Parametric Matrix Models for Scientific Computing

By
Danny Jammooa

STREAMLINE
SmarT Reduction and Emulation Applying Machine Learning In Nuclear Environments

FRIB

MICHIGAN STATE UNIVERSITY

Parametric Matrix Models

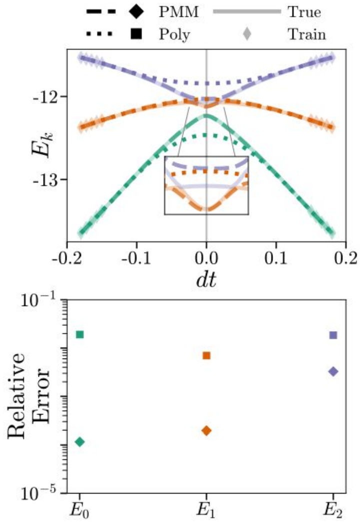
Patrick Cook, Danny Jammooa, Morten Hjorth-Jensen, Daniel D. Lee, Dean Lee

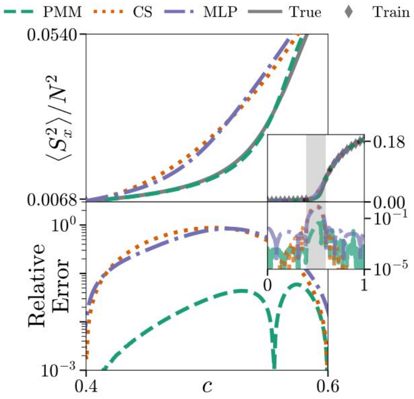
We present a general class of machine learning algorithms called parametric matrix models. Parametric matrix models are based on matrix equations, and the design is motivated by the efficiency of reduced basis methods for approximating solutions of parametric equations. The dependent variables can be defined implicitly or explicitly, and the equations may use algebraic, differential, or integral relations. Parametric matrix models can be trained with empirical data only, and no high-fidelity model calculations are needed. While originally designed for scientific computing, parametric matrix models are universal function approximators that can be applied to general machine learning problems. After introducing the underlying theory, we apply parametric matrix models to a series of different challenges that show their performance for a wide range of problems. For all the challenges tested here, parametric matrix models produce accurate results within a computational framework that allows for parameter extrapolation and interpretability.

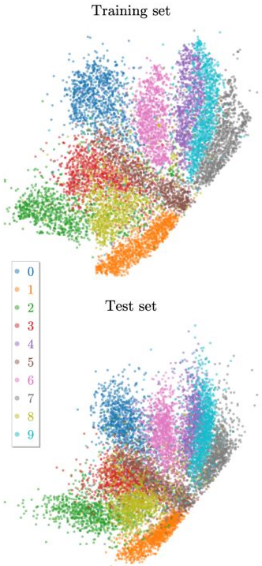
$$M(\vec{x}) = M_0 + \sum_j x_j M_i$$

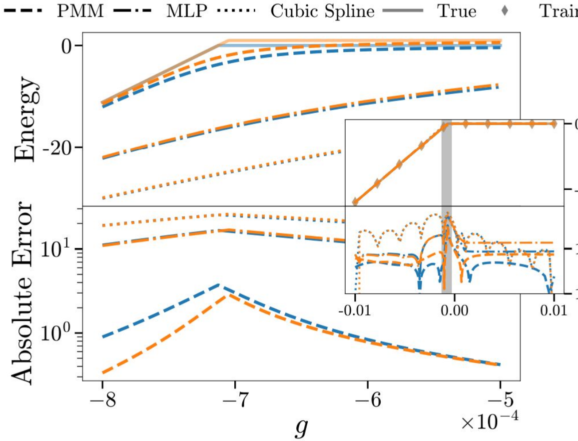
$$\mathcal{L}(\vec{x}, \vec{m}) = \sum_i ||f(\vec{x}_i) - \lambda_k(\vec{x}_i, \vec{m})||^p$$

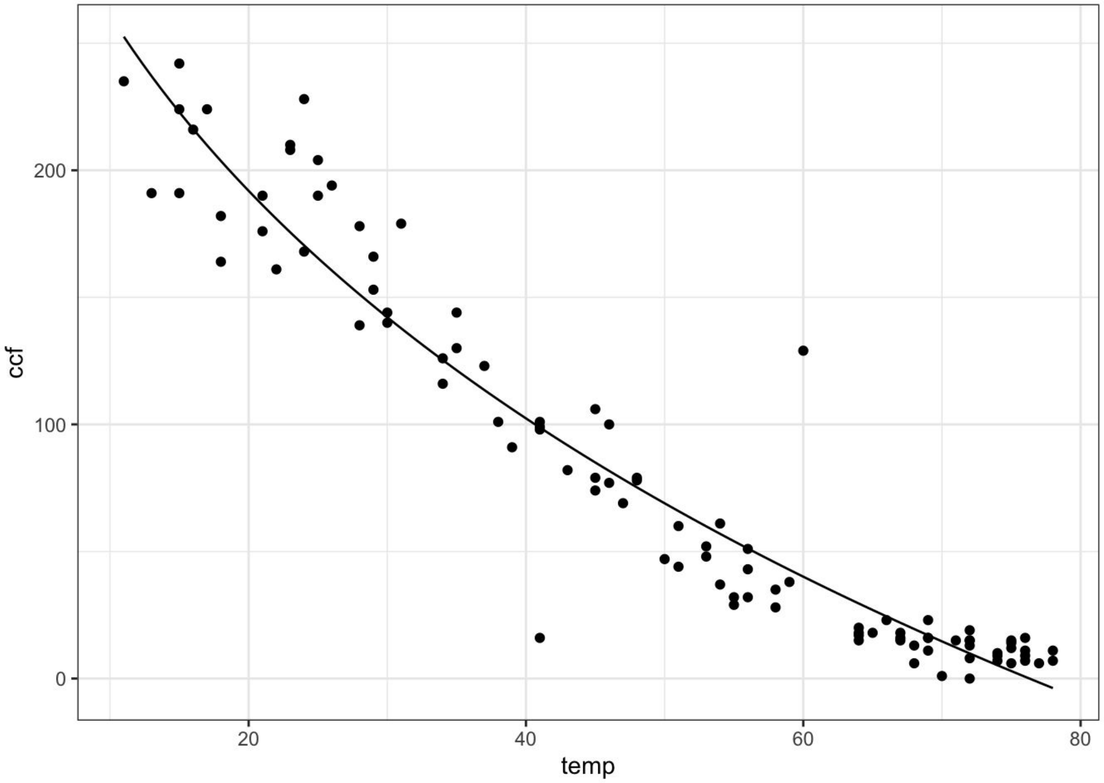
$$H(g) = a^\dagger a + g\left(a^\dagger + a\right)^4$$











1. Linear function:

$$y = mx + b$$

2. Polynomial function:

$$y = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

3. Exponential function:

$$y = ae^{bx}$$

4. Logarithmic function:

$$y = a + b \log(x)$$

5. Power function:

$$y = ax^b$$

6. Sigmoid function:

$$y = \frac{1}{1+e^{-x}}$$

7. Trigonometric function:

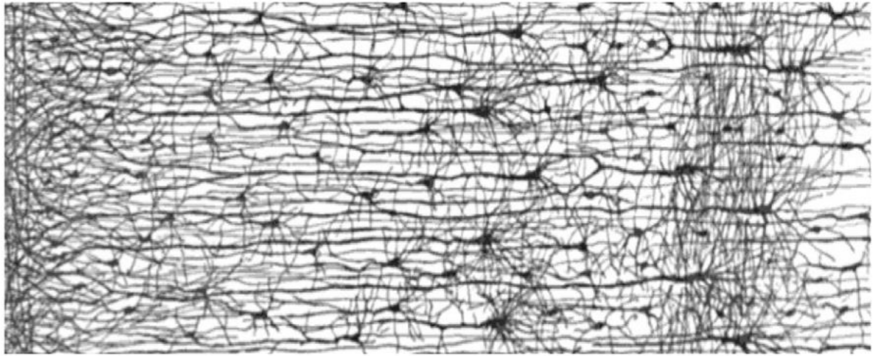
$$y = \sin(x), \cos(x), \tan(x), \ldots$$

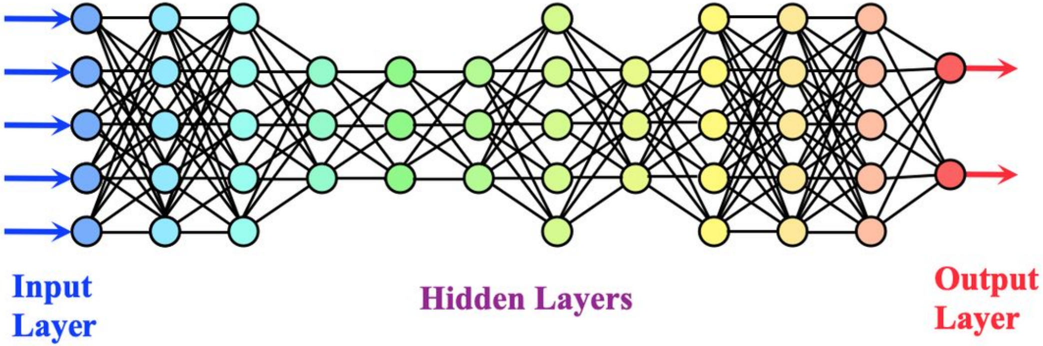
8. Piecewise function:

$$y = \begin{cases} f_1(x), & \text{if } x < a \\ f_2(x), & \text{if } x \geq a \end{cases}$$

9. Gaussian function:

$$y = ae^{-\frac{(x-\mu)^2}{2\sigma^2}}$$





1. Linear function:

$$y = mx + b$$

2. Polynomial function:

$$y = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

3. Exponential function:

$$y = ae^{bx}$$

4. Logarithmic function:

$$y = a + b\log(x)$$

5. Power function:

$$y = ax^b$$

6. Sigmoid function:

$$y = \frac{1}{1+e^{-x}}$$

7. Trigonometric function:

$$y = \sin(x), \cos(x), \tan(x), \ldots$$

8. Piecewise function:

$$y = \begin{cases} f_1(x), & \text{if } x < a \\ f_2(x), & \text{if } x \geq a \end{cases}$$

9. Gaussian function:

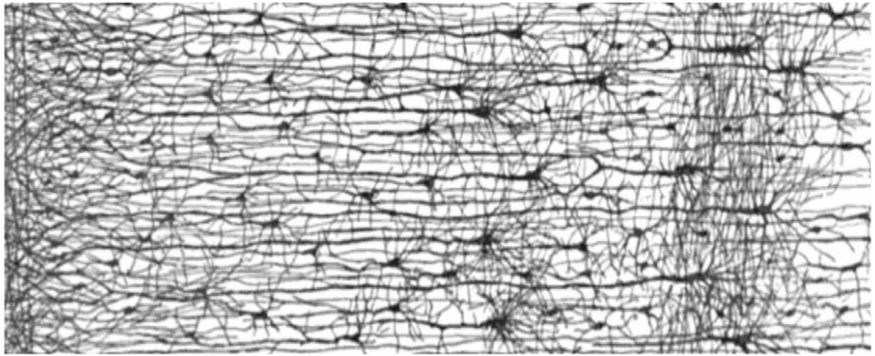
$$y = ae^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

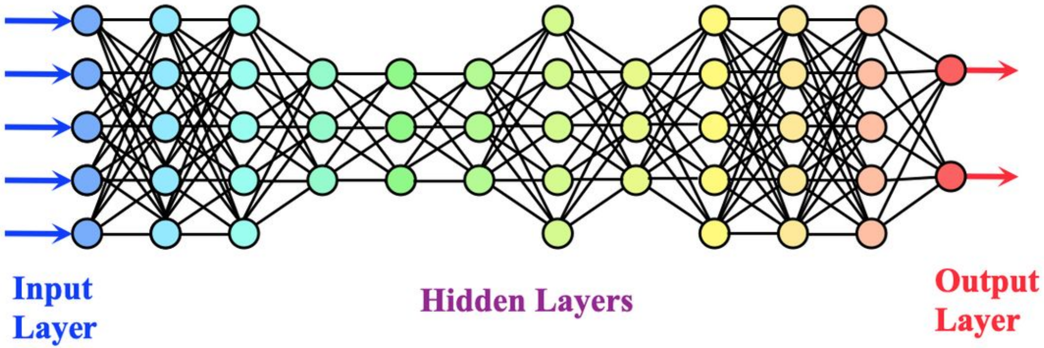
Hamiltonian

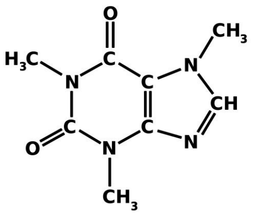
the energy

$$H\Psi = E\Psi$$

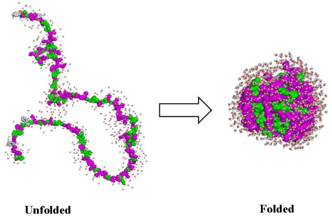
"wave function" that tells location and velocity of the particle



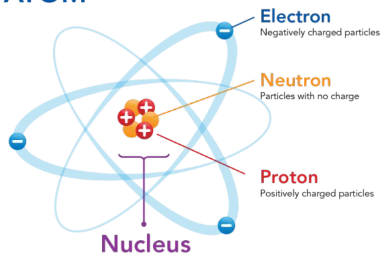




CAFFEINE



ATOM



Machine learning

- Input parameter $\vec{x}$
- Model parameter $\vec{w}$
- Model $h(\vec{x}, \vec{w})$

$$\mathcal{L}(\vec{x}, \vec{w}) = \sum_i ||f(\vec{x}_i) - h(\vec{x}_i, \vec{w})||^p$$

$$\vec{w}^{k+1} = \vec{w}^k - \eta \nabla_{\vec{w}} \mathcal{L}(\vec{x}, \vec{w})$$

Parametric Matrix Models (PMM)

- Parametric matrix models are based on solving matrix equations
- The dependent variables can be defined implicitly or explicitly. Combines elements of RBMs with machine learning
- The simplest PMM we studied is of the form

$$M(\vec{x}) = M_0 + \sum_i x_i M_i$$

$$\mathcal{L}(\vec{x}, \vec{m}) = \sum_i ||f(\vec{x}_i) - \lambda_k(\vec{x}_i, \vec{m})||^p$$

Motivation

Eigenvector Continuation

$$\tilde{N}_{ij} = \langle \psi_i(c) | \psi_j(c) \rangle$$

$$\tilde{H}_{ij} = \langle \psi_i(c) | H(c) | \psi_j(c) \rangle$$

$$\tilde{H}_{ij} | \psi_j(c) \rangle = E \tilde{N}_{ij} | \psi_j(c) \rangle$$

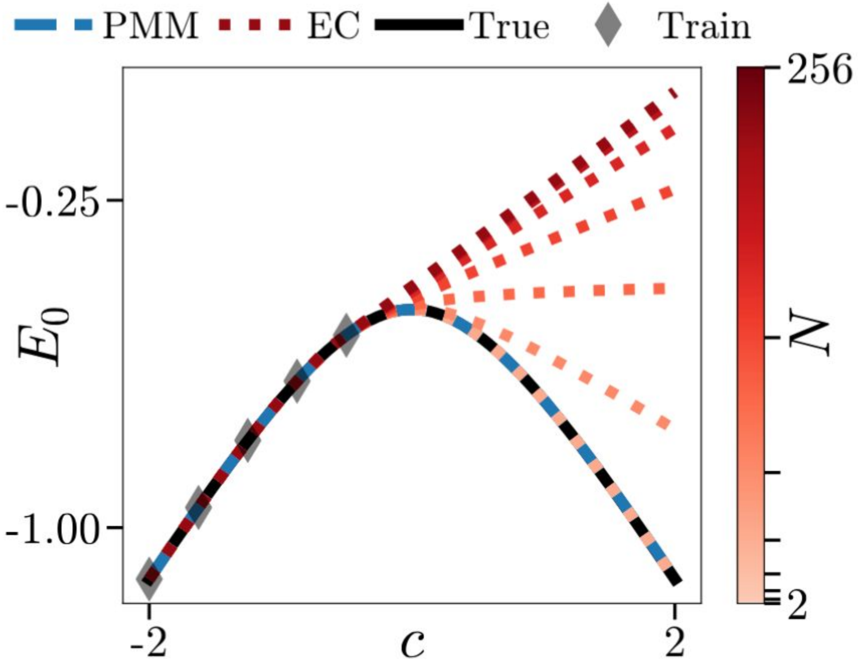
$$\tilde{H}_{ij}(c) = \tilde{H}_0 + c \tilde{H}_I$$

Parametric Matrix Models

$$M(c) = D + cS$$

$$\mathcal{L}(\vec{x}, \vec{m}) = \sum_i ||f(\vec{x}_i) - \lambda_k(\vec{x}_i, \vec{m})||^p$$

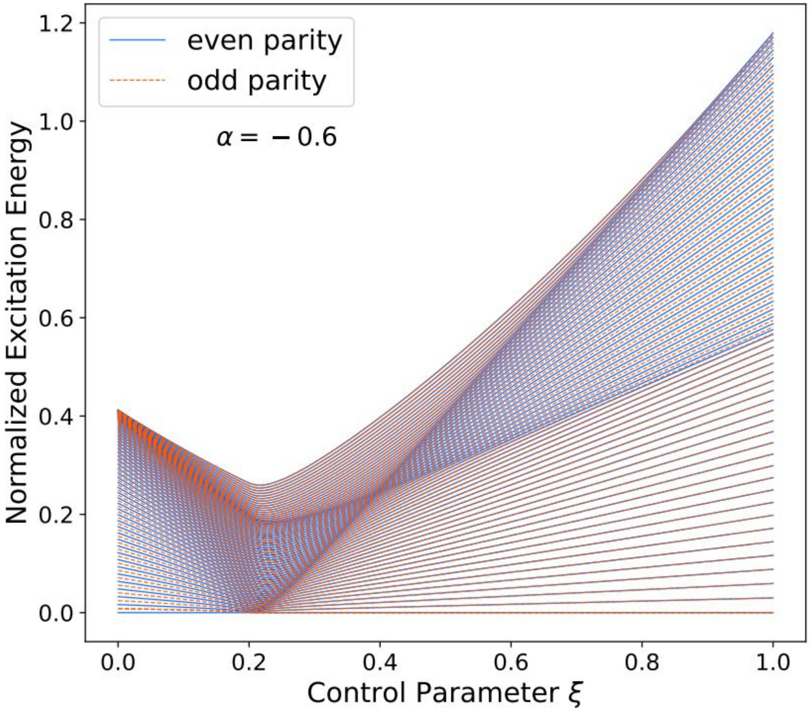
$$H(c) = -\frac{1}{N} \sum_i^N (\sigma_i^z + c\sigma_i^x)$$



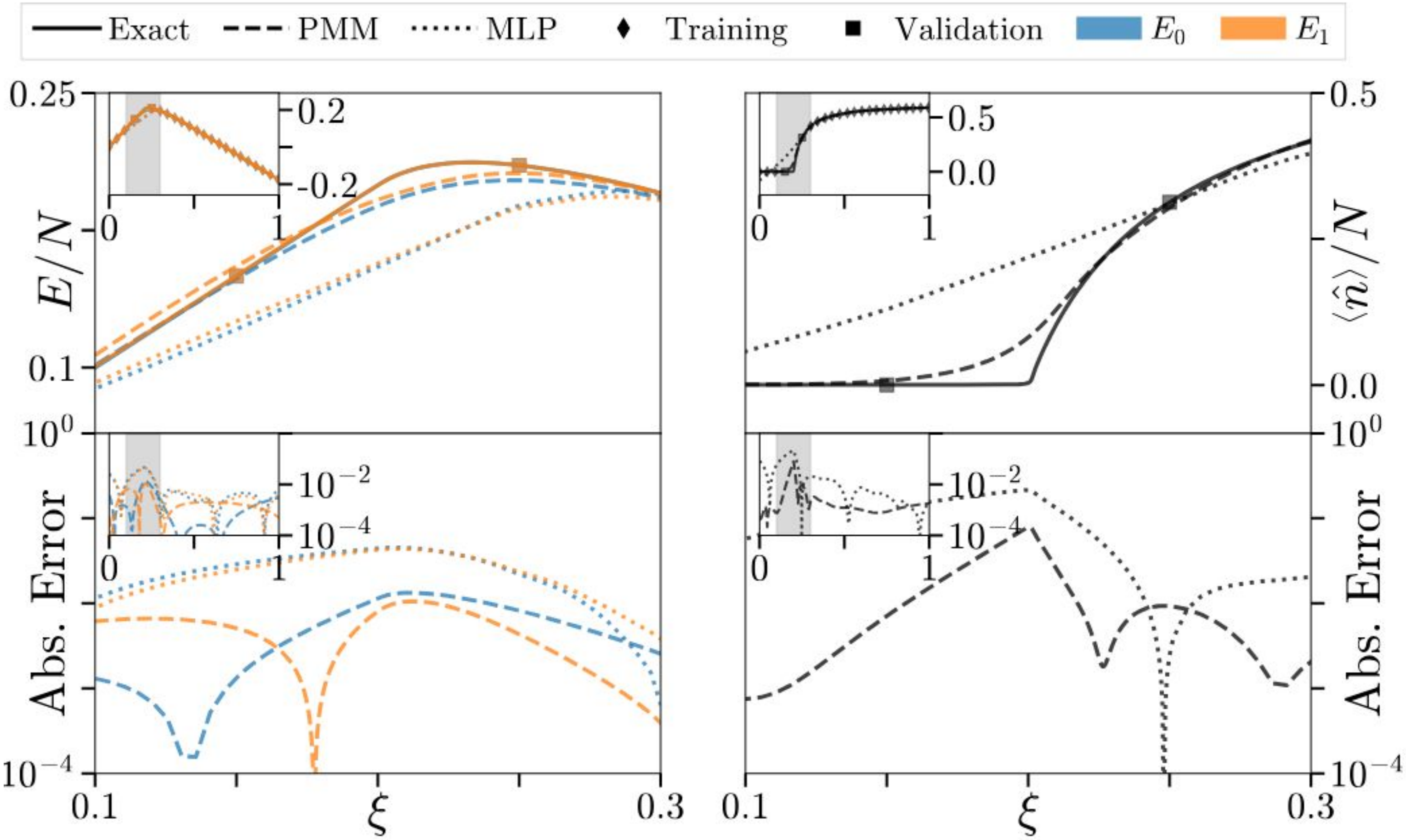
Frame, Dillon, et al. "Eigenvector continuation with subspace learning." *Physical review letters* 121.3 (2018): 032501.

Anharmonic LMG model

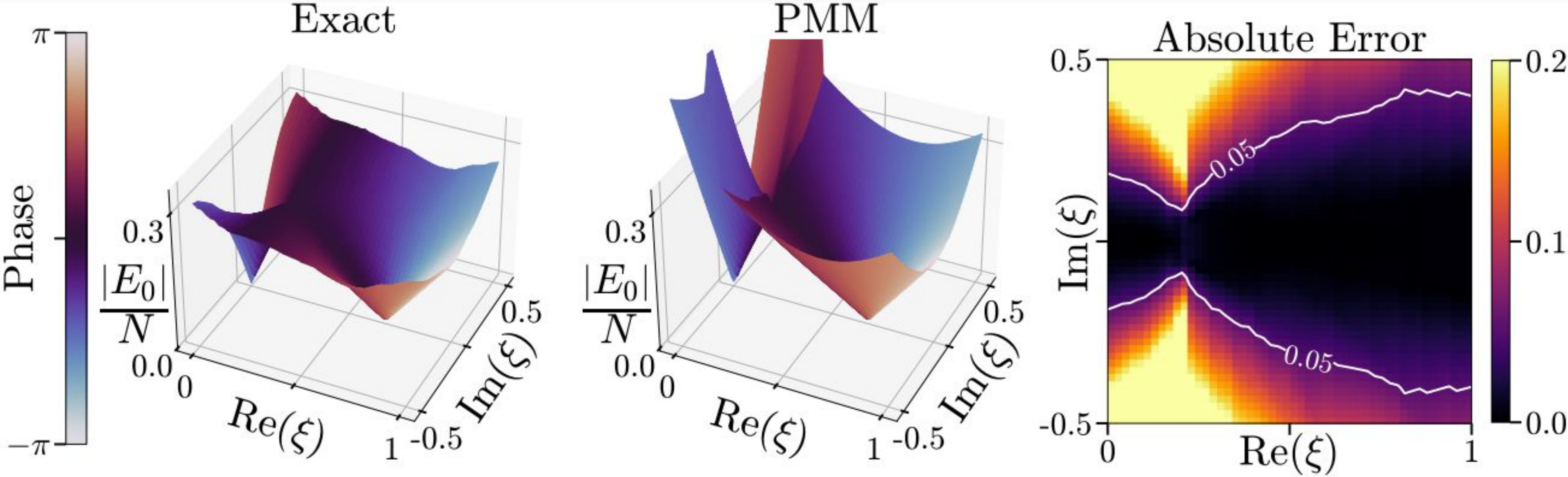
$$\hat{H}_{anh} = (1-\xi)\left(S+\hat{\mathcal{S}}_z\right) + \frac{\alpha}{2S}\left(S+\hat{\mathcal{S}}_z\right)\left(S+\hat{\mathcal{S}}_z+1\right) + \frac{2\xi}{S}\left(S^2-\hat{\mathcal{S}}_x^2\right)$$



Khalouf-Rivera, Jamil, et al. "Excited-state quantum phase transitions in the anharmonic Lipkin-Meshkov-Glick model: Dynamical aspects." *Physical Review E* 107.6 (2023): 064134.

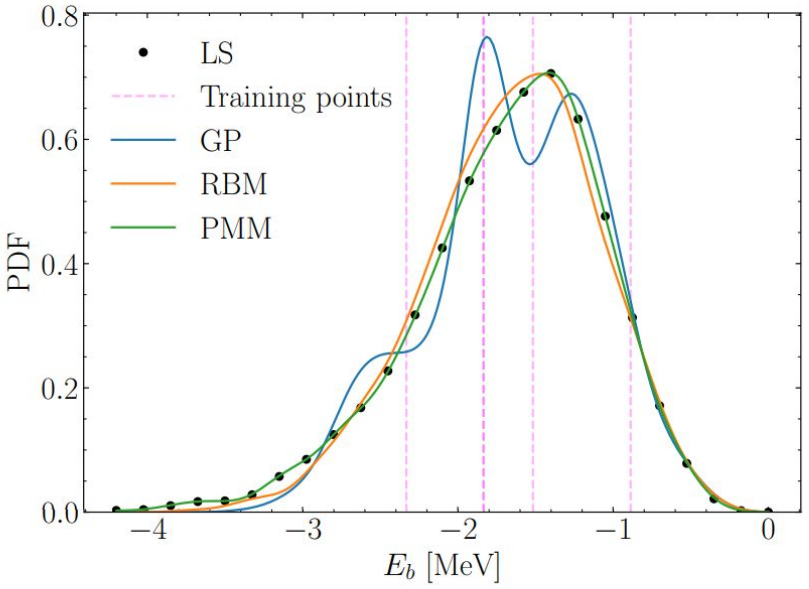


Exact
PMM
MLP
Training
Validation
E_0
E_1
E/N
$\langle \hat{n} \rangle / N$
Abs. Error
Abs. Error
ξ
ξ
0.25
0.1
0.5
0.0
10^0
10^{-4}
10^{-2}
0.2
-0.2
0.1
0.3
0
1



Exact
PMM
Absolute Error
Phase
π
$-\pi$
0.3
0.0
$\frac{|E_0|}{N}$
0
1
Re(ξ)
Im(ξ)
0.5
-0.5
0.05
0.2
0.1
0.0

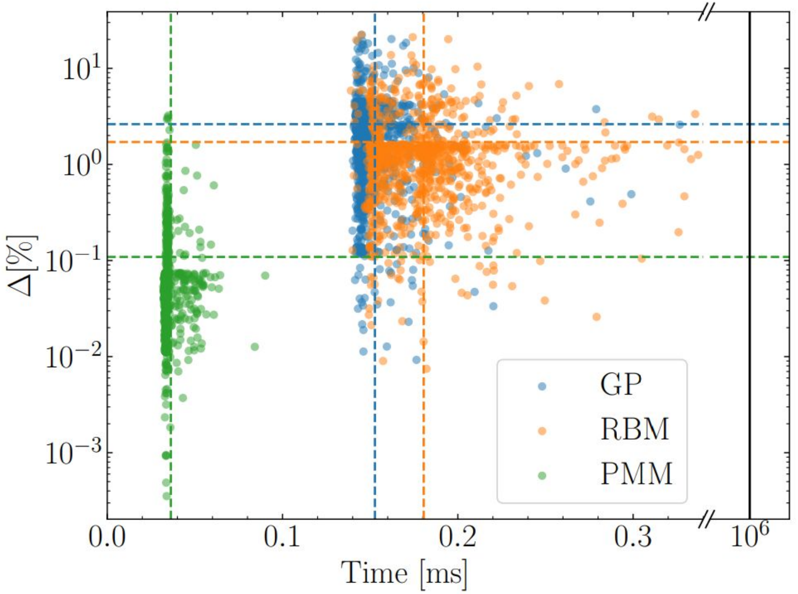
$$A(\vec{c}) = A_0 + \sum_i c_i A_i$$



Emulators for scarce and noisy data: application to auxiliary field diffusion Monte Carlo for the deuteron

Rahul Somasundaram, Cassandra L. Armstrong, Pablo Giuliani, Kyle Godbey, Stefano Gandolfi, Ingo Tews

The validation, verification, and uncertainty quantification of computationally expensive theoretical models of quantum many-body systems require the construction of fast and accurate emulators. In this work, we develop emulators for auxiliary field diffusion Monte Carlo (AFDMC), a powerful many-body method for nuclear systems. We introduce a reduced-basis method (RBM) emulator for AFDMC and study it in the simple case of the deuteron. Furthermore, we compare our RBM emulator with the recently proposed parametric matrix model (PMM) that combines elements of RBMs with machine learning. We contrast these two approaches with a traditional Gaussian Process emulator. All three emulators constructed here are based on a very limited set of 5 training points, as expected for realistic AFDMC calculations, but validated against $\mathcal{O}(10^3)$ exact solutions. We find that the PMM, with emulator errors of only $\approx 0.1\%$ and speed-up factors of $\approx 10^7$, outperforms the other two emulators when applied to AFDMC.



Somasundaram, Rahul, et al. "Emulators for scarce and noisy data: application to auxiliary field diffusion Monte Carlo for the deuteron." *arXiv preprint arXiv:2404.11566* (2024).

Quantum Simulation

- Quantum simulation algorithms are concerned with the solving the Schrodinger equation

$$i\frac{\partial|\psi\rangle}{\partial t} = H|\psi\rangle$$

- Which for a time-independent H, the solution is

$$|\psi\rangle = e^{-iHt}|\psi(0)\rangle$$

- A good first order solution

$$|\psi(t+\Delta t)\rangle \approx (I - iH\Delta t)|\psi(t)\rangle$$

Quantum Simulation

- In most physical systems the Hamiltonian can be written as a sum over k local Hamiltonians.

$$H = \sum_k H_k$$

- H_k terms are often
  - One-body interactions X_i
  - Two-body interactions $X_i\, X_j$
- For example, both the Hubbard and Ising models have Hamiltonians of this form
- Even though e^{iHt} is difficult to compute, it acts on a much smaller subsystem, and is straightforward to approximate using quantum circuits

Trotter product formula

$$[H_i, H_j] \neq 0 \quad e^{-iHt} \neq \Pi_k e^{-iH_k t}$$

- Trotter formula

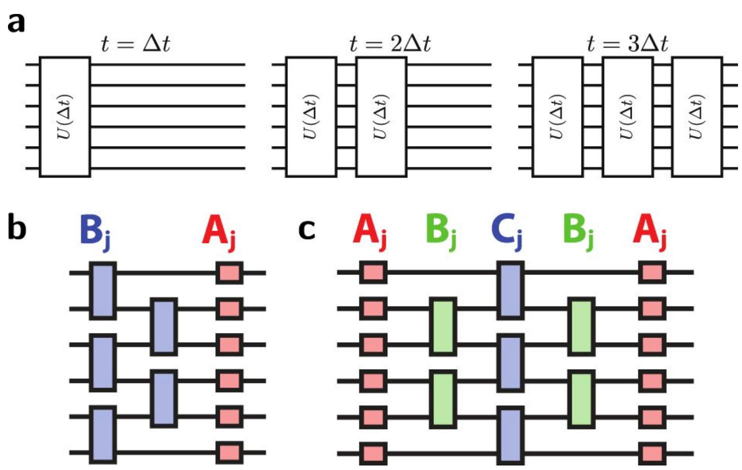
$$\lim_{r\to\infty} (e^{iAt/r} e^{iBt/r})^r = e^{i(A+B)t}$$

- Lie-Trotter

$$e^{i(A+B)\Delta t} = e^{iA\Delta t} e^{iB\Delta t} + \mathcal{O}(\Delta t^2)$$

- Suzuki-Trotter

$$e^{i(A+B)\Delta t} = e^{iA\Delta t/2} e^{iB\Delta t} e^{iA\Delta t/2} + \mathcal{O}(\Delta t^3)$$

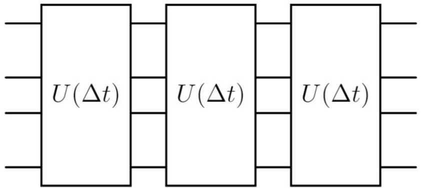


Smith, A., Kim, M.S., Pollmann, F. *et al.* Simulating quantum many-body dynamics on a current digital quantum computer.*npj Quantum Inf* **5**, 106 (2019). https://doi.org/10.1038/s41534-019-0217-0

$$U(t) = e^{-i(A+B)t}, \; t = 1$$

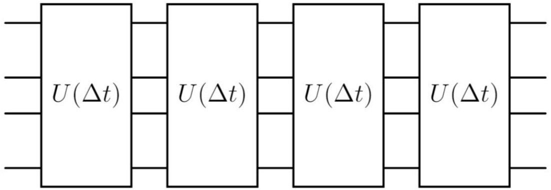
$$e^{-i(A+B)t} = (e^{-iA\Delta t} e^{-iB\Delta t})^r, \; \Delta t = t/r$$

$\Delta t = 0.33$



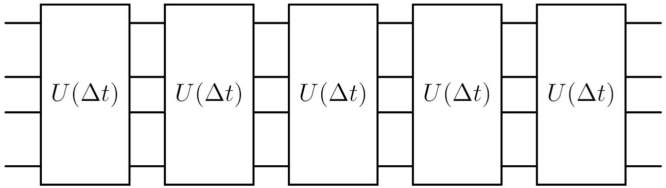
$\rightarrow E(0.33)$

$\Delta t = 0.25$



$\rightarrow E(0.25)$

$\Delta t = 0.2$



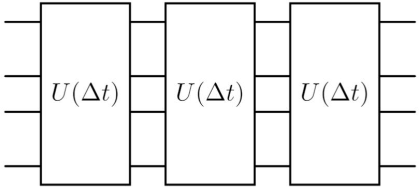
$\rightarrow E(0.2)$

$$U(t) = e^{-i(A+B)t}, \; t = 1$$

$$\epsilon_{total} \propto \epsilon_{theory} + \epsilon_{SPAM} + \epsilon_{gate}^{L}$$

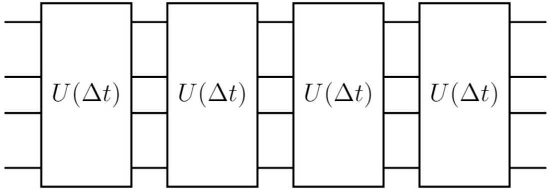
$$e^{-i(A+B)t} = (e^{-iA\Delta t} e^{-iB\Delta t})^r, \; \Delta t = t/r$$

$\Delta t = 0.33$



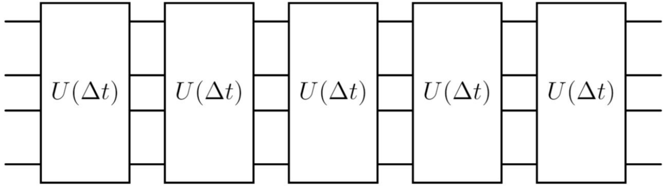
$\rightarrow E(0.33)$

$\Delta t = 0.25$



$\rightarrow E(0.25)$

$\Delta t = 0.2$



$\rightarrow E(0.2)$

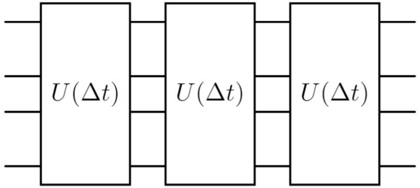
$$U(t) = e^{-i(A+B)t},\ t = 1$$

$$\epsilon_{total} \propto \epsilon_{theory} + \epsilon_{SPAM} + \epsilon_{gate}^{L}$$

$$e^{-i(A+B)t} = (e^{-iA\Delta t}e^{-iB\Delta t})^r,\ \Delta t = t/r$$

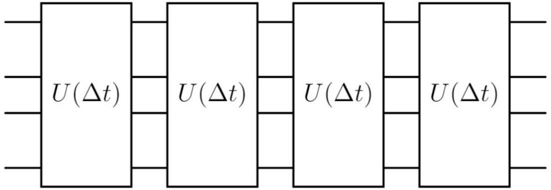
$$e^{i(A+B)\Delta t} = e^{iA\Delta t}e^{iB\Delta t} + \mathcal{O}(\Delta t^2)$$

$\Delta t = 0.33$



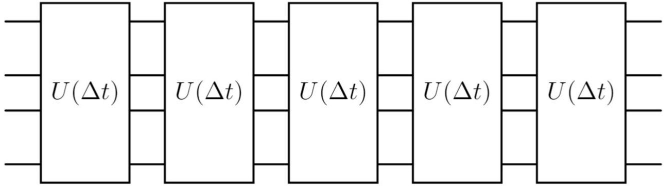
$\rightarrow E(0.33)$

$\Delta t = 0.25$

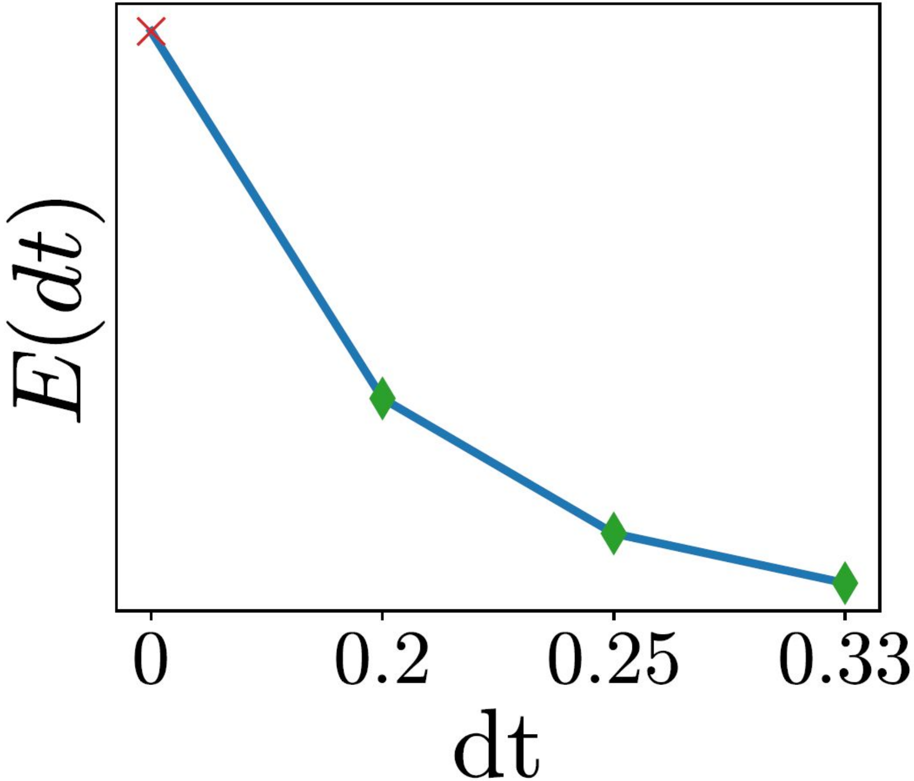


$\rightarrow E(0.25)$

$\Delta t = 0.2$



$\rightarrow E(0.2)$



1D Heisenberg spin chain

$$H = B\sum_{i}^{N} r_i \sigma_i^z + J_1 \sum_{i}^{N} (\sigma_i^x \sigma_{i+1}^x + \sigma_i^y \sigma_{i+1}^y + \sigma_i^z \sigma_{i+1}^z)$$
$$+ J_2 \sum_{i}^{N} (\sigma_i^x \sigma_{i+2}^x + \sigma_i^y \sigma_{i+2}^y + \sigma_i^z \sigma_{i+2}^z)$$

1D Heisenberg spin chain

$$H = B\sum_{i}^{N} r_i\sigma_i^z + J_1\sum_{i}^{N}(\sigma_i^x\sigma_{i+1}^x + \sigma_i^y\sigma_{i+1}^y + \sigma_i^z\sigma_{i+1}^z)$$
$$+ J_2\sum^{N}(\sigma_i^x\sigma_{i+2}^x + \sigma_i^y\sigma_{i+2}^y + \sigma_i^z\sigma_{i+2}^z)$$

$$H_B = B\sum_{i}^{N} r_i\sigma_i^z$$

$$H_{J_1}^{0|1} = J_1 \sum_{i\,\mathrm{even|odd}}^{N} (\sigma_i^x\sigma_{i+1}^x + \sigma_i^y\sigma_{i+1}^y + \sigma_i^z\sigma_{i+1}^z)$$

$$H_{J_2}^{0|1} = J_2 \sum_{i\,\mathrm{even|odd}}^{N} (\sigma_i^x\sigma_{i+2}^x + \sigma_i^y\sigma_{i+2}^y + \sigma_i^z\sigma_{i+2}^z)$$

1D Heisenberg spin chain

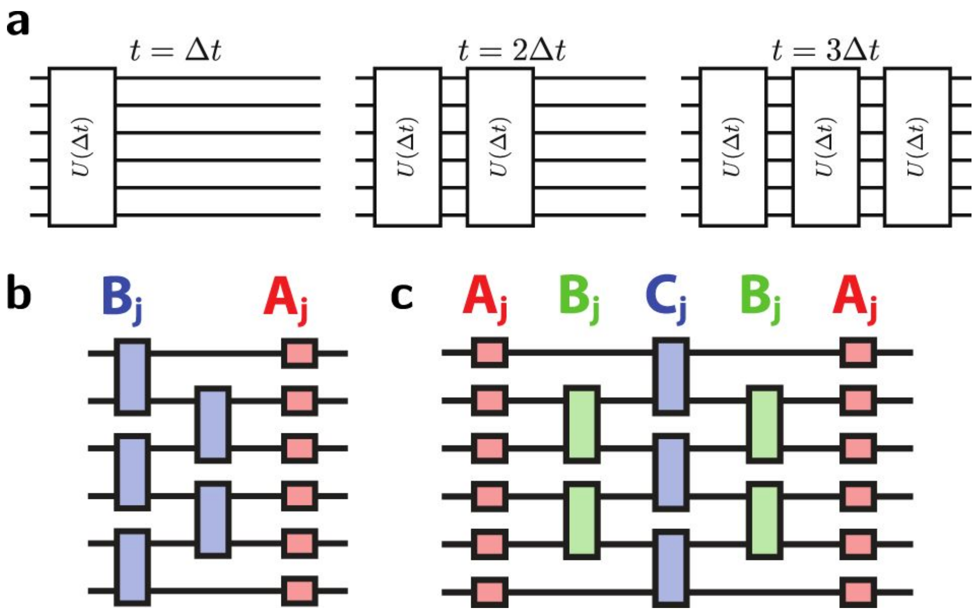
$$H = B\sum_{i}^{N} r_i\sigma_i^z + J_1\sum_{i}^{N}(\sigma_i^x\sigma_{i+1}^x + \sigma_i^y\sigma_{i+1}^y + \sigma_i^z\sigma_{i+1}^z)$$
$$+ J_2\sum^{N}(\sigma_i^x\sigma_{i+2}^x + \sigma_i^y\sigma_{i+2}^y + \sigma_i^z\sigma_{i+2}^z)$$

$$H_B = B\sum_{i}^{N} r_i\sigma_i^z$$

$$H_{J_1}^{0|1} = J_1 \sum_{i\,\text{even}|\text{odd}}^{N} (\sigma_i^x\sigma_{i+1}^x + \sigma_i^y\sigma_{i+1}^y + \sigma_i^z\sigma_{i+1}^z)$$

$$H_{J_2}^{0|1} = J_2 \sum_{i\,\text{even}|\text{odd}}^{N} (\sigma_i^x\sigma_{i+2}^x + \sigma_i^y\sigma_{i+2}^y + \sigma_i^z\sigma_{i+2}^z)$$

$$U(dt) = \exp\{-iH_B dt\}\exp\{-iH_{J_1}^0 dt\} \times \exp\{-iH_{J_1}^1 dt\}\exp\{-iH_{J_2}^0 dt\} \times \exp\{-iH_{J_2}^1 dt\}\,.$$



1D Heisenberg spin chain

$$H = B\sum_i^N r_i\sigma_i^z + J_1\sum_i^N(\sigma_i^x\sigma_{i+1}^x + \sigma_i^y\sigma_{i+1}^y + \sigma_i^z\sigma_{i+1}^z) + J_2\sum_i^N(\sigma_i^x\sigma_{i+2}^x + \sigma_i^y\sigma_{i+2}^y + \sigma_i^z\sigma_{i+2}^z)$$

$$H_B = B\sum_i^N r_i\sigma_i^z$$

$$H_{J_1}^{0|1} = J_1 \sum_{i\,\mathrm{even|odd}}^N (\sigma_i^x\sigma_{i+1}^x + \sigma_i^y\sigma_{i+1}^y + \sigma_i^z\sigma_{i+1}^z)$$

$$H_{J_2}^{0|1} = J_2 \sum_{i\,\mathrm{even|odd}}^N (\sigma_i^x\sigma_{i+2}^x + \sigma_i^y\sigma_{i+2}^y + \sigma_i^z\sigma_{i+2}^z)$$

$$U(dt) = \exp\left\{-iH_B dt\right\}\exp\left\{-iH_{J_1}^0 dt\right\} \times \exp\left\{-iH_{J_1}^1 dt\right\}\exp\left\{-iH_{J_2}^0 dt\right\} \times \exp\left\{-iH_{J_2}^1 dt\right\}.$$

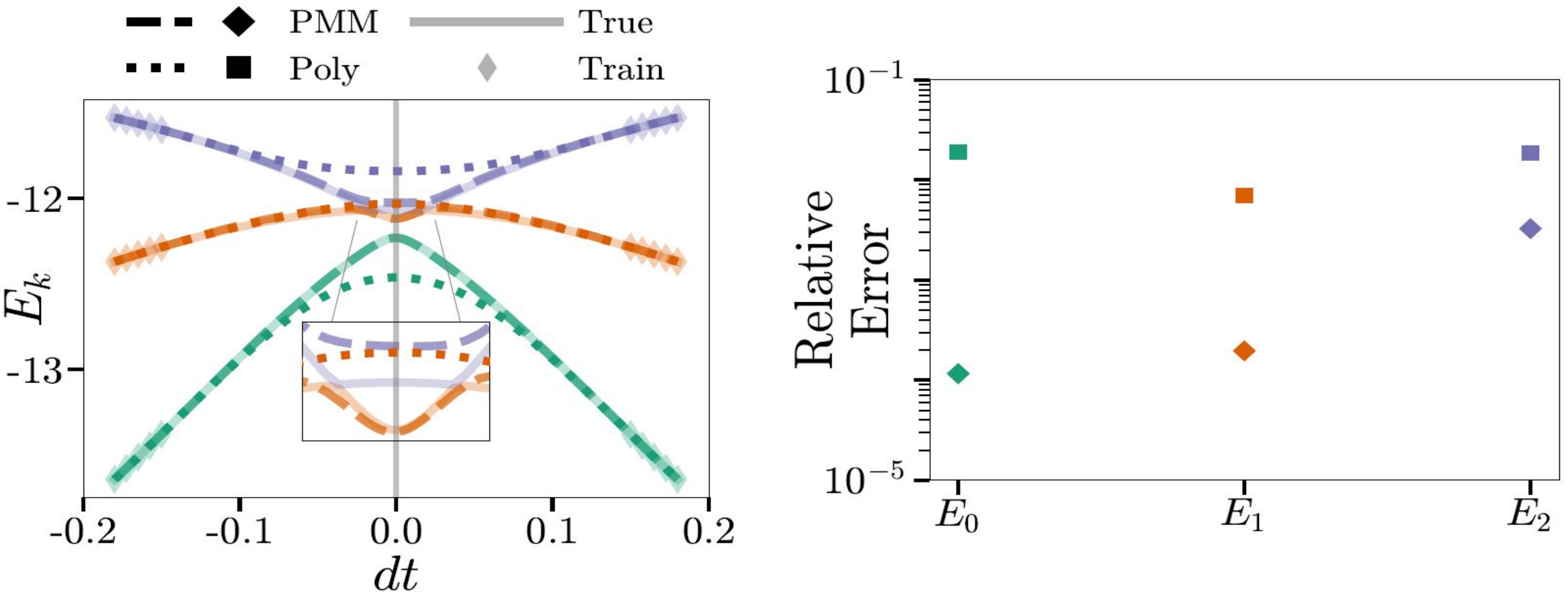
$$\boxed{2^N \times 2^N} \longrightarrow \boxed{m \times m}$$

$$M = M_1 + M_2 + M_3 + M_4 + M_5$$

$$U_M(dt) = \exp\left\{-iM_1 dt\right\}\exp\left\{-iM_2 dt\right\} \times \exp\left\{-iM_3 dt\right\}\exp\left\{-iM_4 dt\right\} \times \exp\left\{-iM_5 dt\right\},$$

$$e^{-iE_k dt} = U_M(dt)$$

$$E_k = \frac{ln[\lambda(U_M(dt)]}{-idt}$$

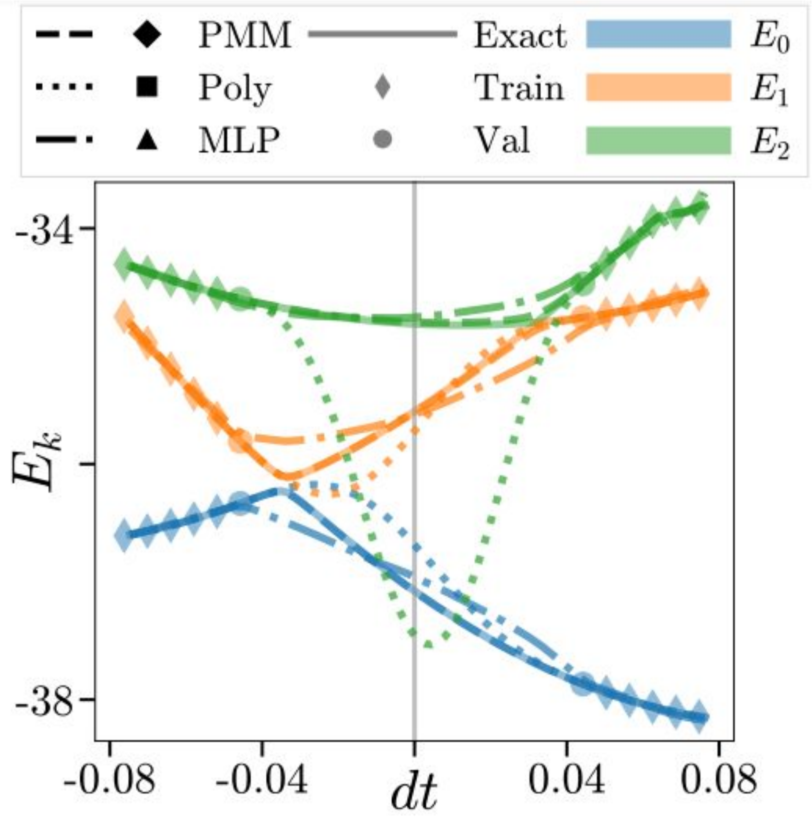

PMM
True
Poly
Train
-12
-13
E_k
-0.2
-0.1
0.0
0.1
0.2
dt
10^{-1}
10^{-5}
Relative Error
E_0
E_1
E_2

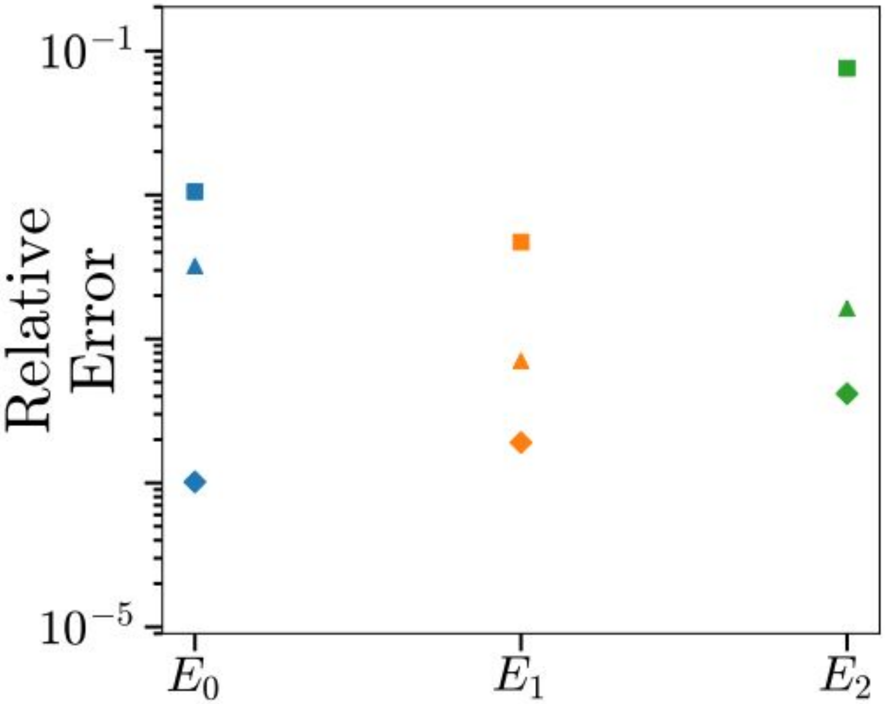
Dzyaloshinskii-Moriya (DM) interaction

$$H_B = B \sum_i r_i \sigma_i^z,$$

$$H_J^{0|1} = J \sum_{i\,\mathrm{even|odd}} (\sigma_i^x \sigma_{i+1}^x + \sigma_i^y \sigma_{i+1}^y + \sigma_i^z \sigma_{i+1}^z),$$

$$H_D^{0|1} = D \sum_{i\,\mathrm{even|odd}} (\sigma_i^x \sigma_{i+1}^y - \sigma_i^y \sigma_{i+1}^x),$$





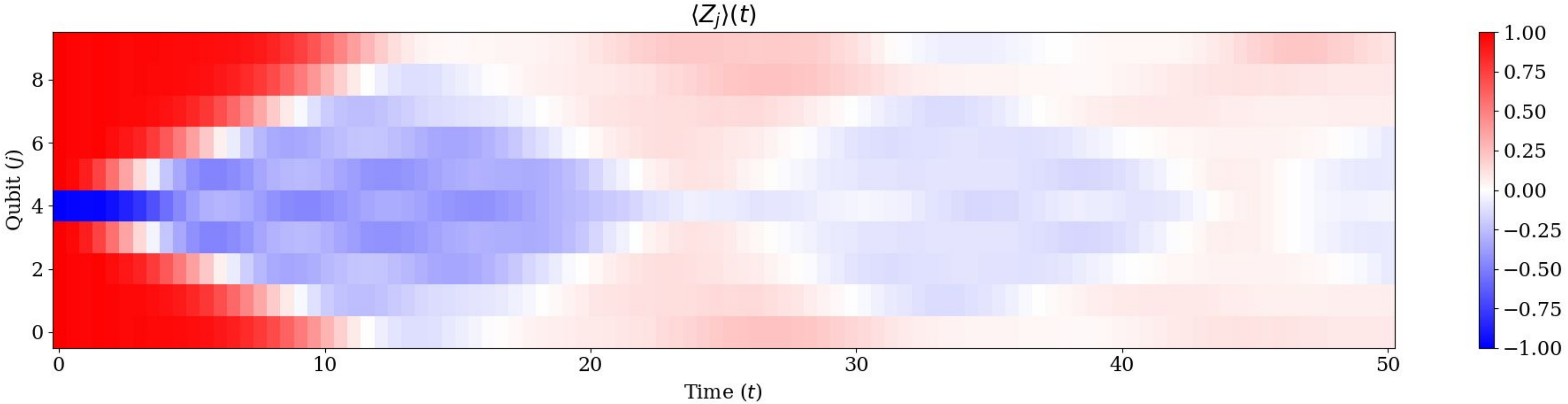
Current and Future Work

Collaboration

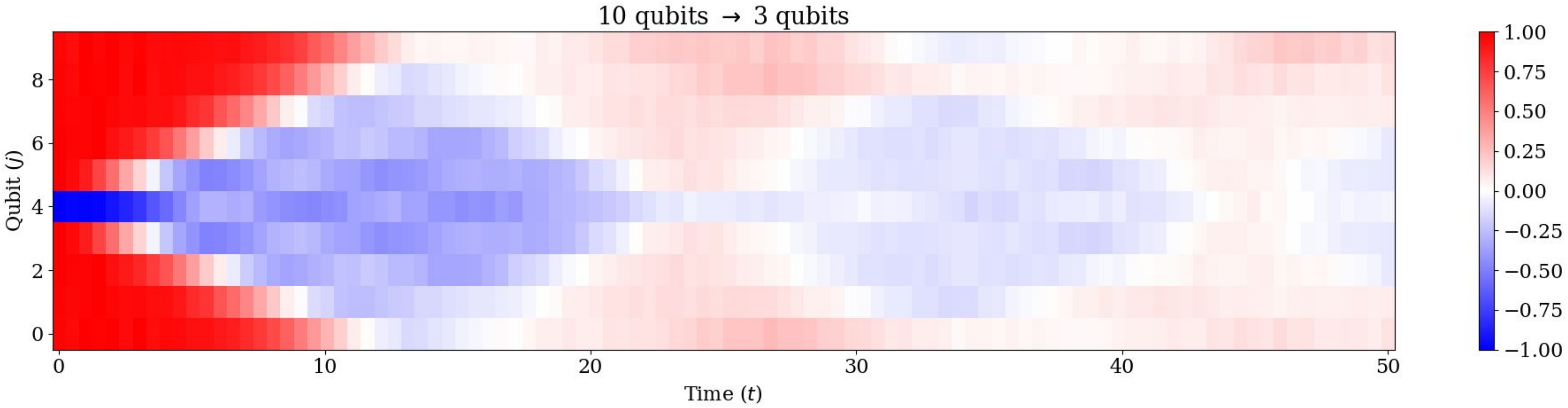
- IMSRG - Heiko Hergert/Scott Bogner
- CC - Morten Hjorth-Jensen
- Fitting Coupling Constant - Daniel Lee
- Fission - Daniel Lay
- Circuit Compression - Ryan Larose
- Scattering - Manuel Catacorarios

Future Work

- Time Evolution
- Time Dependence
- Open Quantum systems
- Learning the Algebra
- ………



$\langle Z_j \rangle(t)$
Qubit (j)
Time (t)
0
2
4
6
8
10
20
30
40
50
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00



10 qubits → 3 qubits
Qubit (j)
Time (t)
0
2
4
6
8
10
20
30
40
50
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Lipkin-Meshkov-Glick (LMG) Model

$$H(c) = -S_z - \frac{2c}{N}(S_x^2 - \frac{1}{2}S_y^2)$$

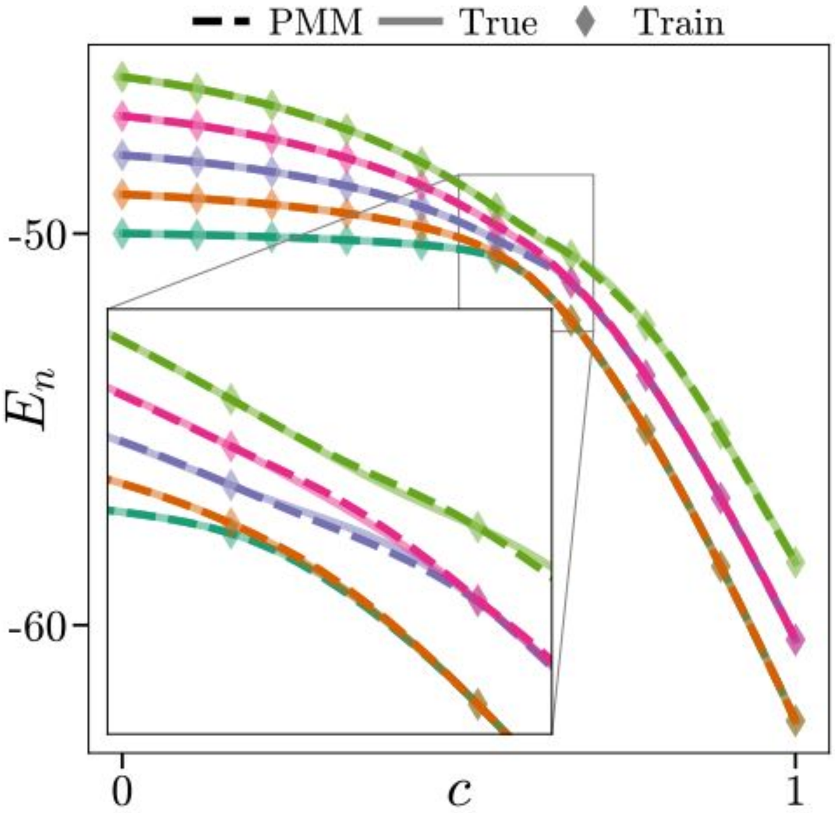
$$M(c) = M_0 + cM_1$$

$$\langle Sx^2(c)\rangle = \langle\psi_{gs}(c)|Sx^2|\psi_{gs}(c)\rangle$$

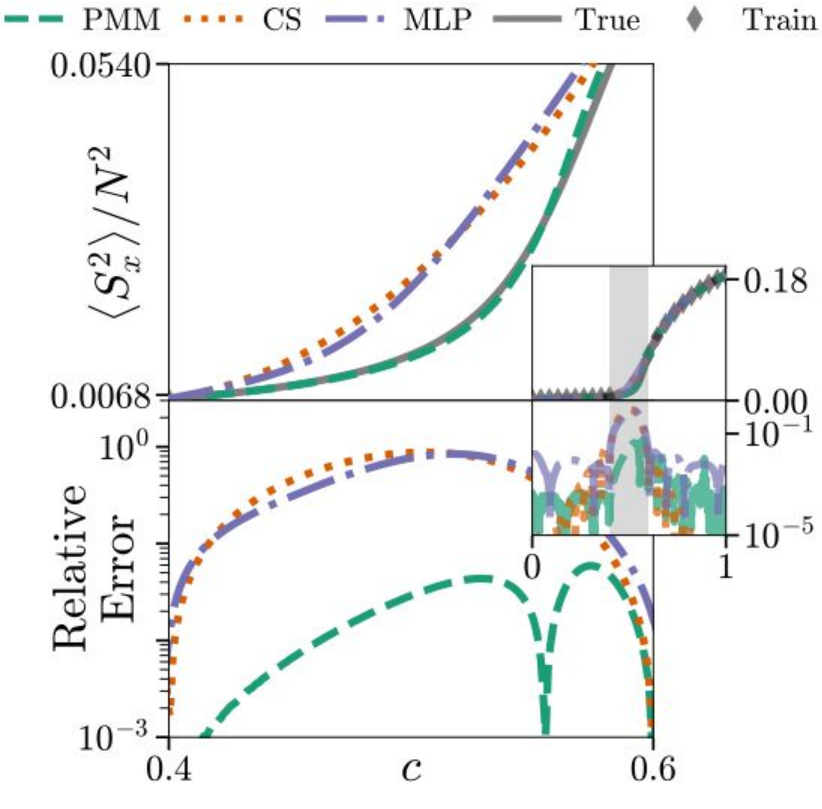
$$\langle O_1(c)\rangle = \langle\phi_{gs(c)}|O_1|\phi_{gs}(c)\rangle$$

$$\langle Sz(c)\rangle = \langle\psi_{gs}(c)|Sz|\psi_{gs}(c)\rangle$$

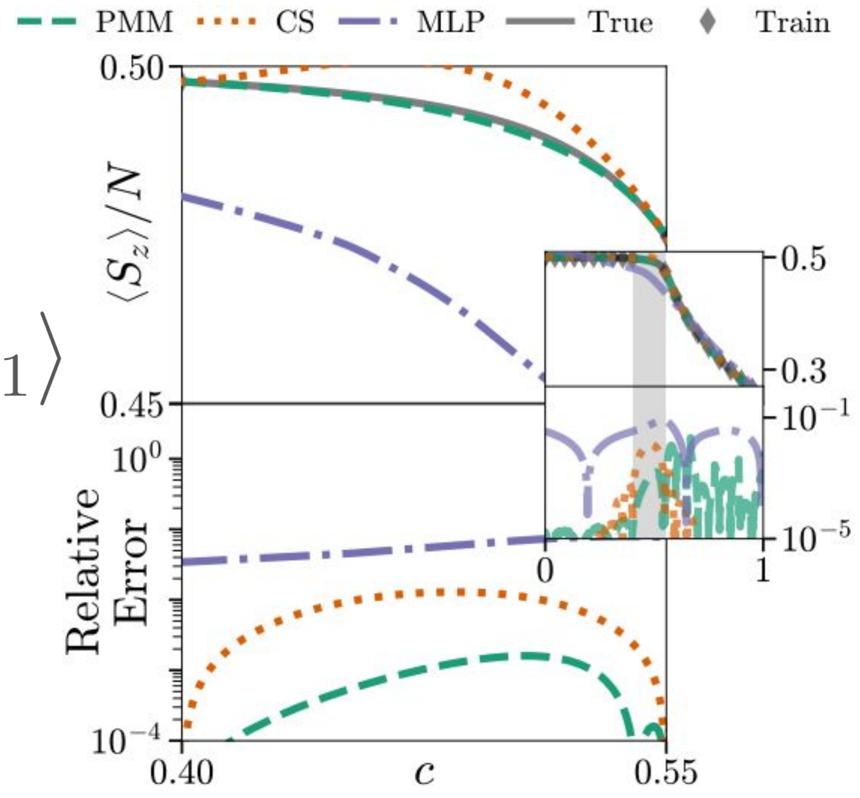
$$\langle O_2(c)\rangle = \langle\phi_{gs(c)}|O_2|\phi_{gs}(c)\rangle$$



LMG Observable



$$\langle O_0 O_1 \rangle$$



References

- Frame, Dillon, et al. "Eigenvector continuation with subspace learning." *Physical review letters* 121.3 (2018): 032501.
- Khalouf-Rivera, Jamil, et al. "Excited-state quantum phase transitions in the anharmonic Lipkin-Meshkov-Glick model: Dynamical aspects." *Physical Review E* 107.6 (2023): 064134.
- Somasundaram, Rahul, et al. "Emulators for scarce and noisy data: application to auxiliary field diffusion Monte Carlo for the deuteron." *arXiv preprint arXiv:2404.11566* (2024).
- Smith, A., Kim, M.S., Pollmann, F. *et al.* Simulating quantum many-body dynamics on a current digital quantum computer.*npj Quantum Inf* **5**, 106 (2019). https://doi.org/10.1038/s41534-019-0217-0

Thank you!
Any Questions?