Q-Flow: Generative Modeling for Differential Equations of Open Quantum Dynamics with Normalizing Flows

Owen Dugan [1 2] **Peter Y. Lu** [3 2] **Rumen Dangovski** [4 2] **Di Luo** [2 5 6] **Marin Soljačić** [1 2]

Abstract

Studying the dynamics of open quantum systems can enable breakthroughs both in fundamental physics and applications to quantum engineering and quantum computation. Since the density matrix ρ, which is the fundamental description for the dynamics of such systems, is high-dimensional, customized deep generative neural networks have been instrumental in modeling ρ. However, the complex-valued nature and normalization constraints of ρ, as well as its complicated dynamics, prohibit a seamless connection between open quantum systems and the recent advances in deep generative modeling. Here we lift that limitation by utilizing a reformulation of open quantum system dynamics to a partial differential equation (PDE) for a corresponding probability distribution Q, the Husimi Q function. Thus, we model the Q function seamlessly with *off-the-shelf* deep generative models such as normalizing flows. Additionally, we develop novel methods for learning normalizing flow evolution governed by high-dimensional PDEs based on the Euler method and the application of the time-dependent variational principle. We name the resulting approach *Q-Flow* and demonstrate the scalability and efficiency of Q-Flow on open quantum system simulations, including the dissipative harmonic oscillator and the dissipative bosonic model. Q-Flow is superior to conventional PDE solvers and state-of-the-art physics-informed neural network solvers, especially in high-dimensional systems.

[1]Department of Physics, Massachusetts Institute of Technology [2]NSF AI Institute for Artificial Intelligence and Fundamental Interactions [3]Data Science Institute, University of Chicago [4]MIT EECS [5]Center for Theoretical Physics, Massachusetts Institute of Technology [6]Department of Physics, Harvard University. Correspondence to: Di Luo <diluo@mit.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

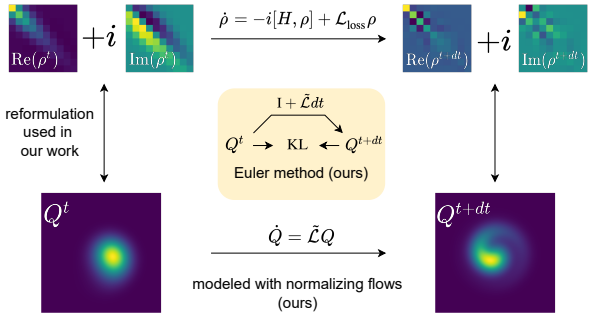

Figure 1. Q-Flow. We reformulate differential equations for density matrix dynamics as PDEs for probability distribution dynamics. We use off-the-shelf normalizing flows and our Euler-KL method for solving such PDEs.

1. Introduction

Understanding open quantum system dynamics is crucial for fundamental physics and high-impact scientific applications such as quantum engineering and quantum computation (Verstraete et al., 2009; Barreiro et al., 2011).

The state of an open quantum system is given by the density matrix ρ, which is an exponentially scaling object with a size that grows as N^{2k} for k subsystems each with a Hilbert space of dimension N. Thus, computing or evolving ρ becomes infeasible as k increases due to the curse of dimensionality. Pioneering work on representing ρ in a compact form as a customized deep generative neural network has shown great promise in advancing the frontier of understanding high-dimensional quantum systems (Vicentini et al., 2019; Yoshioka & Hamazaki, 2019; Hartmann & Carleo, 2019; Nagy & Savona, 2019). However, a number of computational challenges remain when solving for ρ, which motivates the development of novel machine learning methods. Notable challenges are:

1. The density matrix ρ is *complex-valued* and has the constraint $\mathrm{tr}[\rho] = 1$. That makes it non-trivial to model with standard generative models that are real-valued.

2. The differential equation that governs the dynamics of ρ models complicated interactions in high-dimensional

space, which hampers the application of conventional differential equation solvers.

3. Previous efforts to model ρ with neural networks are restricted to discrete spin systems, and it is unclear how to model ρ in continuous or bosonic systems.

The state-of-the-art literature has addressed Challenge 1 by developing customized neural architectures for ρ in spin systems with discrete degrees of freedom only (Vicentini et al., 2019; Yoshioka & Hamazaki, 2019; Hartmann & Carleo, 2019; Nagy & Savona, 2019; Luo et al., 2022b; Reh et al., 2021). Challenge 2 has been attempted by exploring physics-inspired training objectives, such as Physics-informed neural networks (PINNs) (Raissi, 2018; Raissi et al., 2019). PINNs have shown promise in low-dimensional systems, but it is not clear how they can be scaled to high-dimensional PDEs. Furthermore, the existing literature has not addressed Challenge 3 and missed an opportunity to establish a *direct* connection between modeling continuous variable open quantum dynamics and novel deep generative models for standard machine learning benchmarks. Such a connection would drive progress in both machine learning applications for open quantum dynamics and deep generative modeling.

In this paper, we address Challenges 1-3 by establishing a bridge between open quantum systems in a continuous Hilbert space and continuous-variable generative modeling.

Firstly, we reformulate the problem by replacing the density matrix ρ with an alternative representation, the Husimi Q function Q (Carmichael, 1999b), which can be practically considered as a probability distribution. Thus, we enable the use of *off-the-shelf* generative neural networks to model Q. Because the optimization of high dimensional quantum systems requires access to both easy sampling and probability density values, we use *normalizing flows* (Dinh et al., 2014; Rezende & Mohamed, 2015) as our generative model.

Secondly, we develop novel methods for training normalizing flows that obey complicated high-dimensional PDEs, which are an excellent fit for approximating Q. We propose a training method, the stochastic *Euler-KL* method, which is based on the forward discretization of the differential equation for Q and the Kullback-Lieber matching of probability distributions. Our normalizing flows approach can also be equipped with the Time-Dependent Variational Principle (TDVP) method (McMillan, 1965), which can be derived from the Euler method and can be thought of as an analog of the natural gradient method (Amari, 1996; 1998).

We name our contributions *Q-Flow* (see Figure 1). Q-Flow is a new approach to solving open quantum systems based on off-the-shelf normalizing flows and the Euler/TDVP methods for evolving such flows in complicated PDEs. We demonstrate that Q-Flow is scalable and efficient for simulating various open quantum systems. Our contributions can be summarized as follows:

- A new generative modeling approach for open quantum dynamics with continuous degrees of freedom based on the Husimi Q function, which allows for using normalizing flows off the shelf.
- New methods for solving open quantum dynamics PDEs using normalizing flows with stochastic Euler-KL method and TDVP.
- Demonstration of the scalability and efficiency of our methods on simulations of dissipative harmonic oscillator and dissipative bosonic models by surpassing conventional PDE solvers and state-of-the-art machine learning PDE solvers, physics-informed neural networks (PINN).

Importantly, with Q-Flow, the difficulty in simulating quantum dynamics is no longer the dimension of the simulation but instead the complexity of the Q function and its evolution, which opens a new avenue for research.

2. Related Work

2.1. Neural Network Quantum States

Neural network quantum states are generative neural network architectures—including restricted Boltzmann machines (Carleo & Troyer, 2017), autoregressive models (Sharir et al., 2020; Luo et al., 2022a; Chen et al., 2022; Luo et al., 2021), and determinant neural network models (Pfau et al., 2020; Hermann et al., 2020; Luo & Clark, 2019)—that have been adapted to represent quantum wave functions or density matrices (in the case of open quantum systems) rather than probability distributions. They are optimized using variational quantum Monte Carlo methods and have primarily been applied to model discrete spin systems (Carleo & Troyer, 2017; Sharir et al., 2020; Luo et al., 2022a) as well as tackle the continuous many-body wave function in quantum chemistry applications (Pfau et al., 2020; Hermann et al., 2020) and quantum field theories (Luo et al., 2022c; Martyn et al., 2022).

In contrast with prior deep learning-based approaches that directly model the wave function or density matrix, our work focuses on the Q function representation of the quantum state—a continuous quasiprobability distribution (Carmichael, 1999b) that can be modeled using an appropriate generative model, e.g., normalizing flows.

2.2. Partial Differential Equation (PDE) Solvers

To model the dynamics of an open quantum system using the Q function formulation, we are required to solve a high-

dimensional PDE. By parameterizing the Q function using a normalizing flow, our approach can efficiently solve this PDE. For comparison, we benchmark our work against alternative PDE solvers.

Traditional PDE solvers struggle to handle high-dimensional PDEs due to the curse of dimensionality, where storing the state of the system on a grid or mesh grows expontentially with the dimension of the problem. As traditional solver benchmarks, we use finite-difference and pseudo-spectral methods (Courant et al., 1928; Fornberg, 1998). While there are specialized methods for solving high-dimensional PDEs, they are often complex to set up and only apply to a few restricted classes of PDEs, e.g., parabolic PDEs (Weinan et al., 2021). From such specialized methods, we benchmark against a Stochastic method (Martin et al., 2016).

We also benchmark against physics-informed neural networks (PINNs)—a promising deep learning-based variational approach for solving PDEs (Raissi, 2018; Raissi et al., 2019; Berg & Nyström, 2019). PINNs, however, have been shown to have limitations related to the difficulty of the variational optimization problem (Krishnapriyan et al., 2021) and, in their standard form, may also suffer from the curse of dimensionality.

3. Solving Open Quantum Dynamics with Q-Flow

In this work, we develop Q-Flow, an approach to solving open quantum dynamics based on flow-based models under the Q function partial differential equation formulation. The key contributions of our work are twofold. Firstly, we establish a general framework for solving open quantum dynamics learning through the flow-based model representation. Secondly, we develop optimization algorithms for solving high dimensional partial differential equations and apply them to PDEs for the Q function. Note that a more thorough review of the relevant Quantum Mechanics is provided in Appendix A.1.

3.1. Quantum Overview

The fundamental mathematical object in quantum mechanics is a complex vector space known as the *Hilbert space*. It is customary to use the notation $|\cdot\rangle$, known as a *ket*, for vectors in the Hilbert space. We also denote the conjugate transpose of $|a\rangle$ as $\langle a|$, where $\langle\cdot|$ is known as a *bra*. The inner product of two kets $|a\rangle$ and $|b\rangle$ can be written as a *bra-ket* $\langle a|b\rangle$.

Operators on the Hilbert space can be thought of as complex-valued matrices. The most important operators are the Hamiltonian H, which governs the evolution of quantum systems, and the density matrix ρ, which describes the state of an open quantum system.

A particularly ubiquitous Hilbert space is that corresponding to particle number. The particle number Hilbert space has basis kets written $|n\rangle$, for $n \in \{0, 1, 2, \ldots\}$, where $|n\rangle$ represents a system with n particles. This Hilbert space is ubiquitous; it can also be used to represent many 1d bound systems. In particular, this Hilbert space appears in most bosonic and continuous quantum systems.

In the particle number Hilbert space, there are special operators, the *creation and annihilation operators*, which increase and decrease the number of particles, respectively. The creation operator $a^\dagger$ satisfies $a^\dagger |n\rangle = \sqrt{n+1}\,|n+1\rangle$ and the annihilation operator a satisfies $a\,|n\rangle = \sqrt{n}\,|n-1\rangle$ with $a\,|0\rangle = 0$.

The coherent state $|\alpha\rangle$ with a complex number α is defined as $|\alpha\rangle = e^{\alpha a^\dagger - \alpha^* a}\,|0\rangle$, where e should be interpreted as the matrix exponential function.

Hilbert spaces of systems with multiple subsystems are tensor products of the subsystems' Hilbert spaces. Suppose we have two Hilbert spaces, $\mathcal{H}_1$ and $\mathcal{H}_2$. For every two kets $|a\rangle \in \mathcal{H}_1$ and $|b\rangle \in \mathcal{H}_2$, there exists a ket $|a\rangle \otimes |b\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$, where $|a\rangle \otimes |b\rangle$ is the tensor product of $|a\rangle$ and $|b\rangle$ and $\mathcal{H}_1 \otimes \mathcal{H}_2$ is the tensor product space of Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$. The inner product for tensor products of Hilbert spaces is defined as $(\langle a| \otimes \langle b|)\,(|c\rangle \otimes |d\rangle) = \langle a|c\rangle\,\langle b|d\rangle$, where $\langle a| \otimes \langle b|$ is the conjugate transpose of $|a\rangle \otimes |b\rangle$. If an operator O_1 acts on $\mathcal{H}_1$ and O_2 acts on $\mathcal{H}_2$, then $(O_1 \otimes O_2)$ acts on $\mathcal{H}_1 \otimes \mathcal{H}_2$ according to $(O_1 \otimes O_2)\,|a\rangle \otimes |b\rangle = (O_1\,|a\rangle) \otimes (O_2\,|b\rangle)$. Finally, note that we often use shorthands such as O_1 or O_2 to referer to $O_1 \otimes 1$ or $1 \otimes O_2$, respectively.

3.2. Open Quantum System

As discussed in Section 1, in an open quantum system, the state is described as a complex-valued, unit-trace positive definite matrix ρ, known as the *density matrix*. The density matrix is a generalization of the wave function in the Schrödinger equation, which can be viewed as an ensemble of wave functions.

A generic Markovian open quantum system has an evolution equation of the form

$$\dot{\rho} = \mathcal{L}\rho = -i[H, \rho] + \mathcal{L}_{\text{loss}}\rho, \tag{1}$$

where H is the Hamiltonian matrix, $\mathcal{L}_{\text{loss}}$ is a dissipative operator, and $[\cdot, \cdot]$ is the commutation operator between matrices, i.e., $[A, B] = AB - BA$. Often, H is composed of *raising and lowering operators*, a and $a^\dagger$. Here, $\mathcal{L}$ is a *superoperator*; given a matrix ρ, it returns a new matrix $\mathcal{L}\rho$. Eq. 1 is a complex-valued high-dimensional differential equation, which is challenging to solve in general.

Our work applies to open quantum systems with continuous degrees of freedom. Such systems include bosonic systems,

which arise in a variety of contexts (Cazalilla et al., 2011; Adesso et al., 2014). A bosonic particle, also known as a *boson*, is a type of fundamental particle in quantum mechanics that has continuous degrees of freedom. Bosonic systems may be composed of multiple *sites*, which are subsystems described by the particle number Hilbert space.

Simulating quantum systems with continuous variables introduces higher-dimensional complexity compared to those with discrete variables, such as spin systems. Even for a 1-site continuous variable system, there is infinite degree of freedom. In practice, one workaround is to truncate the infinite degree of freedom to some large finite degree N. Even with truncation, k sites live in an exponentially-large-dimensional Hilbert space of size N^k, which is generally intractable to simulation. In contrast, our approach works with the infinite degree-of-freedom Hilbert space directly.

3.3. Q Function Formulation

The Husimi Q function (Carmichael, 1999a) provides an exact reformulation of Eq. 1 into a probabilistic differential equation:

$$\dot{Q} = \tilde{\mathcal{L}} Q \tag{2}$$

where Q is the Husimi Q function, and $\tilde{\mathcal{L}}$ is the Q-function evolution operator including the effects of H and $\mathcal{L}_{\text{loss}}$.

Mathematically, the Q function of n sites is defined as

$$Q(\vec{q}, \vec{p}) = Q(\vec{\alpha}, \vec{\alpha}^*) = \frac{1}{\pi} \langle \vec{\alpha} | \rho | \vec{\alpha} \rangle ,$$

where $\vec{\alpha} = \vec{q} + i\vec{p}$ is a complex number, and

$$|\vec{\alpha}\rangle = |\alpha_1\rangle \otimes \cdots \otimes |\alpha_n\rangle$$

is a tensor product of coherent states.[1] $Q(\vec{\alpha}, \vec{\alpha}^*) \geq 0$ for any $\vec{\alpha}$ and $\int Q = 1$, so Q can be interpreted as a probability distribution in practice.

We use the notations $Q(\vec{q}, \vec{p})$ and $Q(x)$ interchangeably.

To use the Q function formalism, we must convert between the ρ and Q functions and obtain $\tilde{\mathcal{L}}$. We provide the key conversion formulas and the corresponding proofs in Appendix A.

3.4. Q-Flow representation: Flow-based Generative Models of Q function

One important feature of our work is to represent the Q function with off-the-shelf flow-based generative models. This distinguishes our work from previous works (Vicentini et al., 2019; Yoshioka & Hamazaki, 2019; Hartmann & Carleo, 2019; Nagy & Savona, 2019) that represent the high dimensional complex-valued density matrix using customized neural networks. There are several advantages of our approach: i) we do not work with complex-valued functions, which could be complicated by the sign structure problem (Westerhout et al., 2020). ii) Q-Flow is natural for systems with continuous degrees of freedom. iii) Q-Flow allows normalized probability modeling with exact sampling, which is important for solving high dimensional probabilistic PDEs with the stochastic Euler method.

Normalizing Flows. Normalizing flows are generative models for continuous probability distributions that provide both normalized probabilities and exact sampling—making them ideal for modeling the continuous Q function in our approach. Normalizing flows transform a simple initial density p_X (often a unit-normal distribution) to a target density p_Y (i.e., the distribution that we want to model) via a sequence of invertible transformations (Dinh et al., 2014; Rezende & Mohamed, 2015). The invertible transformations are usually parameterized by an invertible neural network architecture $y = f_\theta(x)$ with $x \sim p_X$ and $y \sim p_Y$. The target probability density is then given by

$$p_Y(y) = p_X(f_\theta^{-1}(y)) \left| \frac{\partial f_\theta^{-1}(y)}{\partial y} \right| .$$

Many choices of f_θ are available, including affine coupling layers (RealNVP) (Dinh et al., 2017), continuous normalizing flows (CNF) (Grathwohl et al., 2019), and convex potential flows (CP-Flow) (Huang et al., 2021). While RealNVP is the simplest to implement, affine coupling layers are less expressive than CNFs or CP-Flows, which are provably universal density estimators (Huang et al., 2021). Because of Equation 12, we would like our flow to be infinitely differentiable, which is satisfied by the above flow architectures.

Theorem 3.1. *For a Q function from a given density matrix ρ, there exists a universal approximation with a Q-Flow representation.*

Proof. For any given density matrix ρ, there is a corresponding Q_ρ which satisfies $Q_\rho \geq 0$ and $\int Q_\rho = 1$. Since it has been shown that normalization flow is a universal approximator of probability distribution (Huang et al., 2021), there exists a Q-Flow representation Q_f such that it can be arbitrarily close to Q_ρ. □

Theorem 3.2. *For any local observable expected value to be computed with respect to ρ, there exists a Q-Flow representation which can compute the observable efficiently.*

Proof. We prove the single-site case here and the multi-site case follows from the tensor product structure of the Hilbert space. Consider the corresponding Q function Q_ρ of ρ. Consider a local observable in the form of $O = a^m a^{\dagger n} + a^n a^{\dagger m}$. WLOG, we can consider $O = a^m a^{\dagger n}$ and the other part can

[1] Although α and α^* are both input to Q, it is customary in physics and complex analysis to write $Q(\alpha, \alpha^*)$ instead of $Q(\alpha)$.

be done in a similar way. its expectation$\langle O\rangle_\rho = \mathrm{tr}(\rho a^m a^{\dagger n})$. Eq. A.5 in Appendix shows that it can be equivalently computed by $\int (q+ip)^m (q-ip)^n Q_\rho(p,q)dpdq$, which is a polynomial moment of the Q function. Since normalization flow is a universal approximator, there exists a Q-Flow representation Q_f can be arbitrarily close to Q_ρ, which implies that $\langle Q\rangle_f$ can be arbitrarily close to $\langle Q\rangle_\rho$. Even though computing $\langle Q\rangle_f = \sum_{(q,p)\sim Q_f}(q+ip)^m(q-ip)^n$ has stochastic fluctuation, the exact sampling nature of the flow-based model can suppress the statistical error, which will decay with increasing sample size N_s as $\frac{1}{\sqrt{N_s}}$ due to the Central Limit Theorem. □

3.5. Q-Flow Optimization: Stochastic Euler-KL Method

In the previous section, we discuss the representation of the Q function with flow-based models. To solve the real-time dynamics given by Eq. 2, we further develop the high dimensional stochastic Euler-KL method.

The algorithm represents the Q function at time t with a flow-based model and iteratively updates the representation at the next time $t+dt$ based on the Euler method. It requires two copies of flow-based models for Q^{t+dt} and Q^t. Based on the first-order Euler method with time step dt, Eq. 2 yields

$$Q^{t+dt} = Q^t + \tilde{\mathcal{L}}Q^t dt = (\mathrm{I} + \tilde{\mathcal{L}}dt)Q^t \equiv Q^t_{\mathcal{L}}. \quad (3)$$

Notice that Q^{t+dt} represents the Q function that we obtain in the next time step. At each learning step, we fix Q^t and optimize the parameters θ in Q^{t+dt} to match the above relation. Hence, we also denote Q^{t+dt} by Q^{t+dt}_θ. We train Q^{t+dt}_θ using the KL divergence loss function

$$KL(Q^{t+dt}_\theta || Q^t_{\mathcal{L}}) = \int Q^{t+dt}_\theta \ln \frac{Q^{t+dt}_\theta}{Q^t_{\mathcal{L}}}. \quad (4)$$

The gradient of Eq. 4 can be derived with a control variance technique as follows (see Appendix for a derivation):

$$\frac{1}{N}\sum_{x\sim Q^{t+dt}_\theta}\left[\ln\frac{Q^{t+dt}_\theta(x)}{Q^t_{\mathcal{L}}(x)} - b\right]\nabla_\theta \ln Q^{t+dt}_\theta(x) \quad (5)$$

where $b = \frac{1}{N}\sum_{x\sim Q^{t+dt}_\theta}\ln\frac{Q^{t+dt}_\theta(x)}{Q^t_{\mathcal{L}}(x)}$ is the baseline for control variance.

The stochastic Euler-KL method is summarized in Algorithm. 1. We further provide an error bound by developing the analysis in Gutiérrez & Mendl (2022) to Q-Flow.

Theorem 3.3. *The global error $\epsilon(t_n)$ of the n-step stochastic Euler method is bounded by $|\epsilon_E(t_n)| + |\epsilon_{NN}(t_n)|$, where $\epsilon_E(t_n)$ is the global error of the exact Euler method and $\epsilon_{NN}(t_n) = -P^{-1}\sum_{i=1}^n P^i r^{n+1-i}$ with $P = \mathrm{I} + \tilde{\mathcal{L}}dt$ and r^i being the i-th step stochastic Euler optimization error with neural network representation of the Q-Flow.*

Algorithm 1 Stochastic Euler-KL Method

Input: normalizing flow models for Q^{t+dt}_θ and Q^t, total time T, time step dt, n_{iter}, optimizer Adam.
Output: Optimal parameters $\boldsymbol{\theta}^*$ at time step $t+dt$
Initialization: Random $\boldsymbol{\theta}(t_0)$
for j in range(T/dt) **do**
  for $i = 0$ to n_{iter} **do**
    update θ using Eq. 5 and optimizer Adam
  end for
  $Q^t \leftarrow Q^{t+dt}_{\theta^*}$
end for

Proof. $\epsilon(t_n) = Q^{t_n} - Q^{t_n}_{NN} = (Q^{t_n} - Q^{t_n}_E) + (Q^{t_n}_E - Q^{t_n}_{NN}) \equiv \epsilon_E(t_n) + \epsilon_{NN}(t_n)$, where $Q^{t_n}_E$ and $Q^{t_n}_{NN}$ are the Q function from the exact Euler method and the neural network Q-Flow at time step t_n. By the triangular inequality, $|\epsilon(t_n)| \le |\epsilon_E(t_n)| + |\epsilon_{NN}(t_n)|$. Since the Euler method is a first-order method, it has global error of order $O(dt)$ where dt the time step.

Denote the optimization error of Eq. 4 in time step t_{n+1} as r^{n+1}, such that $Q^{t_{n+1}}_{NN} - PQ^{t_n}_{NN} = r^{n+1}$. It follows that $Q^{t_{n+1}}_E - \epsilon_{NN}(t_{n+1}) - P(Q^{t_n}_E - \epsilon_{NN}(t_n)) = r^{n+1}$, which implies that $\epsilon_{NN}(t_{n+1}) = P\epsilon_{NN}(t_n) - r^{n+1}$ due to the cancellation of $Q^{t_{n+1}}_E - PQ^{t_n}_E$ from the exact Euler method. By induction, $\epsilon_{NN}(t_n) = -P^{-1}\sum_{i=1}^n P^i r^{n+1-i}$. □

Time Dependent Variational Principle (TDVP). Instead of taking the gradient with respect to the KL divergence as Eq. 5 shows, Reh & Gärttner (2022) demonstrate that the minimization of Eq. 4 is equivalent to the time-dependent variational principle, which provides a nonlinear differential equation on the parameter space θ as follows.

$$S_{kk'}\dot{\theta}_{k'} = F_k \quad (6)$$

where $S_{kk'} = \mathbb{E}[(\partial_{\theta_k}\ln Q)(\partial_{\theta'_k}\ln Q)]$ is the Fisher information matrix, and $F_k = \mathbb{E}[(\partial_{\theta_k}\ln Q)(\partial_t \ln Q)]$ with $\partial_t \ln Q = (\partial_t Q)/Q = (\tilde{\mathcal{L}}Q)/Q$.

Reh & Gärttner (2022) has only applied TDVP to solving classical PDEs. Under our Q-Flow approach, we can also apply TDVP to simulate open quantum dynamics.

Complexity Analysis. Even though the stochastic Euler-KL method and the TDVP method are equivalent mathematically, they share different algorithmic complexity. TDVP requires solving the nonlinear differential equation in Eq. 6, which requires explicitly inverting the Fisher information matrix $S_{kk'}$. Besides potential instability, this procedure has complexity scaling as $O(N^3)$ for explicit inversion, or $O(N^2)$ with the conjugate gradient approach, where N is the number of parameters. This may limit its application for parameters beyond the orders of ten thousands. Meanwhile, the stochastic Euler method only requires first order optimization based on Eq. 5, the main cost of which comes from the number of optimization steps in each dt.

Time	Q-Flow Euler (ours)	Q-Flow TDVP (ours)	PINN	PS	FD
1-site					
3	2.08e-3	5.11e-3	1.79e-1	**3.47e-4**	8.90e-4
6	5.10e-4	1.17e-3	1.84e-1	**3.47e-4**	9.01e-4
9	**1.01e-4**	2.16e-4	1.91e-1	3.47e-4	9.01e-4
12	**1.68e-5**	3.58e-5	1.91e-1	3.47e-4	9.01e-4
15	1.58e-5	**5.55e-6**	1.98e-1	3.47e-4	9.01e-4
2-site					
3	**3.91e-3**	1.23e-2	1.00e0	1.83e-1	6.12e-2
6	**1.91e-3**	4.66e-3	1.00e0	1.82e-1	6.09e-2
9	**7.59e-4**	1.77e-3	1.00e0	1.81e-1	6.09e-2
12	**2.92e-4**	6.21e-4	1.00e0	1.81e-1	6.09e-2
15	**1.47e-4**	2.05e-4	1.00e0	1.81e-1	6.09e-2
20-site					
3	**9.94e-2**	1.08e-1	2.17e31	-	-
6	**3.29e-2**	4.10e-2	2.38e30	-	-
9	**2.02e-2**	2.44e-2	1.34e29	-	-
12	**1.46e-2**	1.68e-2	1.46e28	-	-
15	**1.07e-2**	1.23e-2	7.07e26	-	-

Table 1. $L_1[Q_{\text{sim}}, Q_{\text{exact}}]$ for each simulation method over time. For each row, we mark the best result in bold.

3.6. Q-Flow Initialization: Initial State Pretraining

Using a Q-Flow to simulate a quantum system requires initializing the flow to the correct starting Q function. For some simple initial states, we find that it is sufficient to simply make the initial state the prior for the flow and initialize the flow to the identity. However, we find that using more complex initial distributions as priors to a flow tends to hamper their ability to model a system's evolution. In these cases, we instead use the standard Gaussian prior, but we use a two-step process to pretrain the flow to match the initial distribution Q_{init}.

First, we sample from the desired initial distribution using the Metropolis-Hastings Monte Carlo method and update the flow parameters to minimize the negative log-likelihood $-\sum_{x\sim Q_{\text{init}}} \ln Q_\theta(x)$. This ensures that the model has some overlap with Q_{init}, which helps the next step's training algorithm converge more quickly.

Second, we sample from the flow and update the flow parameters to minimize the KL Loss, $KL(Q_{\text{init}}||Q_\theta)$. We compute the gradient according to

$$\nabla_\theta KL \approx -\frac{1}{N}\sum_{x\sim Q_\theta} \frac{Q_{\text{init}}(x)}{Q_\theta(x)} \nabla_\theta \ln Q_\theta(x). \tag{7}$$

4. Experiments

For our experiments, we focus on two types of open quantum systems: dissipative harmonic oscillators and dissipative bosonic systems. We test on dissipative harmonic oscillators because they have an analytic solution, which makes them useful for benchmarking high-dimensional PDE solvers beyond the limits of conventional solvers. We then test on dissipative bosonic systems because they are commonly studied and of practical use in physics.

In these experiments, we compare Euler and TDVP methods to PINNs, Pseudo-spectral solvers, Finite Difference solvers, and stochastic solvers. Although we do not develop the TDVP method, we propose a method to apply it to open bosonic quantum systems. As such, we sometimes describe the Euler and TDVP methods as "our methods."

For our experiments, we use Affine Coupling Flows and Convex-Potential Flows for the Euler and TDVP methods. Affine Coupling Flows are fast but less expressive, so we use them for the dissipative harmonic oscillator experiments. Convex Potential Flows are slow but more expressive, so we use them for problems involving more complex Q functions.

To run our experiments (Dugan et al., 2023), we use the Jax library (Bradbury et al., 2018) for Euler and TDVP methods. We make use of the jax-flows library. To implement the TDVP method, we make use of the NetKet library (Carleo et al., 2019; Vicentini et al., 2022) and its Stochastic Reconfiguration (Sorella, 1998; 2001) feature, which is mathematically equivalent to TDVP. For distributed training, NetKet uses the mpi4jax package (Häfner & Vicentini, 2021). For PINNs, we use the PINA library, which is built on top of PyTorch. Finally, for the other three baselines we use Julia (Rackauckas & Nie, 2017).

Further explanation of the observables chosen and their significance can be found in Appendix A.6. More details about the Normalizing Flow models we use are provided in Appendix C. More details about experimental setup, hyperparameters, and baselines are provided in Appendix D.

4.1. Dissipative Harmonic Oscillator

Experimental Setup. The multi-site dissipative harmonic oscillator evolves according to Equation 1 with Hamiltonian (Carmichael, 1999a) $H = \sum_j \omega_j a_j^\dagger a_j$ and loss term

$$\begin{aligned}\mathcal{L}_{\text{loss}}\rho = \sum_j \gamma_j \Big[&\frac{1}{2}(2a_j\rho a_j^\dagger - a_j^\dagger a_j\rho - \rho a_j^\dagger a_j) \\ &+ \bar{n}_j(a_j\rho a_j^\dagger + a_j^\dagger\rho a_j - a_j^\dagger a_j\rho - \rho a_j a_j^\dagger)\Big].\end{aligned} \tag{8}$$

Here, j labels what we will call *sites*. Converting to the Q function formalism gives (Carmichael, 1999b)

$$\begin{aligned}\tilde{\mathcal{L}} = \sum_j \Bigg[&\gamma_j + \frac{1}{4}\gamma_j(\bar{n}_j+1)\left(\frac{\partial^2}{\partial q_j^2} + \frac{\partial^2}{\partial p_j^2}\right) \\ &+ \left(\frac{\gamma_j}{2}q_j - \omega_j p_j\right)\frac{\partial}{\partial q_j} + \left(\frac{\gamma_j}{2}p_j + \omega_j q_j\right)\frac{\partial}{\partial p_j}\Bigg].\end{aligned}$$

We test the simulation methods on three problems of increasing dimensionality: a 1-site system, a 2-site system,

Time	Q-Flow (Euler)	Q-Flow (TDVP)	PINN
3	$\mathbf{1.30 \cdot 10^{-7}}$	$8.18 \cdot 10^{-7}$	$2.43 \cdot 10^{-3}$
6	$\mathbf{7.64 \cdot 10^{-9}}$	$4.06 \cdot 10^{-8}$	$2.66 \cdot 10^{-3}$
9	$\mathbf{3.16 \cdot 10^{-10}}$	$1.38 \cdot 10^{-9}$	$2.68 \cdot 10^{-3}$
12	$\mathbf{7.49 \cdot 10^{-12}}$	$3.79 \cdot 10^{-11}$	$2.34 \cdot 10^{-3}$
15	$4.30 \cdot 10^{-12}$	$\mathbf{9.21 \cdot 10^{-13}}$	$1.89 \cdot 10^{-3}$

Table 2. L_2 loss for each simulation method's density matrix over time for the 1-site system. We mark each row's best result in bold.

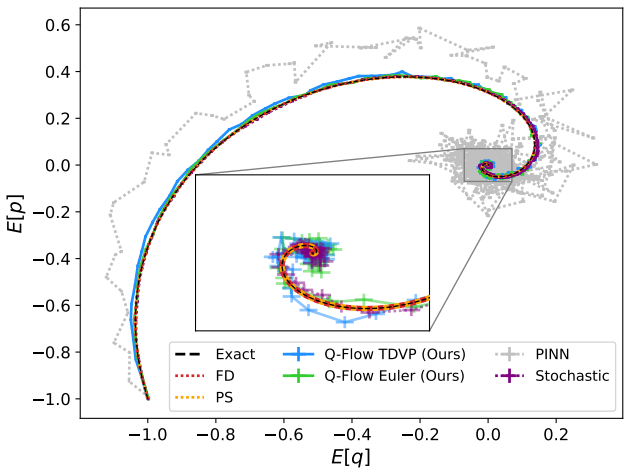

Figure 2. The trajectory of the centroids of the simulated distributions. The PINN baseline is excluded from the inset. Error bars are included for all but the FD and PS methods but are small.

and a 20-site system. For each system, we use a coherent state initial condition, which corresponds to a Gaussian with variance $1/2$. We center the Gaussian at $(-1, \ldots, -1)$. As time passes this Gaussian spirals toward the origin and changes its standard deviation. To make the simulation more challenging, for every site j we uniformly sample the system's parameters $\bar{n}_j \in [3, 7)$, $\gamma_j \in [0.5, 1.5)$, and $\omega_j \in [0.5, 1.5)$. See Appendix D.1 for more details about the choice of system parameters.

Metrics. To evaluate performance, we compute the L_1 Loss between each simulation and the exact distribution:

$$L_1[Q_{\text{sim}}, Q_{\text{exact}}] \equiv \int \mathrm{d}^d x \, |Q_{\text{sim}}(x) - Q_{\text{exact}}(x)| \approx \frac{1}{N} \sum_{x \sim Q_{\text{exact}}} \left| \frac{Q_{\text{sim}}(x)}{Q_{\text{exact}}(x)} - 1 \right|. \tag{9}$$

Although the L_1 Loss is a useful metric, it is also illustrative to examine observables of the system. One observable is the centroid, $\mathbb{E}[\vec{x}] \approx \frac{1}{N} \sum_{x \sim Q_{\text{sim}}} \vec{x}$. With more sites, we cannot easily plot the centroid trajectory, so instead we compute the centroid's distance from the origin, $\|\mathbb{E}[\vec{x}]\|$.

Additionally, we compute the *Liouvillian loss*,

$$\int dx \, \left| [\tilde{\mathcal{L}} Q](x) \right| = \mathbb{E}[|\tilde{\mathcal{L}} Q| / Q].$$

The Liouvillian loss measures the magnitude of the dynamics relative to the distribution, an indicator of how perturbed the system is from equilibrium.

For the Euler and TDVP methods, we sample directly from the flow to compute expected values. For PINNs, we use Markov chain Monte Carlo (MCMC) to obtain samples. For pseudo-spectral results, we compute expected values by summing over the grid and scaling by Q.

Finally, for the 1-site system, we compute the first 4x4 block of the density matrix according to Equation 12 and compute its L_2 distance from the exact density matrix:

$$L_2[\rho_{\text{pred}}, \rho_{\text{exact}}] = \sum_{1 \le i,j \le 4} |(\rho_{\text{pred}})_{ij} - (\rho_{\text{exact}})_{ij}|^2 . \tag{10}$$

Because the density matrix is the standard parametrization of a quantum system, this comparison is another useful benchmark for performance. Equation 12 requires spatial derivatives of the Q-function, so we only compute this loss for solvers that return spatially differentiable Q functions.

Results and Discussion. Table 1 shows the L_1 Loss between each simulation and the exact distribution for a number of simulation times. Although we do not include error bounds in the table for ease of viewing, the error is usually at least an order of magnitude smaller than the L_1 Loss (see Appendix. E). Error bounds for the pseudo-spectral and finite-difference results (standard solvers) are not computed because these methods are deterministic. We exclude standard solvers from the 20-site system because a grid size of only 10 would require storing at least 10^{40} values.

Both the Euler and TDVP methods have extremely low L_1 Loss. Both methods perform better than the standard solvers in the 2-site case and in the later times of the 1-site case. Increasing the number of sites, we find that the Euler and TDVP methods continue to perform well while PINNs and standard solvers struggle. Standard solvers cannot simulate the 20-site system due to the curse of dimensionality, and while PINNs can in principle simulate the system, in practice they perform extremely poorly. On the other hand, both the Euler and TDVP methods still consistently report low fidelities. Finally, note that the Euler method has a consistently lower L_1 loss than the TDVP method.

Figure 2 shows the trajectory of each simulation method's centroid for the 1-site case. Once again, the Euler and TDVP methods both closely match the exact evolution, and the Euler method performs slightly better in general. On the other hand, the PINN solution exhibits consistently biased and rapidly fluctuating estimates of the centroid. As

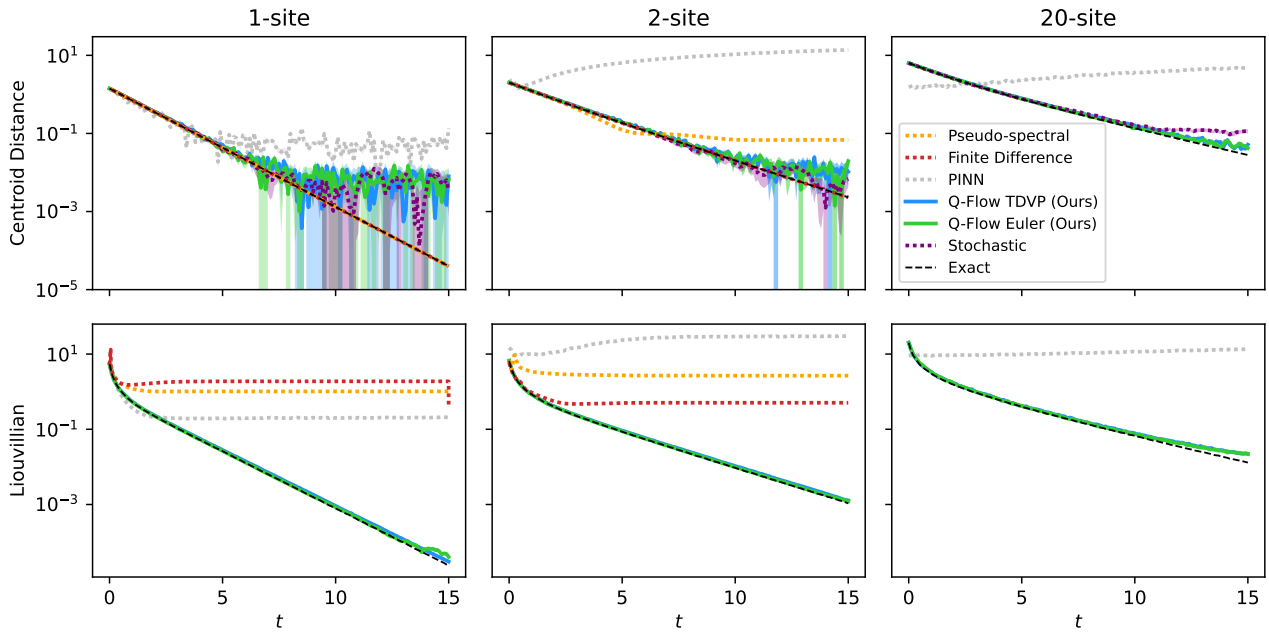

Figure 3. The simulated evolution of two observables for 1-site, 2-site, and 20-site dissipative harmonic oscillators. Error bars are included for all but the Finite Difference and Pseudo-spectral results but are small for most observables.

expected, the standard solvers closely track the exact trajectory. However, we note that although our methods appear to match the exact results less accurately, the large error bars in the cutout demonstrate that this is in large part due to sampling error. We could have computed the centroid for the Euler and TDVP methods using grid integration as with the pseudo-spectral method, but we instead choose to use sampling because this better generalizes to higher dimensions. The stochastic method performs comparably to our Euler and TDVP methods, but we note that it applies to a restricted subset of diffusion-type PDEs. Additionally, the stochastic method cannot provide exact values of the Q function, which makes it challenging to evaluate other observables such as the Liouvillian loss or L_1 and L_2 losses.

Figure 3 shows the evolution of the centroid distance and the Liouvillian Loss for all three problems. Euler and TDVP closely match the exact evolution of the two observables. Although the two methods' estimates of the centroid distance begin to diverge from the exact centroid distance at around time 10, once again the large error bars demonstrate that this is due to error in the sampling estimate. Although the PINN centroid distance also begins to diverge from the desired value, the small error bars for this estimate suggest that the deviation does not come from sampling error.

The Euler and TDVP methods' Liouvillian losses decrease consistently. At around time 15, the Euler Liouvillian loss jumps slightly. This jump occurs at a Liouvillian loss below 10^{-7}, so the simulation is still likely precise enough for most applications. The Euler method's performance can likely be improved by increasing the number of fitting steps per time step and by decreasing the step size. In practice, we find that decreasing the step size improves both the Euler and TDVP methods' performance. Interestingly, the standard solvers provide very poor estimates of the Liouvillian loss. We suspect that this is due to error in numerical derivatives.

Finally, note that unlike the other methods, our methods continue to correctly simulate the system for large numbers of sites. It is only toward the end of time evolution in the 20-site case that our methods begin to show some deviation from the exact observables. Again, this can likely be reduced by decreasing the step size and taking more samples. Interestingly, the stochastic method appears to diverge slightly more than our methods in the 20-site case.

Table 4.1 shows the L_2 loss from Equation 10. The Euler and TDVP methods have extremely low losses, with the Euler method performing slightly better. The PINN L_2 loss, while low, is much larger than the Euler and TDVP methods.

4.2. Dissipative Bosonic Model

Experimental Setup. The dissipative bosonic model is a frequently studied open quantum system (Kordas et al., 2013; Berg & Nyström, 2019). We test our methods on this model because it has a more complex evolution equation with rich real-world applicabilities.

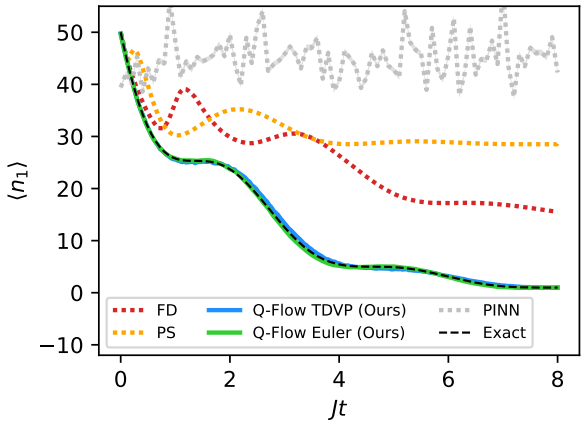

Figure 4. 2-site dissipative bosonic experiment: simulated $\langle n_1 \rangle$.

The dissipative bosonic model we use has (Kordas et al., 2013) $H = -J \sum_j \left(a_{j+1}^\dagger a_j + a_j^\dagger a_{j+1} \right)$ and

$$\mathcal{L}_{\text{loss}} \rho = -\frac{1}{2} \sum_j \gamma_j \left(n_j \rho + \rho n_j - 2 a_j \rho a_j^\dagger \right) \tag{11}$$

where $n_j = a_j^\dagger a_j$ and j enumerates the sites. Converting to the Q function formalism gives that $\tilde{\mathcal{L}}$ is

$$\sum_j \gamma_j \left(\frac{1}{4} \left(\frac{\partial^2}{\partial q_j^2} + \frac{\partial^2}{\partial p_j^2} \right) + \frac{1}{2} \left(q_j \frac{\partial}{\partial q_j} + p_j \frac{\partial}{\partial p_j} + 1 \right) \right)$$
$$+ J \sum_j \left(p_{j+1} \frac{\partial}{\partial q_j} - q_{j+1} \frac{\partial}{\partial p_j} + p_j \frac{\partial}{\partial q_{j+1}} - q_j \frac{\partial}{\partial p_{j+1}} \right).$$

Following Figure 3 of (Kordas et al., 2015), we consider a 2-site system with $J = 1$, $U = 0$, and $\gamma = [1, 0]$. We simulate the evolution of an antisymmetric Bose-Einstein Condensate (BEC) with 50 particles in each site, which has a Q function $Q(q_1, p_1, q_2, p_2)$ given by

$$Q = \frac{\left[(q_1 - q_2)^2 + (p_1 - p_2)^2 \right]^{100}}{\pi^2 \cdot 2^{100} \cdot 100!} e^{-(q_1^2 + p_1^2 + q_2^2 + p_2^2)}.$$

Because of the complex multimodal initial distribution, we use the Convex Potential Flow for these experiments. We pretrain the flow as described in Section 3.6.

Metric. For this system, we compute the observable $\langle n_1 \rangle \approx \frac{1}{N} \sum_{(\vec{q}, \vec{p}) \sim Q_{\text{sim}}} (q_1^2 + p_1^2 - 1)$ because it's exact evolution is given in (Kordas et al., 2015).

Results and Discussion. We show the simulated evolution of $\langle n_1 \rangle$ in Figure 4. The existence of the J term is responsible for the oscillations shown because it causes the two sites to exchange particles, which could cause challenges for simulations. Even so, both the Euler and TDVP methods closely match the exact evolution, demonstrating the wide applicability of our methods.

5. Conclusion

In this work, we made an important contribution to the problem of simulating open quantum systems. We used a reformulation of the density matrix to the Husimi Q function, which allowed us to study open quantum systems as an evolution of a probability distribution under dynamics, described by a partial differential equation that we derive for each system. This allowed us to establish a direct connection between simulating continuous or bosonic open quantum systems and the rich literature on generative models in standard machine learning. With off-the-shelf normalizing flows, Affine Coupling Flows and Convex Potential Flows, and a new efficient method for solving high-dimensional PDEs, Euler-KL, we established Q-Flow, a new and efficient approach to simulation of open quantum systems.

We compared Q-Flow to the state-of-the-art numerical and deep learning approaches on two important systems to the field, the dissipative harmonic oscillator and dissipative bosonic models. We established superior performance across the board, especially for large system dimensionality.

We believe the significance of our results is twofold. On one hand, Q-Flow's accurate simulation of open quantum systems can be further developed to aid progress in fundamental physics and engineering applications, such as superconductors and quantum computers. On the other hand, through our reformulation from evolving the density matrix to evolving the Q function, we shifted the modeling challenges from the curse of dimensionality to the accurate evolution of a high-dimensional deep generative model. Q-Flow can aid progress in evolving probability distributions under PDE dynamics and inspire future work on deep generative models.

6. Acknowledgements

The authors acknowledge helpful discussions with Zhuo Chen and Matija Medvidović. The authors acknowledge support from the National Science Foundation under Cooperative Agreement PHY-2019786 (The NSF AI Institute for Artificial Intelligence and Fundamental Interactions, `http://iaifi.org/`). This material is based upon work supported by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers, Co-design Center for Quantum Advantage (C2QA) under contract number DE-SC0012704. This work is also work supported in part by the Air Force Office of Scientific Research under the award number FA9550-21-1-0317. P.Y. Lu is grateful for the support of the Eric and Wendy Schmidt AI in Science Postdoctoral Fellowship at the University of Chicago.

References

Adesso, G., Ragy, S., and Lee, A. R. Continuous variable quantum information: Gaussian states and beyond. *Open Systems & Information Dynamics*, 21(01n02):1440001, 2014.

Amari, S.-i. Neural learning in structured parameter spaces-natural riemannian gradient. *Advances in neural information processing systems*, 9, 1996.

Amari, S.-I. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Barreiro, J. T., Müller, M., Schindler, P., Nigg, D., Monz, T., Chwalla, M., Hennrich, M., Roos, C. F., Zoller, P., and Blatt, R. An open-system quantum simulator with trapped ions. *Nature*, 470(7335):486–491, 2011.

Berg, J. and Nyström, K. Data-driven discovery of PDEs in complex datasets. *Journal of Computational Physics*, 384:239–252, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2019.01.036. URL http://www.sciencedirect.com/science/article/pii/S0021999119300944.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Carleo, G. and Troyer, M. Solving the quantum many-body problem with artificial neural networks. *Science*, 355 (6325):602–606, 2017. doi: 10.1126/science.aag2302.

Carleo, G., Choo, K., Hofmann, D., Smith, J. E. T., Westerhout, T., Alet, F., Davis, E. J., Efthymiou, S., Glasser, I., Lin, S.-H., Mauri, M., Mazzola, G., Mendl, C. B., van Nieuwenburg, E., O'Reilly, O., Théveniaut, H., Torlai, G., Vicentini, F., and Wietek, A. Netket: A machine learning toolkit for many-body quantum systems. *SoftwareX*, pp. 100311, 2019. doi: 10.1016/j.softx.2019.100311. URL http://www.sciencedirect.com/science/article/pii/S2352711019300974.

Carmichael, H. J. *Dissipation in Quantum Mechanics: The Master Equation Approach*, pp. 1–28. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999a. ISBN 978-3-662-03875-8. doi: 10.1007/978-3-662-03875-8_1. URL https://doi.org/10.1007/978-3-662-03875-8_1.

Carmichael, H. J. *Quantum—Classical Correspondence for the Electromagnetic Field II: P, Q, and Wigner Representations*, pp. 101–145. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999b. ISBN 978-3-662-03875-8. doi: 10.1007/978-3-662-03875-8_4. URL https://doi.org/10.1007/978-3-662-03875-8_4.

Cazalilla, M. A., Citro, R., Giamarchi, T., Orignac, E., and Rigol, M. One dimensional bosons: From condensed matter systems to ultracold gases. *Reviews of Modern Physics*, 83(4):1405, 2011.

Chen, Z., Luo, D., Hu, K., and Clark, B. K. Simulating 2+ 1d lattice quantum electrodynamics at finite density with neural flow wavefunctions. *arXiv preprint arXiv:2212.06835*, 2022.

Courant, R., Friedrichs, K., and Lewy, H. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100:32–74, 1928. doi: 10.1007/BF01448839. URL https://doi.org/10.1007/BF01448839.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

Dugan, O., Lu, P. Y., Dangovski, R., Luo, D., and Soljačić, M. The code repository is going to be made public in arxiv and the related materials are available on reasonable request from the corresponding author. *arXiv preprint arXiv:2302.12235*, 2023.

Fornberg, B. *A practical guide to pseudospectral methods*. Number 1. Cambridge university press, 1998.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., and Duvenaud, D. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJxgknCcK7.

Gutiérrez, I. L. and Mendl, C. B. Real time evolution with neural-network quantum states. *Quantum*, 6:627, 2022.

Hartmann, M. J. and Carleo, G. Neural-network approach to dissipative quantum many-body dynamics. *Phys. Rev. Lett.*, 122:250502, Jun 2019. doi: 10.1103/PhysRevLett.122.250502. URL https://link.aps.org/doi/10.1103/PhysRevLett.122.250502.

Hendrycks, D. and Gimpel, K. Gaussian Error Linear Units (GELUs). *arXiv e-prints*, art. arXiv:1606.08415, June 2016. doi: 10.48550/arXiv.1606.08415.

Hermann, J., Schätzle, Z., and Noé, F. Deep-neural-network solution of the electronic schrödinger equation. *Nature Chemistry*, 12(10):891–897, Oct 2020. ISSN 1755-4349. doi: 10.1038/s41557-020-0544-y.

Huang, C.-W., Chen, R. T. Q., Tsirigotis, C., and Courville, A. Convex potential flows: Universal probability distributions with optimal transport and convex optimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=te7PVH1sPxJ.

Häfner, D. and Vicentini, F. mpi4jax: Zero-copy mpi communication of jax arrays. *Journal of Open Source Software*, 6(65):3419, 2021. doi: 10.21105/joss.03419. URL https://doi.org/10.21105/joss.03419.

Kingma, D. P. and Dhariwal, P. Glow: Generative Flow with Invertible 1x1 Convolutions. *arXiv e-prints*, art. arXiv:1807.03039, July 2018. doi: 10.48550/arXiv.1807.03039.

Kordas, G., Wimberger, S., and Witthaut, D. Decay and fragmentation in an open bose-hubbard chain. *Phys. Rev. A*, 87:043618, Apr 2013. doi: 10.1103/PhysRevA.87.043618. URL https://link.aps.org/doi/10.1103/PhysRevA.87.043618.

Kordas, G., Witthaut, D., Buonsante, P., Vezzani, A., Burioni, R., Karanikas, A. I., and Wimberger, S. The dissipative Bose-Hubbard model. *European Physical Journal Special Topics*, 224(11):2127–2171, November 2015. doi: 10.1140/epjst/e2015-02528-2.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 26548–26560. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/df438e5206f31600e6ae4af72f2725f1-Paper.pdf.

Luo, D. and Clark, B. K. Backflow transformations via neural networks for quantum many-body wave functions. *Physical review letters*, 122(22):226401, 2019.

Luo, D., Chen, Z., Hu, K., Zhao, Z., Hur, V. M., and Clark, B. K. Gauge invariant autoregressive neural networks for quantum lattice models. *arXiv preprint arXiv:2101.07243*, 2021.

Luo, D., Chen, Z., Carrasquilla, J., and Clark, B. K. Autoregressive neural network for simulating open quantum systems via a probabilistic formulation. *Phys. Rev. Lett.*, 128:090501, Feb 2022a. doi: 10.1103/PhysRevLett.128.090501.

Luo, D., Chen, Z., Carrasquilla, J., and Clark, B. K. Autoregressive neural network for simulating open quantum systems via a probabilistic formulation. *Physical review letters*, 128(9):090501, 2022b.

Luo, D., Yuan, S., Stokes, J., and Clark, B. K. Gauge equivariant neural networks for 2+ 1d u (1) gauge theory simulations in hamiltonian formulation. *arXiv preprint arXiv:2211.03198*, 2022c.

Martin, R. M., Reining, L., and Ceperley, D. M. *Interacting Electrons: Theory and Computational Approaches*. Cambridge University Press, 2016. doi: 10.1017/CBO9781139050807.

Martyn, J. M., Najafi, K., and Luo, D. Variational neural-network ansatz for continuum quantum field theory. *arXiv preprint arXiv:2212.00782*, 2022.

McMillan, W. L. Ground state of liquid he 4. *Physical Review*, 138(2A):A442, 1965.

Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. Monte carlo gradient estimation in machine learning. *The Journal of Machine Learning Research*, 21(1):5183–5244, 2020.

Nagy, A. and Savona, V. Variational quantum monte carlo method with a neural-network ansatz for open quantum systems. *Phys. Rev. Lett.*, 122:250501, Jun 2019. doi: 10.1103/PhysRevLett.122.250501. URL https://link.aps.org/doi/10.1103/PhysRevLett.122.250501.

Nocedal, J. and Wright, S. J. (eds.). *Large-Scale Quasi-Newton and Partially Separable Optimization*, pp. 222–249. Springer New York, New York, NY, 1999. ISBN 978-0-387-22742-9. doi: 10.1007/0-387-22742-3_9. URL https://doi.org/10.1007/0-387-22742-3_9.

Pfau, D., Spencer, J. S., Matthews, A. G. D. G., and Foulkes, W. M. C. Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Phys. Rev. Res.*, 2:033429, Sep 2020. doi: 10.1103/PhysRevResearch.2.033429.

Rackauckas, C. and Nie, Q. Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1):15, 2017.

Raissi, M. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research*, 19(25):1–24, 2018. URL http://jmlr.org/papers/v19/18-046.html.

Raissi, M., Perdikaris, P., and Karniadakis, G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL http://www.sciencedirect.com/science/article/pii/S0021999118307125.

Reh, M. and Gärttner, M. Variational monte carlo approach to partial differential equations with neural networks. *Machine Learning: Science and Technology*, 3(4):04LT02, 2022.

Reh, M., Schmitt, M., and Gärttner, M. Time-dependent variational principle for open quantum systems with artificial neural networks. *Physical Review Letters*, 127(23): 230501, 2021.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Shankar, R. *Principles of quantum mechanics*. Springer Science & Business Media, 2012.

Sharir, O., Levine, Y., Wies, N., Carleo, G., and Shashua, A. Deep autoregressive models for the efficient variational simulation of many-body quantum systems. *Phys. Rev. Lett.*, 124:020503, Jan 2020. doi: 10.1103/PhysRevLett.124.020503.

Sorella, S. Green function monte carlo with stochastic reconfiguration. *Physical review letters*, 80(20):4558, 1998.

Sorella, S. Generalized lanczos algorithm for variational quantum monte carlo. *Physical Review B*, 64(2):024512, 2001.

Verstraete, F., Wolf, M. M., and Ignacio Cirac, J. Quantum computation and quantum-state engineering driven by dissipation. *Nature physics*, 5(9):633–636, 2009.

Vicentini, F., Biella, A., Regnault, N., and Ciuti, C. Variational neural-network ansatz for steady states in open quantum systems. *Phys. Rev. Lett.*, 122: 250503, Jun 2019. doi: 10.1103/PhysRevLett.122.250503. URL https://link.aps.org/doi/10.1103/PhysRevLett.122.250503.

Vicentini, F., Hofmann, D., Szabó, A., Wu, D., Roth, C., Giuliani, C., Pescia, G., Nys, J., Vargas-Calderón, V., Astrakhantsev, N., and Carleo, G. NetKet 3: Machine Learning Toolbox for Many-Body Quantum Systems. *SciPost Phys. Codebases*, pp. 7, 2022. doi: 10.21468/SciPostPhysCodeb.7. URL https://scipost.org/10.21468/SciPostPhysCodeb.7.

Weinan, E., Han, J., and Jentzen, A. Algorithms for solving high dimensional pdes: from nonlinear monte carlo to machine learning. *Nonlinearity*, 35(1):278, 2021.

Westerhout, T., Astrakhantsev, N., Tikhonov, K. S., Katsnelson, M. I., and Bagrov, A. A. Generalization properties of neural network approximations to frustrated magnet ground states. *Nature communications*, 11(1):1593, 2020.

Yoshioka, N. and Hamazaki, R. Constructing neural stationary states for open quantum many-body systems. *Phys. Rev. B*, 99:214306, Jun 2019. doi: 10.1103/PhysRevB.99.214306. URL https://link.aps.org/doi/10.1103/PhysRevB.99.214306.

Zhuang, J., Tang, T., Ding, Y., Tatikonda, S., Dvornek, N., Papademetris, X., and Duncan, J. S. AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients. *arXiv e-prints*, art. arXiv:2010.07468, October 2020. doi: 10.48550/arXiv.2010.07468.

A. Further Details about the Q Function Formalism

A.1. Quantum Preliminaries

Here, we provide a brief and intuitive introduction to the theory of bosonic systems. We intentionally simplify most of the definitions and focus on the important concepts to our study. For an in-depth discussion, please refer to (Shankar, 2012).

We will discuss a few important terms that we use throughout the main text.

Hilbert Space. In a closed system, i.e., one which is insulated from the environment, each subsystem's state can be described as a unit vector in a complex vector space $\mathbb{C}^n$, a Hilbert space, for some dimension n which we will from now on denote as the *Hilbert space dimension* or the *degrees of freedom*.

Open Quantum Systems In an open quantum system, interactions with the environment introduce additional uncertainty about the quantum state of the system. To model open systems, we must thus resort to the *density matrix*. The density matrix is an $n \times n$ positive definite unit-trace complex-valued matrix, where n is the Hilbert space dimension of the system. The space $\mathbb{C}^{n\times n}$ of density matrices is sometimes known as the *double Hilbert space*. This space is spanned by the set of outer products $|b_1\rangle \langle b_2|$ of basis vectors. The density matrix can be thought of as an operator on the Hilbert space.

Braket notation. Such notation is used throughout the text to denote quantum states. Quantum states are elements of a complex vector space V, equipped with a Hermitian form. In our work we use the standard Hermitian inner product, which in math notation is $(\mathbf{v}, \mathbf{w}) = \mathbf{v}^\dagger \mathbf{w}$. Here $\dagger$ denotes the complex conjugate, for any two vectors $\mathbf{v}, \mathbf{w} \in V$. In physics notation, we write $\mathbf{v}$ as $|\mathbf{v}\rangle$ (known as a *ket*) and likewise for $\mathbf{w}$. We also use the notation $\langle \mathbf{v}| \equiv \mathbf{v}^\dagger$, and call this a *bra*. Then, $\mathbf{v}^\dagger \mathbf{w}$ can be written $\langle \mathbf{v}| |\mathbf{w}\rangle$, or more concisely as $\langle \mathbf{v}|\mathbf{w}\rangle$. Furthermore, $|\mathbf{v}\rangle \langle \mathbf{w}|$ denotes the outer product of $\mathbf{v}$ and $\mathbf{w}^\dagger$.

Particle number Hilbert space, Vacuum states, the Fock space. There exists a special Hilbert space known as the particle number Hilbert space or the *Fock space*. This Hilbert space describes a location, such as a potential well, with varying number of particles. It is spanned by a countably infinite set of orthonormal basis vectors, which we label $|0\rangle, |1\rangle, \ldots$, where $|n\rangle$ represents a system with n particles. To represent a general element of $|0\rangle, |1\rangle, \ldots$, we will use a Roman letter inside the ket or bra. We denote $|0\rangle$ as the *vaccum state* because it represents a system with no particles.

Although a system with varying number of particles can be described by the particle number Hilbert space, there are many other systems that can be similarly described. For example, a particle confined to move in a 1d potential well can be described by this Hilbert space. In this paper, we use the term *site* to refer to any system with a Hilbert space that is the particle number Hilbert space. For multiple sites, the total Hilbert space is the tensor product of each particle's Hilbert space.

Creation, annihilation operators and Coherent state. The creation operator $a^\dagger$ satisfies $a^\dagger |n\rangle = \sqrt{n+1}\, |n+1\rangle$ and the annihilation operator a satisfies $a|n\rangle = \sqrt{n}\, |n-1\rangle$ with $a|0\rangle = 0$. The coherent state $|\alpha\rangle$ with a complex number α is defined as $|\alpha\rangle = e^{\alpha a^\dagger - \alpha^* a} |0\rangle$, where e should be interpreted as matrix exponential function. A more practical equivalent definition of the coherent state is

$$|\alpha\rangle = e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle .$$

Compute observables. In quantum mechanics, a density matrix ρ can be expressed as $\rho = \sum_{n,m} \rho_{n,m} |n\rangle \langle m|$ and an observable O can be expressed as $O = \sum_{n,m} O_{n,m} |n\rangle \langle m|$, where both ρ and O can be viewed as Hermitian matrices. It follows that the expectation value of the observable $\langle O\rangle = \mathrm{tr}(\rho O) = \sum_{n,m} O_{n,m}\rho_{m,n}$.

Tensor products. Suppose we have two Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$. For every two kets $|a\rangle \in \mathcal{H}_1$ and $|b\rangle \in \mathcal{H}_2$, there exists a ket $|a\rangle \otimes |b\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$, where $|a\rangle \otimes |b\rangle$ is the tensor product of $|a\rangle$ and $|b\rangle$ and $\mathcal{H}_1 \otimes \mathcal{H}_2$ is the tensor product space $\mathcal{H}_1$ and $\mathcal{H}_2$. The inner product for tensor products of Hilbert spaces is defined as $(\langle a| \otimes \langle b|)(|c\rangle \otimes |d\rangle) = \langle a|c\rangle \langle b|d\rangle$, where $\langle a| \otimes \langle b|$ is the conjugate transpose of $|a\rangle \otimes |b\rangle$. From this it is clear that the basis kets for the new Hilbert space are all pairs of tensor products of basis kets of the two smaller Hilbert spaces. If an operator O_1 acts on $\mathcal{H}_1$ and O_2 acts on $\mathcal{H}_2$, then $(O_1 \otimes O_2)$ acts on $\mathcal{H}_1 \otimes \mathcal{H}_2$ according to $(O_1 \otimes O_2)|a\rangle \otimes |b\rangle = (O_1|a\rangle) \otimes (O_2|b\rangle)$. Finally, note that we often use shorthands such as O_1 or O_2 to refer to $O_1 \otimes 1$ or $1 \otimes O_2$, respectively.

A.2. Q Function to ρ

In this section, we show that for a given $Q(\alpha, \alpha^*)$, the density matrix ρ corresponding to it is given by

$$\langle m| \rho |n\rangle = \pi\sqrt{m!n!} \sum_{k=0}^{\min(m,n)} \frac{Q_{m-k,n-k}}{k!}, \tag{12}$$

where

$$Q_{a,b}(\alpha, \alpha^*) = \frac{1}{a!b!} \frac{\partial^{a+b}}{\partial^a \alpha \, \partial^b \alpha^*} Q(\alpha, \alpha^*)\Big|_{\alpha=\alpha^*=0}.$$

This result generalizes to multi-site Q functions using the tensor product structure, but for simplicity we consider only a single site here.

From expressing $\langle\alpha|$ and $|\alpha\rangle$ in terms of Harmonic Oscillator eigenstates, we have that

$$\begin{aligned}
Q(\alpha, \alpha^*) &= \frac{1}{\pi} e^{-\alpha\alpha^*} \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \frac{\langle m| \rho |n\rangle}{\sqrt{m!n!}} \alpha^{*m}\alpha^n \\
&= \frac{1}{\pi}\left(\sum_{s=0}^{\infty} \frac{(-1)^s}{s!}(\alpha\alpha^*)^s\right) \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \frac{\langle m| \rho |n\rangle}{\sqrt{m!n!}} \alpha^{*m}\alpha^n \\
&= \frac{1}{\pi}\sum_{s=0}^{\infty}\sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \frac{(-1)^s}{s!} \frac{\langle m| \rho |n\rangle}{\sqrt{m!n!}} (\alpha^*)^{m+s} \alpha^{n+s}
\end{aligned}$$

To determine $\langle m| \rho |n\rangle$, we must thus invert this series. However, since we know the correct form, we can simply substitute Equation 12 into the expression above and show that it correctly gives $Q(\alpha, \alpha^*)$:

$$\begin{aligned}
&\frac{1}{\pi}\sum_{s=0}^{\infty}\sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \frac{(-1)^s}{s!} \frac{\langle m| \rho |n\rangle}{\sqrt{m!n!}} (\alpha^*)^{m+s} \alpha^{n+s} \\
=&\frac{1}{\pi}\sum_{s=0}^{\infty}\sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \frac{(-1)^s}{s!} \frac{\pi\sqrt{m!n!}\sum_{k=0}^{\min(m,n)} \frac{Q_{m-k,n-k}}{k!}}{\sqrt{m!n!}} (\alpha^*)^{m+s} \alpha^{n+s} \\
=&\sum_{s=0}^{\infty}\sum_{m=0}^{\infty}\sum_{n=0}^{\infty}\sum_{k=0}^{\min(m,n)} \frac{(-1)^s}{s!} \frac{Q_{m-k,n-k}}{k!} (\alpha^*)^{m+s} \alpha^{n+s}.
\end{aligned}$$

Setting $a = m + s$ and $b = n + s$ gives

$$\sum_{a=0}^{\infty}\sum_{b=0}^{\infty}\sum_{s=0}^{\min(a,b)}\sum_{k=0}^{\min(a,b)-s} \frac{(-1)^s}{s!} \frac{Q_{a-k-s,b-k-s}}{k!} (\alpha^*)^a \alpha^b.$$

Then, setting $d = s + k$ gives

$$\begin{aligned}
&\sum_{a=0}^{\infty}\sum_{b=0}^{\infty}\sum_{d=0}^{\min(a,b)}\sum_{s=0}^{d} \frac{(-1)^s}{s!} \frac{Q_{a-d,b-d}}{(d-s)!} (\alpha^*)^a \alpha^b \\
=&\sum_{a=0}^{\infty}\sum_{b=0}^{\infty} (\alpha^*)^a \alpha^b \sum_{d=0}^{\min(a,b)} Q_{a-d,b-d} \sum_{s=0}^{d}(-1)^s \binom{d}{s}.
\end{aligned}$$

Now, by the Binomial Theorem, $\sum_{s=0}^{d}(-1)^s\binom{d}{s} = (1-1)^d = 0^d$, which is 0 unless $d = 1$. So, we get

$$\begin{aligned}
&\sum_{a=0}^{\infty}\sum_{b=0}^{\infty} Q_{a,b}(\alpha, \alpha^*) \cdot (\alpha^*)^a \alpha^b \\
=&\sum_{a=0}^{\infty}\sum_{b=0}^{\infty} \frac{(\alpha^*)^a \alpha^b}{a!b!} \frac{\partial^{a+b}}{\partial^a\alpha\partial^b\alpha^*} Q(\alpha, \alpha^*) \\
=&Q(\alpha, \alpha^*),
\end{aligned}$$

as desired. The last step comes from the Taylor series representation of Q, which is only valid if Q is analytic. So as long as Q is analytic, this result holds.

A.3. Coherent State Identities

Here we present a few coherent state identities that prove useful in A.4.

$$
\begin{aligned}
a^\dagger |\alpha\rangle &= a^\dagger e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \\
&= e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} \sqrt{n+1}\, |n+1\rangle \\
&= e^{-|\alpha|^2/2} \frac{\partial}{\partial \alpha} \sum_{n=0}^{\infty} \frac{\alpha^{n+1}}{\sqrt{(n+1)!}} |n+1\rangle \\
&= e^{-|\alpha|^2/2} \frac{\partial}{\partial \alpha} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \\
&= e^{-|\alpha|^2/2} \frac{\partial}{\partial \alpha} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \\
&= \frac{\partial}{\partial \alpha} e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle - \left(\frac{\partial}{\partial \alpha} e^{-|\alpha|^2/2} \right) \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \\
&= \frac{\partial}{\partial \alpha} e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle - \left(\frac{\partial}{\partial \alpha} e^{-|\alpha|^2/2} \right) \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \\
&= \frac{\partial}{\partial \alpha} e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle + \frac{\alpha^*}{2} e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \\
&= \left(\frac{\alpha^*}{2} + \frac{\partial}{\partial \alpha} \right) e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \\
&= \left(\frac{\alpha^*}{2} + \frac{\partial}{\partial \alpha} \right) |\alpha\rangle .
\end{aligned}
$$

Similarly,

$$
\langle \alpha | a = \left(\frac{\alpha}{2} + \frac{\partial}{\partial \alpha^*} \right) \langle \alpha | .
$$

Also,

$$
\frac{\partial}{\partial \alpha^*} |\alpha\rangle = \frac{\partial}{\partial \alpha^*} e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle = -\frac{\alpha}{2} e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle = -\frac{\alpha}{2} |\alpha\rangle \tag{13}
$$

and

$$
\frac{\partial}{\partial \alpha} \langle \alpha | = -\frac{\alpha^*}{2} \langle \alpha | .
$$

A.4. ρ Evolution to Q Function Evolution

In this section, we demonstrate how to convert to a general equation of motion for a density matrix to an equation of motion for the corresponding Q function using the tensor product structure. This result generalizes to multi-site Q functions, but for simplicity we consider only a single site here.

Note that

$$\langle\alpha| a^\dagger \hat{O}_1 \rho \hat{O}_2 |\alpha\rangle = \alpha^* \langle\alpha| \hat{O}_1 \rho \hat{O}_2 |\alpha\rangle ,$$
$$\langle\alpha| \hat{O}_1 \rho \hat{O}_2 a |\alpha\rangle = \alpha \langle\alpha| \hat{O}_1 \rho \hat{O}_2 |\alpha\rangle .$$

Also,

$$\begin{aligned}
\langle\alpha| a\hat{O}_1 \rho \hat{O}_2 |\alpha\rangle &= \left[\left(\frac{\alpha}{2} + \frac{\partial}{\partial\alpha^*}\right)\langle\alpha|\right] \hat{O}_1 \rho \hat{O}_2 |\alpha\rangle \\
&= \left(\frac{\alpha}{2} + \frac{\partial}{\partial\alpha^*}\right)\langle\alpha| \hat{O}_1 \rho \hat{O}_2 |\alpha\rangle - \langle\alpha| \hat{O}_1 \rho \hat{O}_2 \frac{\partial}{\partial\alpha^*} |\alpha\rangle \\
&= \left(\frac{\alpha}{2} + \frac{\partial}{\partial\alpha^*}\right)\langle\alpha| \hat{O}_1 \rho \hat{O}_2 |\alpha\rangle + \frac{\alpha}{2}\langle\alpha| \hat{O}_1 \rho \hat{O}_2 |\alpha\rangle \\
&= \left(\alpha + \frac{\partial}{\partial\alpha^*}\right)\langle\alpha| \hat{O}_1 \rho \hat{O}_2 |\alpha\rangle ,
\end{aligned}$$

and

$$\begin{aligned}
\langle\alpha| \hat{O}_1 \rho \hat{O}_2 a^\dagger |\alpha\rangle &= \langle\alpha| \hat{O}_1 \rho \hat{O}_2 \left(\frac{\alpha^*}{2} + \frac{\partial}{\partial\alpha}\right)|\alpha\rangle \\
&= \left(\frac{\alpha^*}{2} + \frac{\partial}{\partial\alpha}\right)\langle\alpha| \hat{O}_1 \rho \hat{O}_2 |\alpha\rangle - \left[\frac{\partial}{\partial\alpha}\langle\alpha|\right] \hat{O}_1 \rho \hat{O}_2 |\alpha\rangle \\
&= \left(\frac{\alpha^*}{2} + \frac{\partial}{\partial\alpha}\right)\langle\alpha| \hat{O}_1 \rho \hat{O}_2 |\alpha\rangle + \frac{\alpha^*}{2}\langle\alpha| \hat{O}_1 \rho \hat{O}_2 |\alpha\rangle \\
&= \left(\alpha^* + \frac{\partial}{\partial\alpha}\right)\langle\alpha| \hat{O}_1 \rho \hat{O}_2 |\alpha\rangle .
\end{aligned}$$

With these results, we now have that for an equation of the form

$$\dot{\rho} = \sum_{j,k,l,m} c_{j,k,l,m} (a^\dagger)^j a^k \rho (a^\dagger)^l a^m,$$

we can convert the the Q function equation of motion by inserting $\frac{1}{\pi}\langle\alpha|\ |\alpha\rangle$ to get

$$\begin{aligned}
\frac{1}{\pi}\langle\alpha| \dot{\rho} |\alpha\rangle &= \frac{1}{\pi}\sum_{j,k,l,m} c_{j,k,l,m} \langle\alpha| (a^\dagger)^j a^k \rho (a^\dagger)^l a^m |\alpha\rangle \\
\implies \dot{Q}(\alpha,\alpha^*) &= \frac{1}{\pi}\sum_{j,k,l,m} c_{j,k,l,m} (\alpha^*)^j \langle\alpha| a^k \rho (a^\dagger)^l a^m |\alpha\rangle \\
\implies \dot{Q}(\alpha,\alpha^*) &= \frac{1}{\pi}\sum_{j,k,l,m} c_{j,k,l,m} (\alpha^*)^j \left(\alpha + \frac{\partial}{\partial\alpha^*}\right)^k \langle\alpha| \rho (a^\dagger)^l a^m |\alpha\rangle \\
\implies \dot{Q}(\alpha,\alpha^*) &= \frac{1}{\pi}\sum_{j,k,l,m} c_{j,k,l,m} (\alpha^*)^j \left(\alpha + \frac{\partial}{\partial\alpha^*}\right)^k \alpha^m \langle\alpha| \rho (a^\dagger)^l |\alpha\rangle \\
\implies \dot{Q}(\alpha,\alpha^*) &= \frac{1}{\pi}\sum_{j,k,l,m} c_{j,k,l,m} (\alpha^*)^j \left(\alpha + \frac{\partial}{\partial\alpha^*}\right)^k \alpha^m \left(\alpha^* + \frac{\partial}{\partial\alpha}\right)^l \langle\alpha| \rho |\alpha\rangle \\
\implies \dot{Q}(\alpha,\alpha^*) &= \sum_{j,k,l,m} c_{j,k,l,m} (\alpha^*)^j \left(\alpha + \frac{\partial}{\partial\alpha^*}\right)^k \alpha^m \left(\alpha^* + \frac{\partial}{\partial\alpha}\right)^l Q(\alpha,\alpha^*).
\end{aligned}$$

A.5. Observable calculation with respect to Q function

In this section, we demonstrate how to efficiently compute observables by sampling from the Q function using the tensor product structure. This result generalizes to multi-site Q functions, but for simplicity we consider only a single site here.

Consider a general observable $\hat{O}$. Its expected value given a density matrix ρ is

$$\langle \hat{O} \rangle = \mathrm{Tr}\Big(\hat{O}\rho\Big). \tag{14}$$

Inserting the coherent state resolution of the identity, we get that

$$\begin{aligned}\mathrm{Tr}\Big(\hat{O}\rho\Big) &= \int \frac{d\alpha d\alpha^*}{\pi} \mathrm{Tr}\Big(\hat{O}\rho \left|\alpha\right\rangle \left\langle\alpha\right|\Big) \\ &= \int \frac{d\alpha d\alpha^*}{\pi} \left\langle\alpha\right| \hat{O}\rho \left|\alpha\right\rangle .\end{aligned}$$

Depending on the operator, it may be most useful to insert the resolution of the identity elsewhere.

Example The expected value of $a^m(a^\dagger)^n$ given a density matrix ρ is

$$\begin{aligned}\langle a^m \left(a^\dagger\right)^n \rangle &= \mathrm{Tr}\left(a^m \left(a^\dagger\right)^n \rho\right) \\ &= \int \frac{d\alpha d\alpha^*}{\pi} \mathrm{Tr}\Big(a^m \left|\alpha\right\rangle \left\langle\alpha\right| \left(a^\dagger\right)^n \rho\Big) \\ &= \int \frac{d\alpha d\alpha^*}{\pi} \left\langle\alpha\right| \left(a^\dagger\right)^n \rho a^m \left|\alpha\right\rangle \\ &= \int \frac{d\alpha d\alpha^*}{\pi} \alpha^m (\alpha^*)^n \left\langle\alpha\right| \rho \left|\alpha\right\rangle \\ &= \int dq dp\ (q+ip)^m (q-ip)^n Q(q,p).\end{aligned}$$

If $m \neq n$, this is not an observable, but could be made an observable by adding its Hermitian conjugate.

A.6. Choice of observables

The Liouvillian is chosen because the differential equation evolution is governed by the Liouvillian. In particular, as Eq. 2 shows that $\dot{Q} = LQ$, when the norm of LQ goes to zero, $\dot{Q}$ approaches zero which is the steady state of interest. Hence, the observable Liouvillian signifies how soon the system evolves to steady state. The centroid is chosen as an observable because it is the macroscopic observable that can be directly measured in the experiment. It behaves as the center of the mass of the system, which naturally connects to the classical limit and provides a good intuition and direct visualization on how the system evolves.

B. Stochastic Euler-KL Method

Here we derive Equation 5 for the control-variance gradient of the KL-Divergence. We start with

$$KL(Q_\theta^{t+dt} || Q_{\mathcal{L}}^t) = \int Q_\theta^{t+dt} \ln \frac{Q_\theta^{t+dt}}{Q_{\mathcal{L}}^t}.$$

Taking the gradient gives

$$\begin{aligned}\nabla_\theta KL(Q_\theta^{t+dt} || Q_{\mathcal{L}}^t) &= \nabla_\theta \int Q_\theta^{t+dt} \ln \frac{Q_\theta^{t+dt}}{Q_{\mathcal{L}}^t} \\ &= \int \left(\nabla_\theta Q_\theta^{t+dt}\right) \ln \frac{Q_\theta^{t+dt}}{Q_{\mathcal{L}}^t} + \int Q_\theta^{t+dt} \nabla_\theta \ln \frac{Q_\theta^{t+dt}}{Q_{\mathcal{L}}^t} \\ &= \int Q_\theta^{t+dt} \left(\nabla_\theta \ln Q_\theta^{t+dt}\right) \ln \frac{Q_\theta^{t+dt}}{Q_{\mathcal{L}}^t} + \int Q_\theta^{t+dt} \nabla_\theta \ln Q_\theta^{t+dt} \\ &= \int Q_\theta^{t+dt} \left[\ln \frac{Q_\theta^{t+dt}}{Q_{\mathcal{L}}^t} + 1\right] \nabla_\theta \ln Q_\theta^{t+dt}.\end{aligned}$$

Now, note that

$$\int Q_\theta^{t+dt}(x)\nabla_\theta \ln Q_\theta^{t+dt}(x) = \int Q_\theta^{t+dt}(x)\frac{\nabla_\theta Q_\theta^{t+dt}(x)}{Q_\theta^{t+dt}(x)} = \int \nabla_\theta Q_\theta^{t+dt}(x) = \nabla_\theta \int Q_\theta^{t+dt}(x) = \nabla_\theta 1 = 0. \tag{15}$$

So, letting

$$b = \int Q_\theta^{t+dt} \ln \frac{Q_\theta^{t+dt}(x)}{Q_{\mathcal{L}}^t(x)} \approx \frac{1}{N}\sum_{x\sim Q_\theta^{t+dt}} \ln \frac{Q_\theta^{t+dt}(x)}{Q_{\mathcal{L}}^t(x)}, \tag{16}$$

we can subtract a control variance to get

$$\begin{aligned}
\nabla_\theta KL(Q_\theta^{t+dt}||Q_{\mathcal{L}}^t) &= \int Q_\theta^{t+dt}\left[\ln\frac{Q_\theta^{t+dt}}{Q_{\mathcal{L}}^t}+1\right]\nabla_\theta \ln Q_\theta^{t+dt} \\
&= \int Q_\theta^{t+dt}\left[\ln\frac{Q_\theta^{t+dt}}{Q_{\mathcal{L}}^t}+1\right]\nabla_\theta \ln Q_\theta^{t+dt} - (b+1)\int Q_\theta^{t+dt}(x)\nabla_\theta \ln Q_\theta^{t+dt}(x) \\
&= \int Q_\theta^{t+dt}\left[\ln\frac{Q_\theta^{t+dt}}{Q_{\mathcal{L}}^t}-b\right]\nabla_\theta \ln Q_\theta^{t+dt} \\
&= \int Q_\theta^{t+dt}\left[\ln\frac{Q_\theta^{t+dt}}{Q_{\mathcal{L}}^t}-b\right]\nabla_\theta \ln Q_\theta^{t+dt}
\end{aligned}$$

Finally, approximating the integral gives

$$\begin{aligned}
\nabla_\theta KL(Q_\theta^{t+dt}||Q_{\mathcal{L}}^t) &= \int Q_\theta^{t+dt}\left[\ln\frac{Q_\theta^{t+dt}}{Q_{\mathcal{L}}^t}-b\right]\nabla_\theta \ln Q_\theta^{t+dt} \\
&\approx \frac{1}{N}\sum_{x\sim Q_\theta^{t+dt}}\left[\ln\frac{Q_\theta^{t+dt}}{Q_{\mathcal{L}}^t}-b\right]\nabla_\theta \ln Q_\theta^{t+dt},
\end{aligned}$$

as desired.

Similar technique on baseline control variance has been used in the context of reinforcement learning (Mohamed et al., 2020). It has been shown that it can reduce the variance of the gradient and helpful for the optimization.

C. Additional Normalizing Flow Implementation Details

C.1. Affine Coupling Flow

For our Affine Coupling models, we use the following architecture:

For each Affine Coupling layer, we split each input vector into two equal-sized vectors `v1 = input[:input.shape[0]//2]` and `v2 = input[input.shape[0]//2:]`. We then compute two neural networks $s = NN_1(\mathbf{W}_1, v_2)$ and $t = NN_2(\mathbf{W}_2, v_2)$. We then return the concatenation of $e^s v_1 + t$ and v_2.

The neural networks used in the Affine Coupling layers are fully-connected feed-forward neural networks. They have a set number of hidden layers of a fixed size. The input is fed into a linear layer with output size equal to the hidden layer size (usually 3) and then fed into a GELU nonlinearity (Hendrycks & Gimpel, 2016). Then, this output is concatenated with the original input vector. We refer to the output concatenated with the previous layer as the "augmented hidden layer." For each subsequent internal layer, we feed the previous augmented hidden layer into a linear layer with output size equal to the hidden layer size. We then feed this output into a GELU and concatenate the previous augmented hidden layer. Finally, for the final linear layer, the output has size equal to the input dimension of the neural network, and we do not apply a GELU or concatenate the previous augmented hidden layer.

The Affine Coupling flow is then constructed as follows: We use a unit Gaussian centered at the origin as our flow prior. To transform an input vector from the data coordinate system to the coordinate system of the prior, we apply an Affine Coupling

layer and then reverse the order of the input vector. We repeat this process a user specified number of times (usually 3). To transform an input vector from the prior coordinate system to the coordinate system of the data, we apply the inverse of the above transformation.

For the Affine Coupling model, the number of inputs and outputs of the neural networks defining the coupling transforms are equal to the dimension of the probability distribution. Increasing the number of sites will increase the number of inputs and outputs of these neural networks. We do not increase the dimensions of the hidden layers of the neural networks. However, at each hidden layer, we concatenate the previous hidden layer to the current one. As a result, the input is concatenated to every hidden layer, so the dimension of each hidden layer effectively increases. As a result, the number of parameters grows as a quadratic function of the number of sites currently. However, it is feasible to replace the concatenations with skip connection, so that the number of parameters in the hidden layers will stay constant as a function of the number of sites.

C.2. Convex Potential Flows

For our implementation of the Convex Potential Flows, we closely follow the methods described in (Huang et al., 2021). Our architecture is described below.

For our Input Convex Neural Network (ICNN), we follow the ICNN architecture given in section 5 of (Huang et al., 2021). In particular, given an input vector x, we use the following procedure to compute the output of the ICNN:

Following Huang et al. (2021), let L denote a linear layer, L^+ denote a linear layer with positive weights, and s denote a softplus. Also, let a denote an ActNorm layer, as defined in (Kingma & Dhariwal, 2018) and cat denote concatenation. We first compute $h = L(x)$. Then, for each layer in the network, we set

$$\tilde{h} = s(a(L^+(h) + L(x))) \tag{17}$$

$$h_{aug} = a(s(L(x))) \tag{18}$$

$$h = \text{cat}\left(\tilde{h}, h_{aug}\right). \tag{19}$$

. The output of the neural network is then

$$\text{out} = ICNN(x) = a(L^+(h) + L(x)). \tag{20}$$

Now, let $f(x) = s(a_1)\frac{||x||^2}{2} + s(a_2)ICNN(x)$, where the parameters of f are the parameters of a_1, a_2, and the parameters of $ICNN$. f is an input-convex function.

We use a unit Gaussian centered on the origin as our prior. To transform an input from the data distribution coordinates to the prior coordinates, we apply ∇f. To transform an input x from the prior distribution coordinates to the data coordinates, we use LGBFS (Nocedal & Wright, 1999) to find the y that minimizes

$$f(y) - x \cdot y.$$

For the Convex Potential Flow, the number of inputs to the convex neural network is equal to the dimension of the probability distribution, so increasing the number of sites will increase the number of inputs to the convex potential flow. The internal layer sizes all stay constant, so the only change is the input size. Thus, the number of parameters in the hidden layers is constant as a function of the number of sites and only the number of parameters in the first layer is linear in the number of sites.

D. Additional Experimental Details

D.1. Dissipative Harmonic Oscillator system parameters

As mentioned in the main text, we uniformly sample the dissipative harmonic oscillator system's parameters $\bar{n}_j \in [3, 7)$, $\gamma_j \in [0.5, 1.5)$, and $\omega_j \in [0.5, 1.5)$.

The values sampled are as follows:

For the 1-site system, the sampled parameters were

```
n_bar = [4.84872804]
gamma = [1.39682866]
omega_0 = [1.08564521],
```

for the 2-site system, the sampled parameters were

```
n_bar = [6.32870528, 5.24684123]
gamma = [0.85629208, 1.14026682]
omega_0 = [1.04436318, 0.75820899],
```

and for the 20-site system, the sampled parameters were

```
n_bar = [5.56599508, 3.65962215, 4.00906784, 4.18744726,
         6.60341254, 6.11329932, 4.84059527, 5.81464032,
         5.79018256, 3.86014355, 6.84045506, 4.05790151,
         6.92639748, 3.43788247, 4.17439805, 5.9303111,
         5.63412769, 4.52153322, 3.56601688, 4.38005014]

gamma = [0.71584443, 1.08686172, 1.40588976, 1.47121715,
         0.87775305, 0.78137437, 1.27848082, 1.14247345,
         0.95718403, 0.76484186, 1.22056516, 1.24775589,
         0.57332893, 1.06557609, 0.60105471, 1.32710909,
         0.90712674, 0.67560123, 0.98142727, 0.84515189]

omega_0 = [1.48982032, 0.52079238, 1.30285575, 0.89810077,
           1.35623683, 0.78528379, 1.0019163,  0.77954035,
           0.93705822, 1.32502792, 0.53982753, 0.67051701,
           0.95749435, 0.98833336, 0.86078757, 1.00879361,
           0.88153798, 1.05195061, 1.17483548, 1.1718404].
```

D.2. Pseudo-spectral and finite difference baseline details

As a baseline approach to solving the Q function evolution PDE (Eq. 2), we implement a pseudo-spectral and finite difference discretization of the PDE (Fornberg, 1998) in a square domain with $-10 < q_j < 10$ and $-10 < p_j < 10$ for each site j, set $Q = 0$ at the boundaries, and integrate using an adaptive Tsitouras 5/4 Runge-Kutta solver (Tsit5) (Rackauckas & Nie, 2017) while projecting at each time step to ensure the probability density Q remains positive and normalized. The psuedo-spectral method uses periodic boundary conditions and computes spatial derivatives using a fast Fourier transform. The finite difference method uses Dirichlet boundary conditions set at zero and computes spatial derivatives using the standard second-order finite difference stencil. We use a grid size of 256 grid points per dimension for 1-site and 32 grid points per dimension for 2-sites, resulting in a state size of $256^2 = 65{,}536$ for 1-site and $32^4 = 1{,}048{,}576$ for 2-sites. Note that for a fixed grid size, the state grows exponentially with the number of sites—i.e., the curse of dimensionality. This limits our ability to perform more fine-grained simulations on larger domains and makes this baseline approach intractable for more than a few sites.

D.3. PINN baseline details

We also use Physics Informed Neural Networks (PINNs) as a baseline. To implement this, we use the PINA library which is built on top of PyTorch. For each problem, we have two loss terms. The first computes the L_2 loss between the predicted initial distribution and the actual initial distribution for points sampled uniformly from within the domain of the solver at $t = 0$. The second computes the L_2 loss between the PINN time derivative and the predicted time derivative $\tilde{\mathcal{L}}Q$ at points sampled uniformly from within the spacial and temporal domain of the solver. The total loss is the sum of these two losses. We then optimize using gradient decent. Every 500 epochs we re-sample the points with which to compute the loss.

For each experiment, we use a fully connected feed forward neural network. The network has layer sizes of [input size, 40, 40, 40, 1]. For the first layer, we feed the input through a linear layer with output dimension 40 and then apply a GELU nonlinearity. We then concatenate the input. We denote a hidden layer with the previous layer concatenated the "augmented

hidden layer." For each subsequent layer, we take the input, feed it through a linear layer with output the size of the next hidden layer, apply GELU, and then concatenate the previous augmented linear layer. For the final layer, we apply a linear layer with output dimension 1 and do not apply GELU or concatenate.

The following are the hyperparameters used for each of the experiments:

- **1-site Harmonic Oscillator:** We use 1000 samples at a time for the initial condition and 50000 samples at a time for the derivative condition. We train for 25000 epochs with a learning rate of 0.001.
- **2-site Harmonic Oscillator:** We use 1000 samples at a time for the initial condition and 30000 samples at a time for the derivative condition. We train for 25000 epochs with a learning rate of 0.001.
- **20-site Harmonic Oscillator:** We use 3000 samples at a time for the initial condition and 3000 samples at a time for the derivative condition. We train for 50000 epochs with a learning rate of 0.001. Here, we have to decrease the number of samples for the derivative condition because of memory limits.
- **2-site Dissipative Bosonic Model:** We use 1000 samples at a time for the initial condition and 30000 samples at a time for the derivative condition. We train for 25000 epochs with a learning rate of 0.001.

D.4. Stochastic baseline details

Using the known Green's function for the N-site Harmonic Oscillator system (Carmichael, 1999b), we can construct a stochastic differential equation (SDE)

$$\begin{aligned} dq_i &= (-\gamma_i q_i/2 + \omega_0 p_i)\, dt + \sqrt{\gamma_i(\bar{n}_i + 1)/2}\, dW_{q_i} \\ dp_i &= (-\gamma_i p_i/2 - \omega_0 q_i)\, dt + \sqrt{\gamma_i(\bar{n}_i + 1)/2}\, dW_{p_i}, \end{aligned} \tag{21}$$

where dW_{q_i}, dW_{p_i} are independent Wiener processes with unit variance and $i \in \{1, \ldots, N\}$. Starting with samples from the initial Q function, this SDE generates samples q_i, p_i from Q at each time point, which can be used to compute simple observables, such as the centroid (i.e., the mean of the samples). Note that this approach does not explicitly provide the Q function and so cannot be used to compute observables involving Q or derivatives of Q. This method also only works for a limited set of systems whose evolution equations admit a stochastic description, e.g., a Fokker–Planck equation.

In our experiments, we use 100,000 sample points for the 1-site and 2-site Harmonic Oscillators and 10,000 sample points for the 20-site Harmonic Oscillator.

D.5. Flow Initialization details

As discussed in Section 4.2 of the main text, for the Bose Hubbard simulation, we initialize our Normalizing Flow models to the desired initial state of

$$Q = \frac{\left[(q_1 - q_2)^2 + (p_1 - p_2)^2\right]^{100}}{\pi^2 \cdot 2^{100} \cdot 100!} e^{-(q_1^2 + p_1^2 + q_2^2 + p_2^2)}.$$

To do this, we use the two pretraining methods described in Section 3.6. We describe our pretraining hyperparameters in more detail below. At any point below, if we mention sampling from the exact distribution, we do so using MCMC algorithms.

We first initialize the Convex Potential Flow ActNorm layers by providing 10000 samples from the exact distribution. Next, we perform 200 epochs of the following training algorithm (we use a learning rate of 1e-2):

1. Sample 1000 points from the exact distribution Q_{init}.
2. Compute the loss $L = -\sum_{x \sim Q_{\text{init}}} \ln Q_\theta(x)$.
3. Backpropagate to obtain $\nabla_\theta L$.
4. Take a gradient step using the Adabeleif optimizer (Zhuang et al., 2020).

After this, we perform 3000 training steps of the following training algorithm (we use a learning rate of 1e-3):

1. Sample 1000 points from the model distribution Q_θ.
2. Compute the gradient update
$$\nabla_\theta KL \approx -\frac{1}{N}\sum_{x\sim Q_\theta}\frac{Q_{\text{init}}(x)}{Q_\theta(x)}\nabla_\theta \ln Q_\theta(x). \tag{22}$$
3. Take a gradient step using the Adabeleif optimizer (Zhuang et al., 2020).

Note that for the dissipative harmonic oscillator systems, we use the exact initial state as the prior and initialize the normalizing flow transformation to be the identity. As such, we do not need to use pretraining for the dissipative harmonic oscillator systems.

D.6. Euler experiment details

Below are the hyperparameters we use for the Euler method. For the Harmonic Oscillator results, we use a 3 layer RealNVP where each affine transformation is a 2-hidden-layer feed-forward neural network with hidden layers of size 5. Instead of skip connections in the feed-forward neural network, we concatenate the previous activations at each activation layer. For the Dissipative Bosonic Model result, we use a Convex Potential Flow with a 5-hidden-layer input-convex neural network with hidden layers of size 20 and augmented layers of size 4, see (Huang et al., 2021).

- **1-site Harmonic Oscillator:** We train for 1500 steps with a step size of 0.01. For each step, we use the KL control variance loss to fit for 150 epochs with a learning rate of 0.001. We use 1000 samples per fitting epoch.
- **2-site Harmonic Oscillator:** We train for 1500 steps with a step size of 0.01. For each step, we use the KL control variance loss to fit for 150 epochs with a learning rate of 0.001. We use 1000 samples per fitting epoch.
- **20-site Harmonic Oscillator:** We train for 1500 steps with a step size of 0.01. For each step, we use the KL control variance loss to fit for 150 epochs with a learning rate of 0.001. We use 10000 samples per fitting epoch.
- **2-site Dissipative Bosonic Model:** We train for 400 steps with a step size of 0.02. For each step, we use the KL control variance loss to fit for 200 epochs with a learning rate of 0.002. We use 10000 samples per fitting epoch.

D.7. TDVP experiment details

Below are the hyperparameters we use for the TDVP method. For the Harmonic Oscillator results, we use a 3 layer RealNVP where each affine transformation is a 2-hidden-layer feed-forward neural network with hidden layers of size 5. Instead of skip connections in the feed-forward neural network, we concatenate the previous activations at each activation layer. For the Dissipative Bosonic Model result, we use a Convex Potential Flow with a 5-hidden-layer input-convex neural network with hidden layers of size 20 and augmented layers of size 4, see (Huang et al., 2021).

- **1-site Harmonic Oscillator:** We train for 1500 steps with a step size of 0.01. We use 1000 samples per step. We use a diagonal shift of 0.01.
- **2-site Harmonic Oscillator:** We train for 1500 steps with a step size of 0.01. We use 1000 samples per step. We use a diagonal shift of 0.01.
- **20-site Harmonic Oscillator:** We train for 1500 steps with a step size of 0.01. We use 10000 samples per step. We use a diagonal shift of 0.01.
- **2-site Dissipative Bosonic Model:** We train for 2000 steps with a step size of 0.004. We use 10000 samples per step. We use a diagonal shift of 0.01.

E. Additional Experimental Results

Figure 5 displays the L_1 divergence between the simulated Q function and the exact dissipative harmonic oscillator Q function for various simulation methods. Table 3 shows the same information as 1 but with errors included. These give additional information about the L_1 Loss evolution for the dissipative harmonic oscillator.

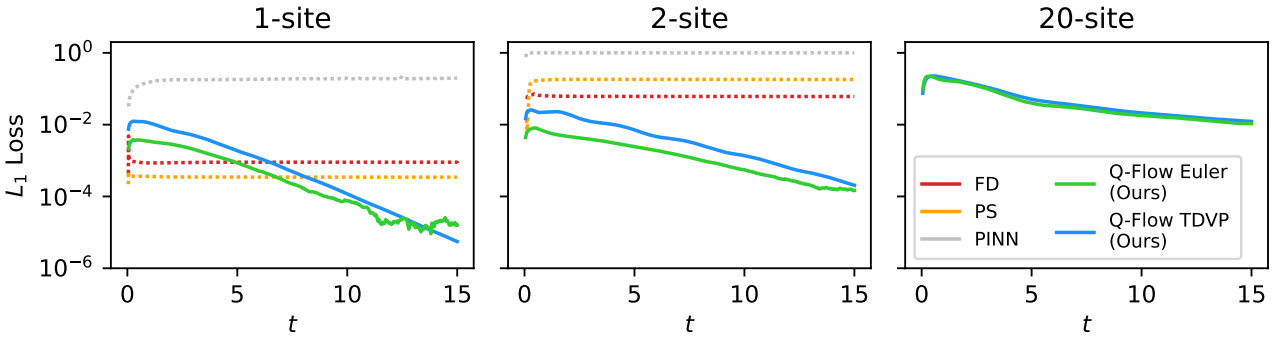

Figure 5. The L1 divergence between the simulated Q function and the exact Q function for 1-site, 2-site, and 20-site dissipative harmonic oscillators. Error bars are included for all but the finite difference and pseudo-spectral results but are small for most observables. In the 20-site case, it is not possible to run finite difference (FD) or pseudo-spectral methods (PS). Further, although the PINN method runs, it produces an L_1 loss on the order of 10^{30}, so we do not display it here.

Time	Q-Flow Euler (ours)	Q-Flow TDVP (ours)	PINN	PS	FD
		1-site			
3	$(2.08 \pm 0.01) \cdot 10^{-3}$	$(5.11 \pm 0.01) \cdot 10^{-3}$	$(1.79 \pm 0.01) \cdot 10^{-1}$	$\mathbf{3.47 \cdot 10^{-4}}$	$8.90 \cdot 10^{-4}$
6	$(5.10 \pm 0.01) \cdot 10^{-4}$	$(1.17 \pm 0.00) \cdot 10^{-3}$	$(1.84 \pm 0.01) \cdot 10^{-1}$	$\mathbf{3.47 \cdot 10^{-4}}$	$9.01 \cdot 10^{-4}$
9	$\mathbf{(1.01 \pm 0.00) \cdot 10^{-4}}$	$(2.16 \pm 0.01) \cdot 10^{-4}$	$(1.91 \pm 0.01) \cdot 10^{-1}$	$3.47 \cdot 10^{-4}$	$9.01 \cdot 10^{-4}$
12	$\mathbf{(1.68 \pm 0.01) \cdot 10^{-5}}$	$(3.58 \pm 0.01) \cdot 10^{-5}$	$(1.91 \pm 0.01) \cdot 10^{-1}$	$3.47 \cdot 10^{-4}$	$9.01 \cdot 10^{-4}$
15	$(1.58 \pm 0.01) \cdot 10^{-5}$	$\mathbf{(5.55 \pm 0.01) \cdot 10^{-6}}$	$(1.98 \pm 0.01) \cdot 10^{-1}$	$3.47 \cdot 10^{-4}$	$9.01 \cdot 10^{-4}$
		2-site			
3	$\mathbf{(3.91 \pm 0.01) \cdot 10^{-3}}$	$(1.23 \pm 0.00) \cdot 10^{-2}$	1.00 ± 0.00	$1.83 \cdot 10^{-1}$	$6.12 \cdot 10^{-2}$
6	$\mathbf{(1.91 \pm 0.00) \cdot 10^{-3}}$	$(4.66 \pm 0.01) \cdot 10^{-3}$	1.00 ± 0.00	$1.82 \cdot 10^{-1}$	$6.09 \cdot 10^{-2}$
9	$\mathbf{(7.59 \pm 0.02) \cdot 10^{-4}}$	$(1.77 \pm 0.00) \cdot 10^{-3}$	1.00 ± 0.00	$1.81 \cdot 10^{-1}$	$6.09 \cdot 10^{-2}$
12	$\mathbf{(2.92 \pm 0.01) \cdot 10^{-4}}$	$(6.21 \pm 0.01) \cdot 10^{-4}$	1.00 ± 0.00	$1.81 \cdot 10^{-1}$	$6.09 \cdot 10^{-2}$
15	$\mathbf{(1.47 \pm 0.00) \cdot 10^{-4}}$	$(2.05 \pm 0.00) \cdot 10^{-4}$	1.00 ± 0.00	$1.81 \cdot 10^{-1}$	$6.09 \cdot 10^{-2}$
		20-site			
3	$\mathbf{(9.94 \pm 0.03) \cdot 10^{-2}}$	$(1.08 \pm 0.00) \cdot 10^{-1}$	$(2.17 \pm 0.62) \cdot 10^{31}$	-	-
6	$\mathbf{(3.29 \pm 0.01) \cdot 10^{-2}}$	$(4.10 \pm 0.01) \cdot 10^{-2}$	$(2.38 \pm 0.81) \cdot 10^{30}$	-	-
9	$\mathbf{(2.02 \pm 0.01) \cdot 10^{-2}}$	$(2.44 \pm 0.01) \cdot 10^{-2}$	$(1.34 \pm 0.52) \cdot 10^{29}$	-	-
12	$\mathbf{(1.46 \pm 0.00) \cdot 10^{-2}}$	$(1.68 \pm 0.00) \cdot 10^{-2}$	$(1.46 \pm 1.19) \cdot 10^{28}$	-	-
15	$\mathbf{(1.07 \pm 0.00) \cdot 10^{-2}}$	$(1.23 \pm 0.00) \cdot 10^{-2}$	$(7.07 \pm 3.66) \cdot 10^{26}$	-	-

Table 3. $L_1[Q_{\text{sim}}, Q_{\text{exact}}]$ for each simulation method over time, with errors. For each row, we mark the best result in bold.