one should order annihilation operators to the right and creation operators to the left. In addition, ENR states require the use of a range of custom functions, like `enr_fock()`, and many standard utility functions in QuTiP will fail when used with them. In v5 the addition of unique dimension objects for ENR states potentially allows these issues to be resolved, but general compatibility is still ongoing work.

To demonstrate the utility of ENR states in a complex problem, we now consider an important example from the literature; that of a qubit interacting with a one-dimensional waveguide [77] truncated by a mirror. Generally, this type of problem involving time-delayed feedback is difficult to model numerically [78]. One common method to capture the finite time delay of photons reaching the mirror and returning to the system is discretization of modes in the waveguide. This discretization can be performed in multiple ways (see [77] and [79]), but the approach taken in [80, 81] is particularly amenable to using ENR states. In this approach, spatial discretization and temporal discretization of the waveguide are done hand-in-hand, and open ends of the wave guide are truncated using a Monte Carlo-like measurement step. This procedure still requires modelling a large number of waveguide modes, or “boxes”, which the authors of [80] were able to achieve using an ENR-like truncation of their basis states (albeit done manually, not using QuTiP).

It is relatively straightforward to implement this procedure in QuTiP. However, following [80], we implement the time evolution manually using a product of propagators for small discrete time steps rather than using one of the standard QuTiP solvers directly. We refer readers to [80] for a complete description, but

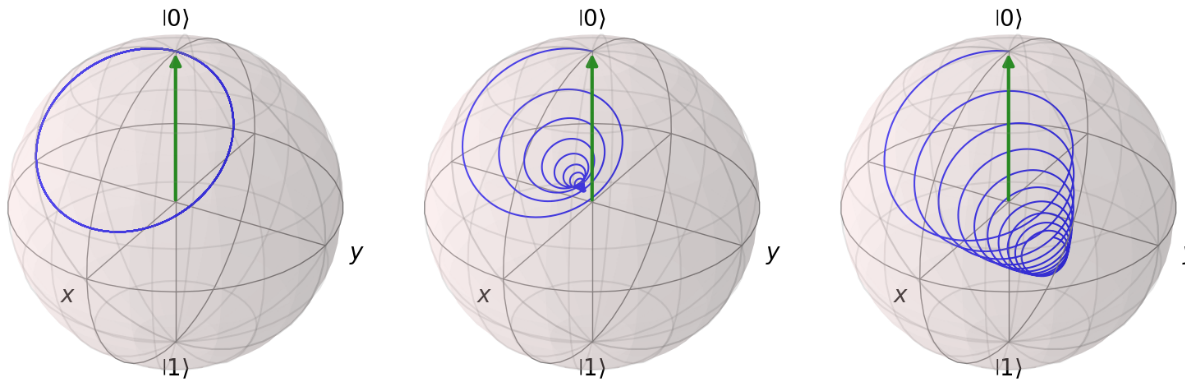

Figure 14: The left figure shows the Bloch sphere representation of the dynamics of a qubit undergoing unitary evolution under the Hamiltonian $H = \Delta[\cos(\theta)\sigma_z + \sin(\theta)\sigma_x]$ with $\theta = 0.1\,\pi$ for an initially excited qubit. The middle figure shows the dynamics of the same system with added dephasing, with a rate $\gamma_p = 0.5\,\Delta/(2\pi)$. The right figure shows the same model but with added relaxation at the rate $\gamma_r = 0.5\,\Delta/(2\pi)$.

succinctly, one starts with the Hamiltonian for the system coupled to a discrete-frequency waveguide:

$$H = \frac{\epsilon}{2}\sigma_z + \sum_{\alpha\in\{L,R\}}\sum_{k=0}^{N-1}\omega_k b^\dagger_{k,\alpha}b_{k,\alpha} + \sqrt{\frac{2\pi}{L_0}}\sum_{\alpha\in\{L,R\}}\sum_{k=0}^{N-1}\kappa_\alpha(\omega_k)\,[\sigma_+ b_{k,\alpha} + \text{H.c.}]\ . \tag{44}$$

Here, it was assumed that a mirror truncates the left side of the waveguide at some finite distance $L_0/2$, and the sums run over left- and right-moving modes and over their discretized frequencies ω_k. The coupling terms are $\kappa_L(\omega) = \sqrt{\gamma_L/(2\pi)}$ and $\kappa_R(\omega) = \sqrt{\gamma_R/(2\pi)}e^{i\phi}e^{i\omega\tau}$, where γ_α are coupling constants, $\tau = L_0/c$ is the total travel time of photons to the mirror and back (with c being the speed of light in the waveguide), and ϕ is an additional phase change incurred from the reflection at the mirror. In other words, a photon can propagate to the left, hit the mirror, and then return as a right propagating photon that then interacts with the system with an accumulated phase $(\omega\tau + \phi)$.

The key step is to transform the discrete frequencies into spatially discretized modes with the discrete Fourier transform

$$B_{n,\alpha} = (1/\sqrt{N})\sum_{k=0}^{N-1} b_{k,\alpha}\exp[(\pm)_\alpha i\omega_k n\Delta t]\,, \tag{45}$$

where $\Delta t = L_0/N$ is the time-domain sampling corresponding to the spatial discretization of the total travel length L_0. The number of modes is N and, assuming linear dispersion, their frequencies are $\omega_k = 2\pi k/L_0$ (setting now $c = 1$). Finally, the sign in the exponent is "+" for right-moving and "−" for left-moving modes.

Under this transformation the model becomes one where photons emitted in a time interval Δt are then moved, conveyor-belt-like, through these discrete modes ("boxes") until they hit the mirror at time $\tau/2 \equiv N\Delta t/2$, and then return in right-moving boxes until they again interact with the system at time τ. Photons emitted into the right side of the waveguide never return, and can be accommodated by projecting, at each time step, the system onto the appropriate state depending on whether a photon is observed in the right-most-box or not. As previously mentioned, this projection technique has formal similarity with the Monte-Carlo wavefunction method.

Because the interaction between the qubit and the waveguide is assumed to be weak, and in a rotating wave approximation, we can model the whole setup including the qubit and N modes efficiently using ENR states. In Fig. 16 we show, reproducing [80], the dynamics of the excited state population of the qubit with two different choices of phase, $\phi = 0$ and $\phi = \pi$, demonstrating the effect of interference with the returning photons. Here we used $N = 21$, and only a single excitation, which with ENR states can be presented with

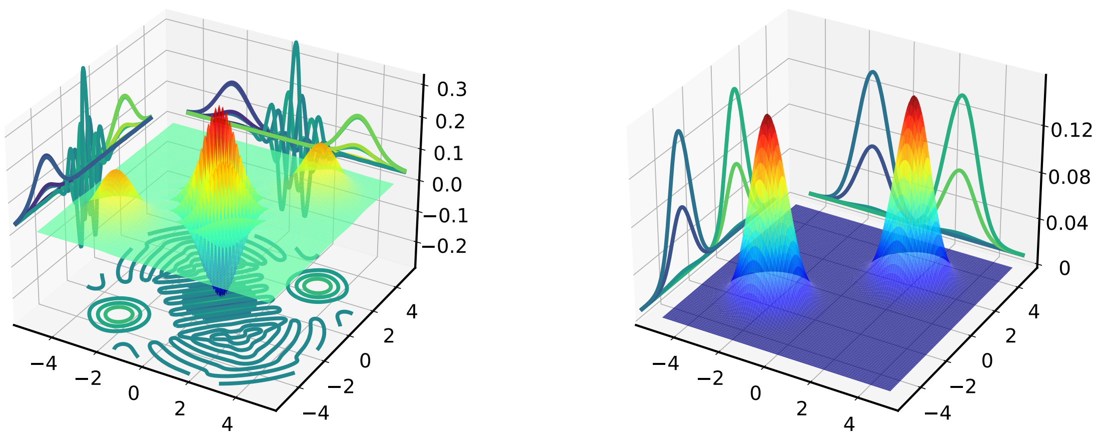

Figure 15: Left figure shows the Wigner function, a pseudo-probability distribution, of a cavity prepared in a Schrödinger cat state $\psi = \frac{1}{\sqrt{2}}(|\alpha_1\rangle + |\alpha_2\rangle)$, with $\alpha_1 = -2.0 - 2j$ and $\alpha 2 = 2.0 + 2j$. The right figure shows instead the Husimi-Q function for the same state.

a very small state space of just 23 states (to be compared to 2^{22} states needed for a brute force calculation with a Fock space truncation of each mode of just 2, without ENR states). Figure 17 shows the occupation of the waveguide modes as a function of time, illustrating the linear transportation of the excitation through the waveguide and the effect of the reflection phase ϕ on the waveguide populations.

3.3.2. Automatic Differentiation: JAX

In an earlier section, we showed how QuTiP's new JAX data layer can be used to run calculations on the GPU. An additional feature of using the JAX data layer is access to automatic differentiation, or auto-differentiation. In problems where derivatives are important, like optimization, we must often resort to numerical approximations, e.g., finite difference, to evaluate them. When higher order derivatives are required, such approximations can be numerically costly and inaccurate. Auto-differentiation relies on the concept that any numerical function is, at a low-level, expressible in elementary analytical functions and operations. This can be exploited, via the chain rule, to give access to the derivative of almost any higher-level function.

The JAX library makes it possible to conveniently and easily use auto-differentiation for a variety of applications. For example, in the `QuTiP-QOC` library we take advantage of this feature to find derivatives of a control objective with respect to the control parameters in order to find the optimal pulse shape implementing a complex operation.

A full explanation and set of examples of automatic differentiation is beyond the scope of this work, but we will showcase two basic examples here: one arising from the field of counting statistics, and the other relevant to Hamiltonian control. In addition, the auto-differentiation capabilities of QuTiP-JAX will also be used in the section on the optimal control package QuTiP-QOC.

Counting statistics of an open quantum system.. – In the first example, we have an open quantum system in contact with an environment, and there is a measurement device which keeps track of excitations flowing between system and environment.

The measurement results of this process can be expressed in a variety of ways, but considerable insight can be gained by thinking about the statistics of such events; the mean, variance, skewness, and so on, of the probability distribution describing the number of excitations n that have been exchanged by a certain time t. This distribution $P_n(t)$ is called the full counting statistics, and many common experimental observables such as current or shot noise can be extracted from its properties.