Inverse Design with Fourier Neural Operators for Quantum System Control

Anastasia Pipi
UCLA

Nivedha Gopinath
IIT Bombay

Valentin Duruisseaux
Caltech

Myrl G. Marmarelis
Caltech

Taylor L. Patti
NVIDIA

Brucek Khailany
NVIDIA

Prineha Narang
UCLA

Anima Anandkumar
Caltech

Abstract

Effective quantum control is essential for scalable quantum technologies, but existing methods face challenges in high-dimensional systems due to the high cost of conventional solvers. For example, molecular systems, which host many accessible states and serve as key platforms for precision measurements, remain difficult to control efficiently. To address this, we present a Fourier Neural Operator (FNO)-based framework that learns to simulate population dynamics of hyperfine states in CaH^+ from reference simulations. For the 6- and 12-dimensional systems considered, the FNO predicts complete trajectories accurately, given a constant driving frequency and initial state populations, while achieving 2 to 6 orders of magnitude speedups. These speedups are expected to grow further with higher system dimensionality, thereby opening the path to efficient simulations of previously intractable high-dimensional quantum systems. Leveraging the FNO's speed and differentiability, we successfully demonstrate two inverse-design strategies for controlling the driving frequency to steer a system toward a desired final state while minimizing evolution time: (i) extensive parameter space sampling across dense parameter grids, yielding accurate solutions, and (ii) gradient-based optimization, which demands more tuning but holds greater promise due to its substantially higher scalability. These results show that neural operators can provide both accurate forward simulation and scalable inverse design, opening new avenues for quantum control in complex systems.

1 Introduction

Optimal control of quantum systems has been central in advancing quantum technologies by enabling precise manipulation of quantum states. Widely used methods include gradient-based optimal control, such as GRAPE [1] and Krotov's algorithm [2], which have been applied to optimize operations in trapped-ion systems [3] and to design high-fidelity quantum gates [4]. Parameterized-pulse methods like CRAB [5] restrict the control field to a small set of basis functions. Despite recent advances, quantum control of high-dimensional systems remains computationally intensive, making real-time optimization challenging. While accurate numerical propagation can model such systems, it is too slow for scalable control and in some cases lack differentiability with respect to control parameters. Simulating dynamics that involve transitions across multiple timescales, such as optical carrier transitions and motional sidebands that differ by several orders of magnitude in frequency, introduces significant stiffness to the system. As a result, fine time steps are required during numerical propagation, which makes even moderately sized simulations (e.g., 50 internal states coupled with 2 motional degrees of freedom) computationally expensive. Consequently, the repeated trajectory evaluations needed for parameter optimization can take several hours on a typical desktop workstation.

Machine Learning and the Physical Sciences Workshop, NeurIPS 2025.

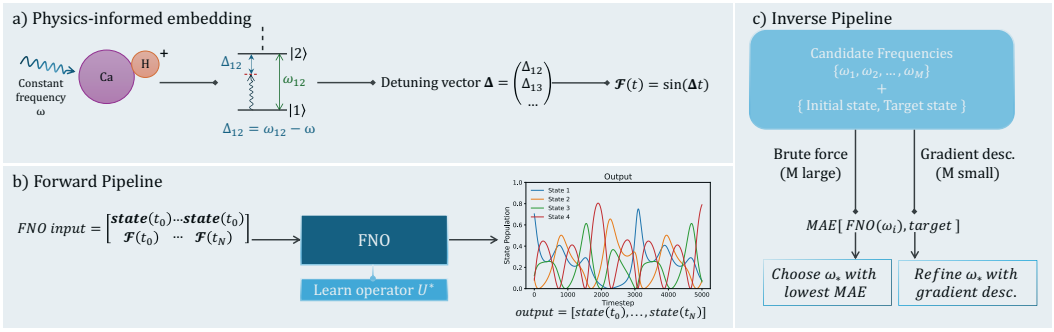

Figure 1: Fourier Neural Operator (FNO) framework for quantum evolution and control. a) Physics-informed embedding: a constant driving frequency ω is compared to molecular transition frequencies ω_{ij} to form detunings $\Delta_{ij} = \omega_{ij} - \omega$, which are modulated as $\mathcal{F}(t) = \sin(\Delta t)$ to encode the oscillatory phase evolution. b) Forward pipeline: given an initial state and the embedded detuning vector $\mathbf{\Delta}$, the FNO learns the operator mapping to the full time evolution of state populations. c) Inverse pipeline: a set of candidate driving frequencies $\{\omega_1, \ldots, \omega_M\}$ together with the initial and target state populations is provided to the trained FNO, which chooses the optimal control parameter ω either by brute-force search (large M) or refinement via gradient descent (small M).

To overcome the high cost and limited scalability of traditional solvers, closed-loop optimization [6] and machine learning approaches [7–9] have emerged, offering adaptive control strategies and fast surrogates. Notably, neural quantum states (NQS) [10] have demonstrated that neural networks can significantly reduce the cost of simulating high-dimensional quantum systems by representing wavefunctions via trainable parameters. Neural operators also provide a well-suited framework to address the computational bottleneck in real-time and large-scale optimization (see Appendix A.1). They generalize neural networks to learn mappings between functions, making them efficient approaches for simulating dynamical systems. For example, Fourier Neural Operators have been used to predict quantum dynamics, even beyond the training window [11]. Related approaches inspired by operator learning [12, 13] instead learn short-time propagators, which are applied iteratively to reconstruct the full dynamics. Overall, the flexibility and differentiability of neural operators allow for very fast simulations and enable inverse design through extensive sampling and gradient-based optimization.

We propose a quantum control framework for steering the population dynamics of complex quantum systems with large Hilbert spaces. We generate data by evolving product states with diagonal density matrices, ensuring initial conditions without coherence terms. We then train a FNO to predict the full time evolution of hyperfine-state populations, given the initial states and a constant driving frequency. While we do not explicitly monitor the coherence terms, their effects are implicitly captured through the evolution of the state populations. The FNO predicts the dynamics with maximum average error across all states under 1.5%, and providing substantial speedups around 2 and 6 orders of magnitude over the numerical solver for 6- and 12-dimensional systems, respectively. While these speedup results should be interpreted with caution as both the solver and FNO could be further optimized, the FNO would still remain multiple orders of magnitude faster. The speedups are also expected to grow further with higher system dimensionality, owing to the superior scalability of the FNO, thereby opening the path to efficient simulations. Leveraging this accuracy and differentiability, we demonstrate successfully two complementary inverse-design strategies to identify driving protocols that prepare desired state populations within a specified time interval: (i) extensive parameter space sampling across dense frequency grids, providing accurate solutions, and (ii) gradient-based optimization via differentiable frequency reparameterization, which converges with far fewer evaluations but requires careful tuning. We apply the framework to the rotational-hyperfine dynamics of CaH^+, a molecular ion relevant to quantum logic spectroscopy and precision measurement experiments, highlighting its potential for scalable optimization and real-time feedback control in regimes where conventional solvers are prohibitively slow.

2 Forward Modeling

We use a Fourier Neural Operator (FNO) [14] (see Appendix A) to simulate the observable dynamics of the quantum states of CaH^+. The dynamics of the system are described by the Hamiltonian $H(t)$, which in the lab frame comprises two parts: a time-independent molecular Hamiltonian H_0 (the hyperfine Hamiltonian) and a time-dependent interaction Hamiltonian that captures the coupling between the laser fields and the internal molecular states. For numerical efficiency, the Hamiltonian is often expressed in the interaction picture. For a detailed description, see Appendix B.

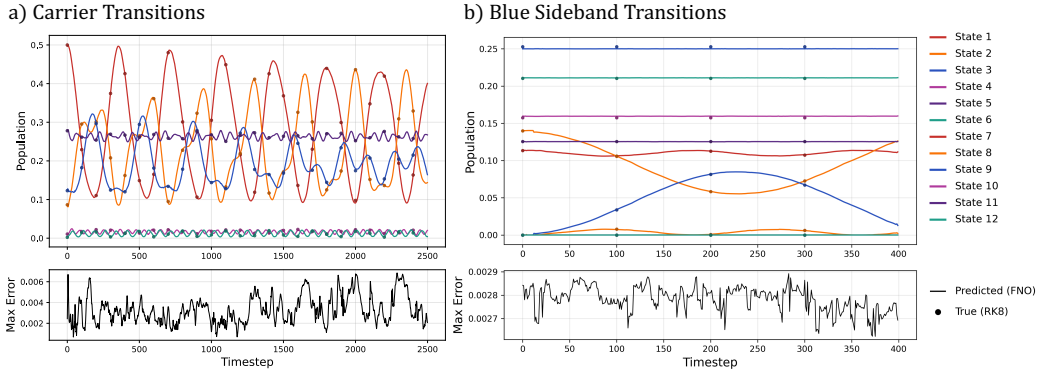

Figure 2: Examples of FNO predictions, compared to reference numerical simulations (RK8) for: a) Carrier transitions: the FNO accurately reproduces the oscillatory dynamics of a randomly initializes 6D system with $\omega = 0.603$ kHz and $4.5\,ms$ duration. b) Blue sideband transitions: the FNO captures the driven population dynamics of a randomly initialized 12D system, with duration = $9.0\,ms$. Bottom panel: maximum absolute error across all states at each timestep. Solid lines denote FNO predictions and dots indicate solver values.

The FNO accurately predicts the evolution of an arbitrary initial state (with no initial coherences) under a constant driving frequency over a predefined time interval, as shown in Fig. 1b). Although the model only tracks the observable dynamics (i.e. diagonal elements of the density matrix) the predicted populations still reflect how coherence influences the system's behavior over time. Efficient learning is enabled by a physics informed embedding (see Figure 1a)) that encodes the laser detuning from the molecule's allowed transitions and measures how far the laser frequency is from driving each one. At each timestep, we further modulate the detuning vector by applying $\sin(\mathbf{\Delta}t)$, where $\mathbf{\Delta}$ is the detuning vector containing the $\Delta_{\mathcal{J},\mathcal{J}'}$ terms defined in Eq. (1), to capture the oscillatory phase evolution appearing in $H_{\mathrm{R}}(t)$. Using only the bare frequency or only the static detuning vector did not yield reliable training performance, likely because these inputs lack the explicit phase information that appears in $H_R(t)$. The use of a sinusoidal feature embedding is particularly synergistic to FNOs, which capture complex interactions between frequency components. It acts as a form of spectral lifting, translating low-dimensional or constant inputs into a richer frequency-domain representation that FNOs can process more effectively and efficiently. The initial state populations are encoded in the channel dimensions of the input, and the output consists of their predicted time evolution.

We train two FNO models, one for a 6-dimensional (6D) system undergoing carrier-type transitions and one for blue sideband transitions (BSB) with a 12-dimensional (12D) subspace, as shown in Figure 2 (see Appendix A.3 for the hyperparameter details). Both models predict dynamics that remain close to the solver, with the average maximum error across all states at each timestep below 0.012, as shown in Figure 4. The FNO generates predictions much faster than the reference solver. For the 6D case, the `qutip.propagator` solver takes on average $\sim$ 0.11s to propagate a single trajectory, while the 12D case takes ~ 185.7 s. With batched inference at the optimal batch sizes, the trained FNO produces results in $\sim 0.32/0.019$ ms per prediction for the 6D/12D cases, corresponding to speedups of about 3.44×10^2 and 9.77×10^6, respectively. For larger Hilbert spaces, such as a 48-state system, direct simulations take closer to the order of hours, and the effective speedup of the FNO is expected to be even greater. While the number of input and output channels increases with the number of states, the FNO lifts them to a fixed high-dimensional representation, meaning only the linear lifting and projection layers grow in size, but these remain small relative to the rest of the model and have a negligible effect on running time. Thus, the gains in efficiency become increasingly apparent with system size. Detailed runtime comparisons with the solver are provided in Appendix A.4. In addition, FNOs can learn not only from numerical simulations but also from experimental data, enabling the study of systems that are intractable for classical simulations.

3 Inverse Design

Quantum state control requires identifying driving protocols that steer the system toward desired target populations within an optimal time. We frame this as an inverse design problem, where the goal is to determine the driving frequency that achieves the target outcome in the shortest possible duration. Using the FNO model, we explore two complementary strategies: i) Extensive parameter space sampling of large parameter spaces becomes feasible thanks to the FNO's speed. ii) Gradient-based optimization, enabled by the full differentiability of the FNO pipeline, providing a direct and efficient route to optimal solutions that can converge with fewer evaluations than exhaustive search. We compare results obtained with the two methods, side-by-side, in Figure 5.

Extensive Parameter Space Sampling. As detailed in Appendix A.4, the trained FNO delivers substantial speedups over the reference numerical solver. This acceleration enables millions of predictions within minutes on a single GPU, allowing experimental parameters to be explored and adjusted within the same day. This process becomes impractical with traditional numerical simulations as system size increases. We use it to probe the trained FNO across a dense grid of control parameters and rank outcomes against target populations, directly identifying which frequencies and initial states best reproduce the target dynamics. Unlike gradient-based optimization methods, which require good initialization, carefully designed loss functions, and may converge to local optima, extensive sampling provides a global, exhaustive view of the control landscape. By tracking error at each timestep, this procedure also reveals the earliest times at which the target can be reached, enabling simultaneous optimization of both state accuracy and evolution duration. In this way, it identifies not only the parameters that yield the best outcome but also the optimal moment to stop the system to minimize control time. Using this approach, we determined the optimal driving frequency and initial state that steer the system toward the desired final state while minimizing evolution time, as illustrated in Figure 5.

Gradient-Based Optimization. The FNO also supports gradient-based optimization due to its inference speed and differentiability, a capability that conventional solvers lack due to their slower forward evaluations and non-differentiability with respect to design parameters. To demonstrate this, we consider the problem of determining an optimal driving frequency that leads to target populations, using the mean-absolute error between the FNO predictions and the desired state populations as the loss function. We begin with a coarse grid search across frequency intervals to identify promising regions, sampling a few frequencies within each physically relevant range that influences the dynamics. Evaluating the loss at these points allows us to select the intervals with the lowest losses. Within each selected interval, we employ a differentiable reparameterization scheme using sigmoids for the driving frequency, ensuring that the optimized frequency remains constrained to physically relevant regions while maintaining smooth gradients. Gradient-based optimization is then applied to converge to an optimal solution, where gradients of the loss function with respect to the design parameters are obtained via backpropagation through the entire pipeline. Although the current implementation optimizes only the driving frequency, we also explore truncating simulated trajectories to reduce pulse duration, effectively treating time as an adjustable cutoff.

4 Discussion

We demonstrated that FNOs can learn the observable dynamics of quantum states using a physics-informed embedding, enabling accurate predictions for experimentally relevant driving pulses: blue sideband and carrier transitions. The FNO takes as input the driving frequency and initial state populations, and reproduces the dynamics of CaH^+ with high fidelity and orders of magnitude faster than the numerical solver. This establishes the FNO as an efficient and accurate dynamics propagator.

Leveraging the substantial speedups offered by the FNO, we first tackled inverse design tasks through extensive sampling, via dense sweeps over frequency and initial-state parameters to directly identify combinations that achieve target outcomes. By tracking fidelity at each timestep, we determined the minimum control duration required to reach the desired states. We then applied gradient-based optimization to the same tasks, taking advantage of the FNO's differentiability and speed. In this one-dimensional control space, dense sampling generally outperformed gradient-based optimization, providing near-global optima while remaining computationally tractable. Gradient-based methods, however, require far fewer evaluations and would likely become more advantageous in higher-dimensional settings where exhaustive search is impractical. Together, these results demonstrate that FNOs not only deliver accurate and efficient forward modeling of quantum dynamics but also enable practical and rapid inverse design, offering a powerful framework for quantum control.

In future work, we plan to extend the proposed framework to handle time-dependent frequency sequences, moving beyond constant-frequency driving. This broadens the accessible control parameter space and enables more flexible and powerful designs. We also plan to include pulse duration and selected initial states as optimization variables, enabling joint optimization of both control fields and system preparation. Looking ahead, we aim to scale these methods to larger Hilbert spaces, targeting at least 50 states in CaH^+ and ultimately hundreds in larger molecular systems. This approach could demonstrate that FNOs can efficiently optimize complex control regimes that would otherwise require days of computation, highlighting their potential for significantly accelerating quantum control tasks.

Acknowledgments

A. Pipi and P. Narang are supported by the Department of Energy (DOE) Office of Science (SC) Grant No DOE DE-FOA-0003432, and also by Grant No GBMF12976 of the Gordon and Betty Moore Foundation. N. Gopinath gratefully acknowledges the support of the SURF Fellowship. A. Anandkumar is supported in part by the Bren endowed chair, ONR (MURI grant N00014-18-12624), and by the AI2050 senior fellow program at Schmidt Sciences.

References

[1] Navin Khaneja, Timo Reiss, Carsten Kehlet, Thomas Schulte-Herbrüggen, and Steffen J Glaser. Optimal control of coupled spin dynamics: design of nmr pulse sequences by gradient ascent algorithms. *Journal of Magnetic Resonance*, 172(2):296–305, 2005.

[2] José P Palao and Ronnie Kosloff. Quantum computing by an optimal control algorithm for unitary transformations. *Physical Review Letters*, 89(18):188301, 2002.

[3] Ferdinand Schmidt-Kaler, Kilian Singer, Ulrich Poschinger, Michael Murphy, Svetoslav S Ivanov, Florian Ziesel, Tommaso Calarco, and Simone Montangero. Optimal control of quantum gates in an exactly solvable spin chain. *New Journal of Physics*, 13(7):075014, 2011.

[4] Daniel J Egger and Frank K Wilhelm. Optimized pulses for the control of uncertain qubits. *Superconductor Science and Technology*, 27(1):014001, 2014.

[5] Tommaso Caneva, Tommaso Calarco, and Simone Montangero. Chopped random-basis quantum optimization. *Physical Review Letters*, 103(24):240501, 2009.

[6] Paul B Wigley, Peter J Everitt, Anton van den Hengel, J W Bastian, M A Sooriyabandara, G D McDonald, K S Hardman, C D Quinlivan, P Manju, C C N Kuhn, Ian R Petersen, A N Luiten, Joseph J Hope, and John D Close. Fast machine-learning online optimization of ultra-cold-atom experiments. *Scientific Reports*, 6(1):25890, 2016.

[7] Murphy Yuezhen Niu, Sergio Boixo, Vadim N Smelyanskiy, and Hartmut Neven. Universal quantum control through deep reinforcement learning. *npj Quantum Information*, 5(1):33, 2019.

[8] Marin Bukov, Alexander GR Day, Dries Sels, Phillip Weinberg, Anatoli Polkovnikov, and Pankaj Mehta. Reinforcement learning in different phases of quantum control. *Physical Review X*, 8(3):031086, 2018.

[9] Jing Zhang, Youle Zhou, Xinyue Wu, Yan Zhou, Xiangyu Hu, Pengfei Li, Yi Wang, Xue Shi, Peng Zhang, Xuebing Wu, et al. Optimal quantum control of rydberg atom arrays via deep reinforcement learning. *Physical Review Research*, 5(2):023053, 2023.

[10] Hannah Lange, Anka Van de Walle, Atiye Abedinnia, and Annabelle Bohrdt. From architectures to applications: a review of neural quantum states. *Quantum Science and Technology*, 9(4): 045001, 2024. doi: 10.1088/2058-9565/ad31d0.

[11] Freya Shah, Taylor L. Patti, Julius Berner, Bahareh Tolooshams, Jean Kossaifi, and Anima Anandkumar. Fourier neural operators for learning dynamics in quantum spin systems. *arXiv preprint arXiv:2310.03077*, 2023. URL `https://arxiv.org/abs/2310.03077`.

[12] Jiaji Zhang, Carlos L. Benavides-Riveros, and Lipeng Chen. Artificial-intelligence-based surrogate solution of dissipative quantum dynamics: physics-informed reconstruction of the universal propagator. *The Journal of Physical Chemistry Letters*, 2024. doi: 10.1021/acs.jpclett.4c00598.

[13] Jiaji Zhang, Carlos L. Benavides-Riveros, and Lipeng Chen. Neural quantum propagators for driven-dissipative quantum dynamics. *Physical Review Research*, 7(L012013), 2025. doi: 10.1103/PhysRevResearch.7.L012013.

[14] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[15] Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, pages 1–9, 2024.

[16] Julius Berner, Miguel Liu-Schiaffini, Jean Kossaifi, Valentin Duruisseaux, Boris Bonev, Kamyar Azizzadenesheli, and Anima Anandkumar. Principled approaches for extending neural architectures to function spaces for operator learning, 2025. URL `https://arxiv.org/abs/2506.10973`.

[17] Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for Fourier neural operators. *J. Mach. Learn. Res.*, 22(1), 2021. ISSN 1532-4435.

[18] Vignesh Gopakumar, Stanislas Pamela, Lorenzo Zanisi, Zongyi Li, Anima Anandkumar, and MAST Team. Fourier neural operator for plasma modelling. *arXiv preprint arXiv:2302.06542*, 2023.

[19] Thorsten Kurth, Shashank Subramanian, Peter Harrington, Jaideep Pathak, Morteza Mardani, David Hall, Andrea Miele, Karthik Kashinath, and Animashree Anandkumar. FourCastNet: Accelerating global high-resolution weather forecasting using adaptive Fourier neural operators, 2022.

[20] Gege Wen, Zongyi Li, Qirui Long, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M. Benson. Real-time high-resolution CO2 geological storage prediction using nested Fourier neural operators. *Energy Environ. Sci.*, 16:1732–1741, 2023. doi: 10.1039/D2EE04204E.

[21] Luca Ghafourpour, Valentin Duruisseaux, Bahareh Tolooshams, Philip H. Wong, Costas A. Anastassiou, and Anima Anandkumar. Noble – neural operator with biologically-informed latent embeddings to capture experimental variability in biological neuron models, 2025. URL `https://arxiv.org/abs/2506.04536`.

[22] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 1(3):1–27, 2024.

[23] Ryan Y. Lin, Julius Berner, Valentin Duruisseaux, David Pitt, Daniel Leibovici, Jean Kossaifi, Kamyar Azizzadenesheli, and Anima Anandkumar. Enabling automatic differentiation with mollified graph neural operators, 2025.

[24] Adarsh Ganeshram, Haydn Maust, Valentin Duruisseaux, Zongyi Li, Yixuan Wang, Daniel Leibovici, Oscar Bruno, Thomas Hou, and Anima Anandkumar. Fc-pino: High precision physics-informed neural operators via fourier continuation, 2025.

[25] Jean Kossaifi, Nikola Kovachki, Zongyi Li, David Pitt, Miguel Liu-Schiaffini, Valentin Duruisseaux, Robert Joseph George, Boris Bonev, Kamyar Azizzadenesheli, Julius Berner, and Anima Anandkumar. A library for learning neural operators, 2025. URL `https://arxiv.org/abs/2412.10354`.

A The Fourier Neural Operator (FNO) for Forward Modeling

A.1 Neural Operators

Machine learning models have been developed to overcome the high computational costs and limited scalability of traditional numerical integrators across scientific fields. Among them, standard neural networks map between finite-dimensional vectors, but PDEs define relationships between infinite-dimensional functions. Neural operators extend the neural network paradigm to learn mappings between these functions [15, 16], making them particularly well-suited for approximating PDE solution operators. They have a universal approximation property for nonlinear operators [17], accept input functions on any discretization, and output functions that can be queried consistently at arbitrary resolutions. This flexibility improves data efficiency by supporting training with data of varying discretization and fidelity. Various neural operators like the Fourier Neural Operator (FNO) [14] have been applied successfully to diverse problems [18–21]. Additionally, physical laws can be incorporated as auxiliary losses terms to enhance or replace supervised data, as demonstrated in physics-informed neural operator frameworks [22–24]. With these capabilities, neural operators enable efficient inverse design, both through extensive parameter space sampling or gradient-based optimization (enabled by the differentiability of neural operators).

Neural operators compose linear integral operators $\mathcal{K}$ with pointwise non-linear activation functions σ to approximate highly non-linear operators. More precisely, we define the neural operator

$$\mathcal{G}_\theta \coloneqq \mathcal{Q} \circ \sigma(W_L + \mathcal{K}_L + b_L) \circ \cdots \circ \sigma(W_1 + \mathcal{K}_1 + b_1) \circ \mathcal{P} \tag{1}$$

where $\mathcal{P}$, $\mathcal{Q}$ are the pointwise neural networks that encode the lower dimension function into higher dimensional space and vice versa. The model stacks L layers of $\sigma(W_l + \mathcal{K}_l + b_l)$ where W_l are pointwise linear operators (matrices), $\mathcal{K}_l$ are integral kernel operators, b_l are bias terms, and σ are fixed activation functions. The parameters θ consists of all the parameters in $\mathcal{P}, \mathcal{Q}, W_l, \mathcal{K}_l, b_l$. Kossaifi et al. [25] maintain a comprehensive open-source library for learning neural operators in PyTorch, which serves as the foundation for our implementation.

A.2 The FNO Architecture

Throughout our numerical experiments, we employ Fourier neural operators to approximate the solution operators of PDEs. A **Fourier neural operator (FNO)** [14] is a neural operator using Fourier integral operator layers, which are defined via

$$\big(\mathcal{K}(\phi)v_t\big)(x) = \mathcal{F}^{-1}\Big(R_\phi \cdot (\mathcal{F}v_t)\Big)(x) \tag{2}$$

where R_ϕ is the Fourier transform of a periodic function κ parameterized by ϕ. On a uniform mesh, the Fourier transform $\mathcal{F}$ can be implemented using the fast Fourier transform (FFT). Note that Fourier neural operators are differentiable. Here is a depiction of the Fourier Neural Operator:

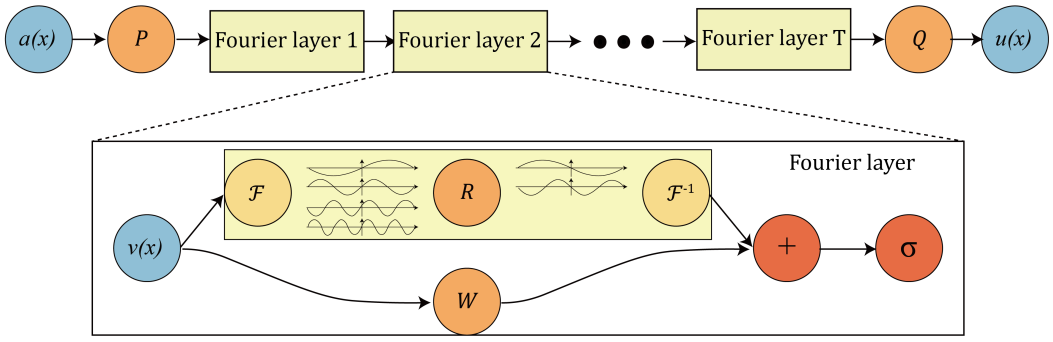

Figure 3: The FNO architecture (extracted from [14]).

A.3 FNO Hyperparameters

For our numerical experiments, we use a 1D Fourier neural operator (FNO) [14].

For carrier transitions, the FNO has 4 layers, with 64 hidden channels and 512 Fourier modes. The resulting architecture has 8.5M trainable parameters and was trained in PyTorch for 150 epochs to minimize the absolute L2 error using the Adam optimizer with learning rate 0.005, and the ReduceLROnPlateau scheduler with factor of 0.6 and patience 4.

For blue sideband transitions, the FNO has 4 layers, with 20 hidden channels and 50 Fourier modes. The resulting architecture was trained in PyTorch for 150 epochs to minimize the absolute L2 error using the Adam optimizer with learning rate 0.008, and the ReduceLROnPlateau scheduler with factor of 0.6 and patience 4.

In both cases, the scalar frequency input is encoded as sinusoidal functions of the detunings. These encodings are then concatenated with the representation of the initial state to form the stacked input to the model. The dimensions of the input are $[D + N, T]$ where D, N, T are the dimensions of the detuning, number of states, and number of timesteps, respectively. The input if then passed through the FNO to generate a prediction for the population dynamics of rotational states in CaH^+.

A.4 Speedups enabled by the FNO

The trained FNO produces predictions orders of magnitude faster than the reference numerical solver for the 6-dimensional (6D) and 12-dimensional (12D) examples considered. Running time measurements were performed on a workstation equipped with a single NVIDIA RTX 4090 GPU (24GB VRAM), an AMD Ryzen 9 7900X CPU, and 64GB of system RAM.

For the reference solver, generating a single trajectory required on average ~ 0.11 s (6D example) and ~ 185.7 s (12D example), measured over 20 different samples. In contrast, running one FNO inference at a time (batch size 1) took only ~ 1.27 ms (6D) and ~ 1.16 ms (12D), averaged over 10,000 samples. This corresponds to a speedup of approximately 8.7×10^1 (6D) and 1.6×10^5 (12D) relative to the duration that the numerical solver takes to propagate all the product states.

In addition, the trained FNO can generate multiple predictions simultaneously on a single GPU by exploiting batched inference. In particular, with a batch size of $9/63$ (which we found to be optimal for speed), the trained FNO takes roughly $0.30/0.019$ ms per prediction for the 6D/12D examples (averaged over $10{,}000$ samples). This corresponds to speedups of 3.44×10^2 (6D) and 9.77×10^6 (12D) compared to the numerical solver.

These computational speedup results should be interpreted with caution, since the reference solver was not optimized for performance and could be further accelerated, in particular by leveraging GPU capabilities. Likewise, additional hyperparameter tuning and model compression could yield smaller, more efficient FNO architectures with similar accuracy. Nevertheless, even accounting for possible improvements of the reference solver, the trained FNO would remain significantly faster by several orders of magnitude.

It is worth noting that the effective speedup of the FNO is expected to become even more pronounced for larger Hilbert spaces. Although the number of input and output channels grows with the number of states, the FNO typically maps these to a fixed high-dimensional latent space. Consequently, only the linear lifting and projection layers expand in size, and their contribution to the total computational cost remains negligible. For instance, a typical latent dimensionality in the FNO is 256 channels, implying that computational cost remains nearly constant until the Hilbert space reaches 256 states or a higher-dimensional latent space is required for sufficient expressivity. Beyond that point, while the cost would increase due to the additional channels, the FNO is still expected to scale more efficiently than the numerical solver for inference, leading to increasingly significant performance gains as system size grows.

B System Hamiltonian

In the lab frame, the time-independent part of the Hamiltonian is given by

$$H_{\mathrm{mol}} + H_{\mathrm{mot}} = \sum_k \hbar\omega_k |k\rangle\langle k| + \hbar\nu \left(a^\dagger a + \frac{1}{2}\right). \tag{3}$$

The system comprises two coupled degrees of freedom: the internal rotational-hyperfine states of the molecule and the motional states, captured by H_{mol} and H_{mot}, respectively. Here, $\hbar\omega_k$ denotes the molecular energy levels, and ν is the motional mode frequency.

The time-dependent part consists of the interaction between the laser field and the internal states of the molecule:

$$H_{\mathrm{int}} = \sum_{\mathcal{J}<\mathcal{J}'} \frac{\Omega_{\mathcal{J},\mathcal{J}'}}{2} \left[e^{i\left[\lambda_{\mathrm{L\text{-}D}}(a_{\mathrm{mot}}+a_{\mathrm{mot}}^\dagger)-\omega t\right]} |\mathcal{J}'\rangle\langle\mathcal{J}| + \mathrm{h.c.}\right]. \tag{4}$$

Here, $\Omega_{\mathcal{J},\mathcal{J}'}$ denotes the effective two-photon Rabi coupling between the initial state $|\mathcal{J}\rangle$ and the target state $|\mathcal{J}'\rangle$. The effective laser frequency of the two-photon transition is $\omega = \omega_1 - \omega_2$ $[\mathrm{rad\,s^{-1}}]$, and the detuning from the molecular transition is $\Delta_{\mathcal{J}'\mathcal{J}} = \omega_{\mathcal{J}'\mathcal{J}} - \omega$. The Lamb-Dicke parameter characterizing the coupling strength between the internal and motional degrees of freedom.

For numerical efficiency, we transform the lab-frame Hamiltonian to the rotating frame with respect to H_0:

$$\begin{aligned} H_{\mathrm{R}}(t) &= e^{iH_0 t} H(t) e^{-iH_0 t} \\ &= \sum_{|\mathcal{J}\rangle\to|\mathcal{J}'\rangle} \frac{\Omega_{\mathcal{J},\mathcal{J}'}}{2} e^{i\Delta_{\mathcal{J}'\mathcal{J}}t} |\mathcal{J}'\rangle\langle\mathcal{J}| \otimes \left[1 + i\lambda_{\mathrm{LD}}\left(a_{\mathrm{mot}} e^{-i\nu t} + a_{\mathrm{mot}}^\dagger e^{i\nu t}\right)\right] + \mathrm{h.c.} \end{aligned} \tag{5}$$

Subsequently, we generate training data numerically (using *QuTiP* or *CUDA-Q*) by evolving the system according to the time-dependent Schrödinger equation:

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = H(t) |\psi(t)\rangle. \tag{6}$$

C Additional Results

C.1 Forward Modeling

While Figure 2 showed examples of FNO predictions with the corresponding errors, we display in Figure 4 the maximum errors obtained over all states, averaged over the entire test dataset.

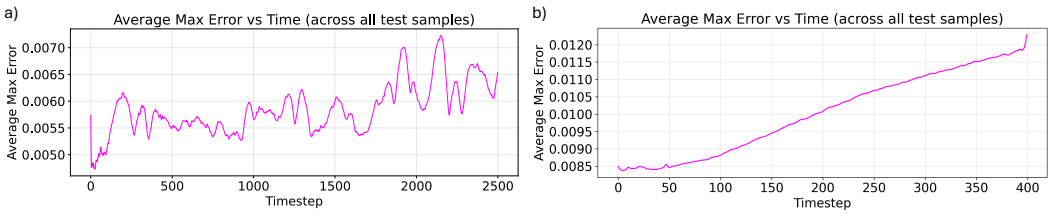

Figure 4: Average maximum error across all states as a function of timestep for a) the 6D system evaluated over 32,000 test samples. and b) the 12D system evaluated over 37,200 test samples.

C.2 Inverse Design

Here, we display the results obtained for an example of inverse design task, both with extensive parameter space sampling and gradient-based optimization.

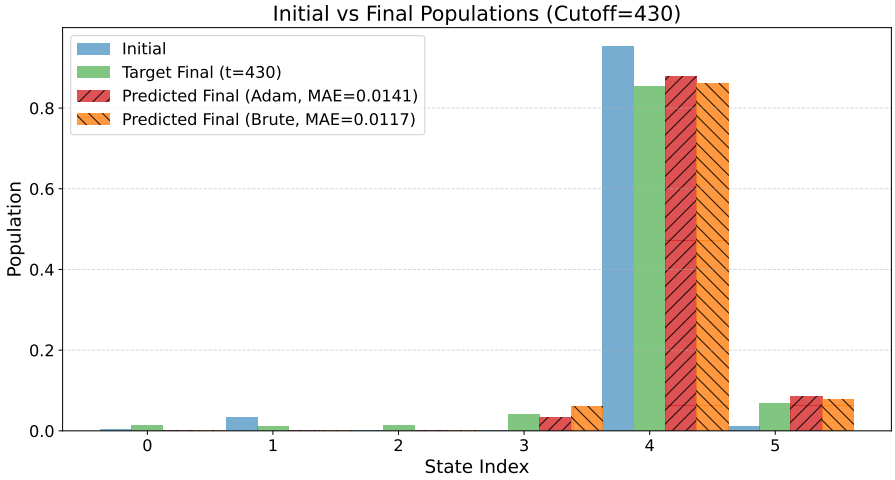

Figure 5: Comparison of initial and final populations across six states at cutoff time $t = 430$. The target final distribution (green) is obtained from numerical dynamics, while the predicted finals are obtained using Adam-based optimization (red, MAE = 0.0141) and extensive sampling search (orange, MAE = 0.0117). Both optimization strategies closely reproduce the target populations starting from the given initial distribution (blue).

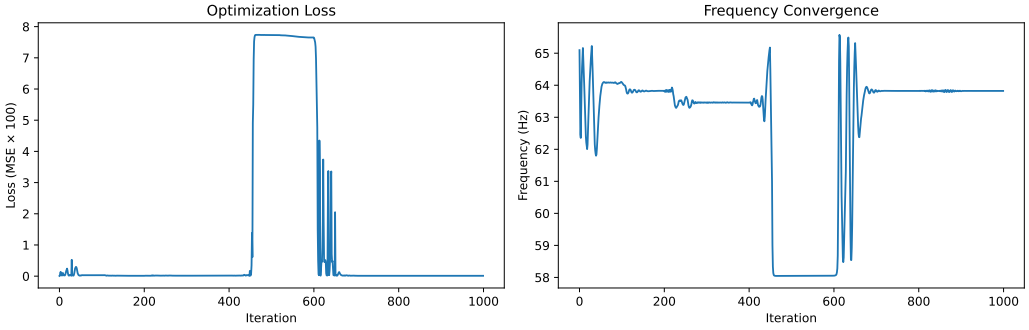

Figure 6: Loss and corresponding frequency trajectory across iterations of the optimization for the example in Figure 5. The left panel shows the loss landscape evolution, while the right panel tracks the frequency updates as the optimizer converges.