Parametric Matrix Models

Model Emulation, Model Discovery, and General Machine Learning

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

Facility for Rare Isotope Beams
Michigan State University

Theory Seminar
Institute for Nuclear Physics
TU Darmstadt

Parametric Matrix Models

Model Emulation, Model Discovery, ~~and General Machine Learning~~

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

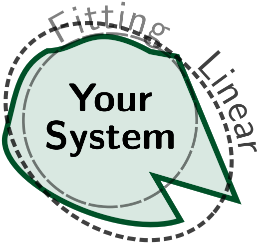
Facility for Rare Isotope Beams
Michigan State University

Theory Seminar
Institute for Nuclear Physics
TU Darmstadt

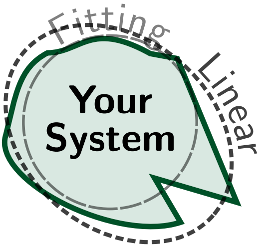
- Your system explores a complicated phase space

- Your system explores a complicated phase space
- Simple fits are fast but rough

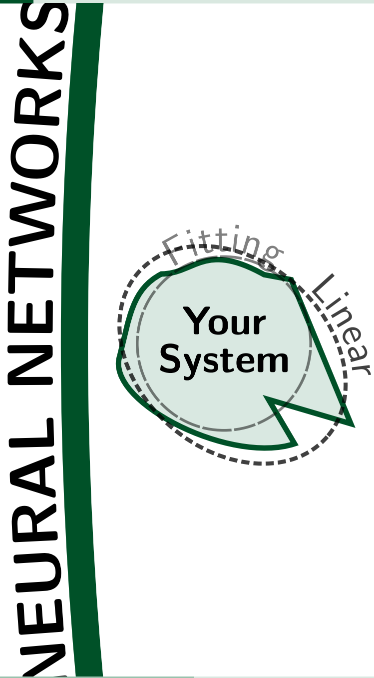
- Your system explores a complicated phase space
- Simple fits are fast but rough
- Linear models are better, but not great, and also slower



- Your system explores a complicated phase space
- Simple fits are fast but rough
- Linear models are better, but not great, and also slower
- Neural networks are powerful...

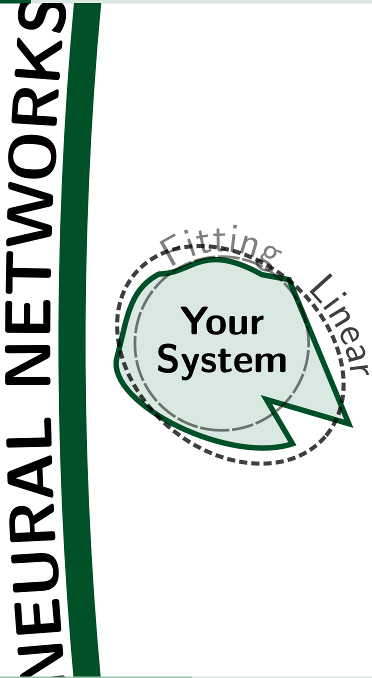


- Your system explores a complicated phase space
- Simple fits are fast but rough
- Linear models are better, but not great, and also slower
- Neural networks are powerful... too powerful...



- Your system explores a complicated phase space
- Simple fits are fast but rough
- Linear models are better, but not great, and also slower
- Neural networks are powerful... too powerful...

- Your system explores a complicated phase space
- Simple fits are fast but rough
- Linear models are better, but not great, and also slower
- Neural networks are powerful... too powerful...



- Your system explores a complicated phase space
- Simple fits are fast but rough
- Linear models are better, but not great, and also slower
- Neural networks are powerful... too powerful...
- **Parametric Matrix Models** *are* your system!

Build-a-PMM$^{\text{TM}}$

Build-a-PMM$^{\text{TM}}$

Your Underlying Equations

$$\begin{cases} H(c) = H_0 + cH_1 \\ H(c)\,|\psi(c)\rangle = E(c)\,|\psi(c)\rangle \\ \langle\psi(c)|O|\psi(c)\rangle = f(c) \end{cases}$$

Build-a-PMM™

Your Underlying Equations

$$\begin{cases} H(c) = H_0 + cH_1 \\ H(c)\,|\psi(c)\rangle = E(c)\,|\psi(c)\rangle \\ \langle\psi(c)|O|\psi(c)\rangle = f(c) \end{cases}$$

$$\begin{cases} \dfrac{\partial\vec{v}}{\partial t} = A(\vec{c})\vec{v} + f(t, c, \vec{v}) \\ \vec{v}(t=0) = \vec{v}_0 \\ v_i(t) = \text{b.c.} \end{cases}$$

Build-a-PMM$^{\text{TM}}$

Your Underlying Equations

$$\begin{cases} H(c) = H_0 + cH_1 \\ H(c)\,|\psi(c)\rangle = E(c)\,|\psi(c)\rangle \\ \langle\psi(c)|O|\psi(c)\rangle = f(c) \end{cases}$$

$$\begin{cases} \dfrac{\partial\vec{v}}{\partial t} = A(\vec{c})\vec{v} + f(t, c, \vec{v}) \\ \vec{v}(t=0) = \vec{v}_0 \\ v_i(t) = \text{b.c.} \end{cases}$$

Your PMM

$$\begin{cases} M(c) = \underline{M_0} + c\underline{M_1} \\ M(c)\,|\tilde{\psi}(c)\rangle = \tilde{E}(c)\,|\tilde{\psi}(c)\rangle \\ \langle\tilde{\psi}(c)|\underline{X}|\tilde{\psi}(c)\rangle = \tilde{f}(c) \end{cases}$$

Build-a-PMM™

Your Underlying Equations

$$\begin{cases} H(c) = H_0 + cH_1 \\ H(c)\left|\psi(c)\right\rangle = E(c)\left|\psi(c)\right\rangle \\ \langle\psi(c)|O|\psi(c)\rangle = f(c) \end{cases}$$

Your PMM

$$\begin{cases} M(c) = \underline{M_0} + c\underline{M_1} \\ M(c)\left|\tilde{\psi}(c)\right\rangle = \tilde{E}(c)\left|\tilde{\psi}(c)\right\rangle \\ \left\langle\tilde{\psi}(c)\middle|\underline{X}\middle|\tilde{\psi}(c)\right\rangle = \tilde{f}(c) \end{cases}$$

$$\begin{cases} \dfrac{\partial\vec{v}}{\partial t} = A(\vec{c})\vec{v} + f(t, c, \vec{v}) \\ \vec{v}(t=0) = \vec{v}_0 \\ v_i(t) = \text{b.c.} \end{cases}$$

Okay it's not always *that* simple. . .

Build-a-PMM™

Your Underlying Equations

$$\begin{cases} H(c) = H_0 + cH_1 \\ H(c)\left|\psi(c)\right\rangle = E(c)\left|\psi(c)\right\rangle \\ \langle\psi(c)|O|\psi(c)\rangle = f(c) \end{cases}$$

$$\begin{cases} \dfrac{\partial\vec{v}}{\partial t} = A(\vec{c})\vec{v} + f(t, c, \vec{v}) \\ \vec{v}(t=0) = \vec{v}_0 \\ v_i(t) = \text{b.c.} \end{cases}$$

Your PMM

$$\begin{cases} M(c) = \underline{M_0} + c\underline{M_1} \\ M(c)\left|\tilde{\psi}(c)\right\rangle = \tilde{E}(c)\left|\tilde{\psi}(c)\right\rangle \\ \left\langle\tilde{\psi}(c)\middle|\underline{X}\middle|\tilde{\psi}(c)\right\rangle = \tilde{f}(c) \end{cases}$$

Okay it's not always *that* simple...

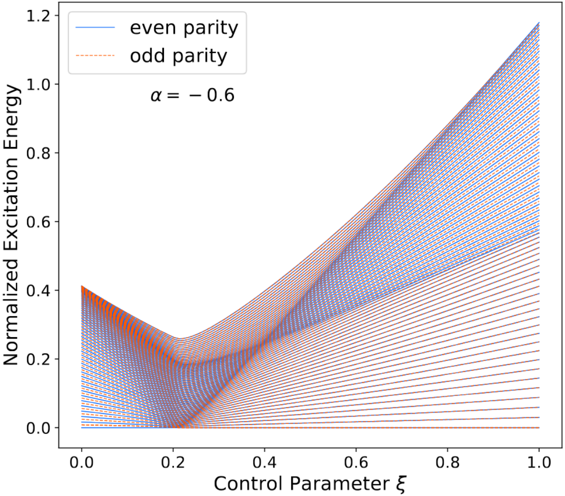
Then fit to whatever data you have!

Example: Anharmonic Lipkin Model

$$H(\xi;\alpha) = (1-\xi)\hat{n} + \frac{2\xi}{S}\left(S^2 - \hat{S}_x{}^2\right) + \frac{\alpha}{2S}\hat{n}(\hat{n}+1)$$

$$\hat{n} = \left(S + \hat{S}_z\right)$$

Example: Anharmonic Lipkin Model



Gamito, Khalouf-Rivera, et al. 2022

Example: Anharmonic Lipkin Model

$$H_\alpha(\xi) = H_0 + \xi H_1$$

$$\langle E_0(\xi)|\hat{n}|E_0(\xi)\rangle$$

Example: Anharmonic Lipkin Model

$$H_\alpha(\xi) = H_0 + \xi H_1$$

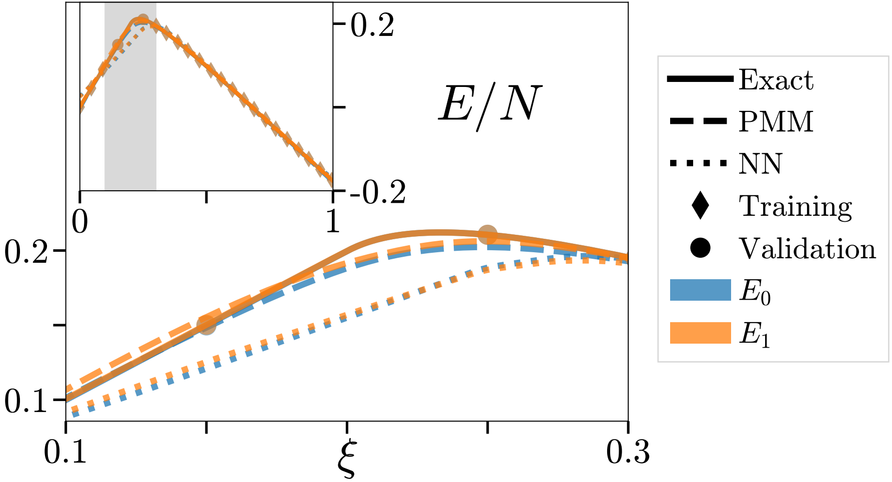
$$\langle E_0(\xi)|\hat{n}|E_0(\xi)\rangle$$

PMM

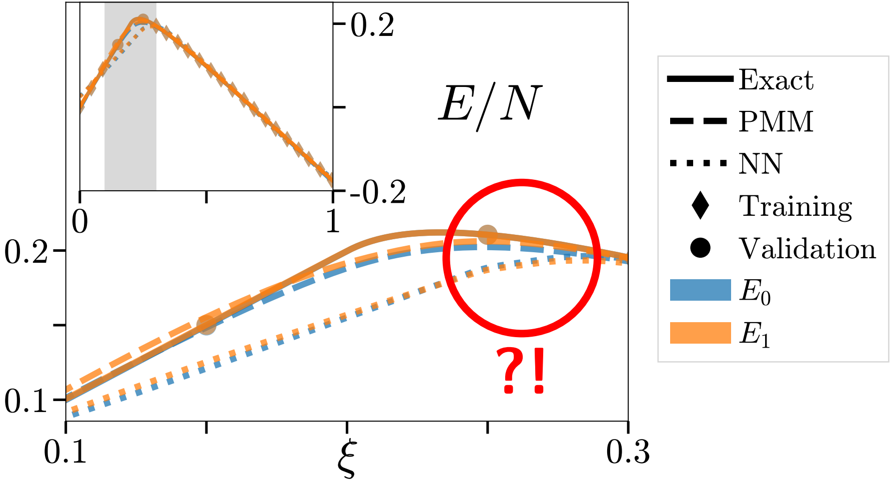
$$M_\alpha(\xi) = \underline{M_0} + \xi \underline{M_1}$$

$$\left\langle \tilde{E}_0(\xi) \middle| \underline{X} \middle| \tilde{E}_0(\xi) \right\rangle$$

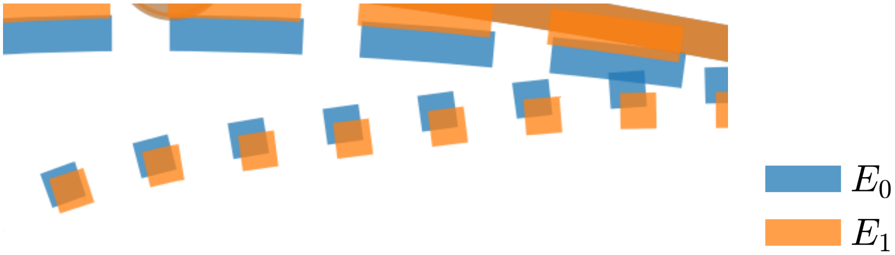
Example: Anharmonic Lipkin Model



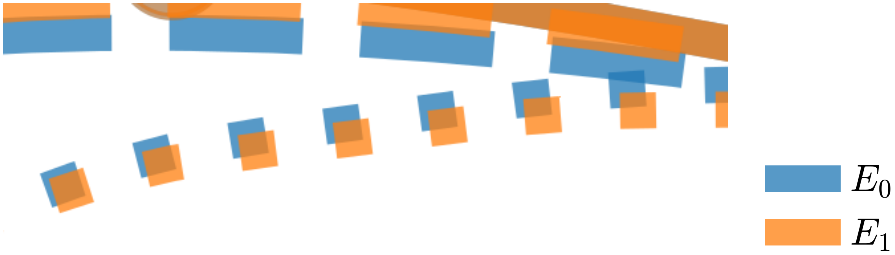
Example: Anharmonic Lipkin Model



Example: Anharmonic Lipkin Model



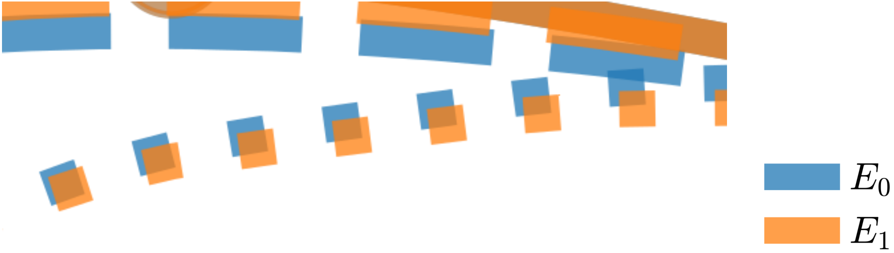
Example: Anharmonic Lipkin Model



Neural Network:
“Sometimes, $E_0 > E_1$”

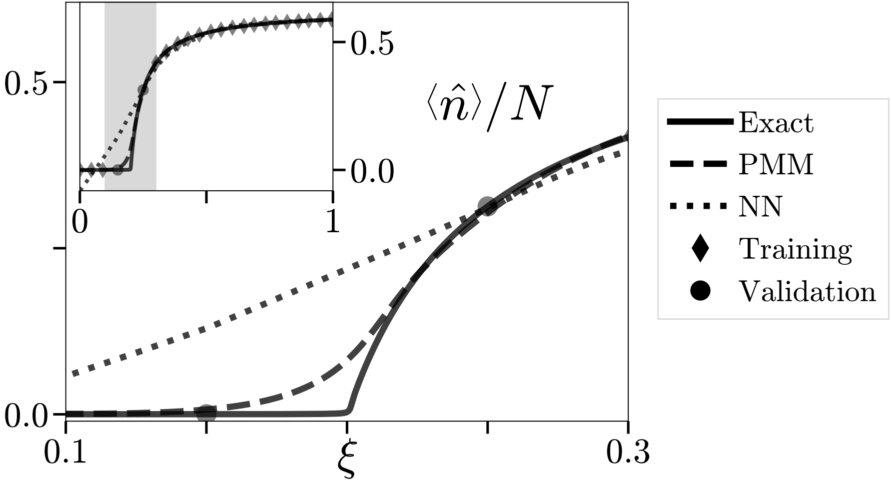
Example: Anharmonic Lipkin Model

WHAT



Neural Network:
"Sometimes, $E_0 > E_1$"

Example: Anharmonic Lipkin Model



Isn't this just Eigenvector Continuation?

Isn't this just Eigenvector Continuation?

Eigenvector Continuation

Isn't this just Eigenvector Continuation?

Eigenvector Continuation

Requires:

- Snapshots of full eigenvectors

Isn't this just Eigenvector Continuation?

Eigenvector Continuation

Requires:

- Snapshots of full eigenvectors
- The evaluation of $H(c)$

Isn't this just Eigenvector Continuation?

Eigenvector Continuation

Requires:

- Snapshots of full eigenvectors
- The evaluation of $H(c)$

Isn't this just Eigenvector Continuation?

Eigenvector Continuation

Requires:

- Snapshots of full eigenvectors
- The evaluation of $H(c)$

Pros:

- Exponential convergence

Isn't this just Eigenvector Continuation?

Eigenvector Continuation

Requires:

- Snapshots of full eigenvectors
- The evaluation of $H(c)$

Pros:

- Exponential convergence
- Constructed rather than trained

Isn't this just Eigenvector Continuation?

Eigenvector Continuation

Requires:

- Snapshots of full eigenvectors
- The evaluation of $H(c)$

Pros:

- Exponential convergence
- Constructed rather than trained

Isn't this just Eigenvector Continuation?

Eigenvector Continuation

Requires:

- Snapshots of full eigenvectors
- The evaluation of $H(c)$

Pros:

- Exponential convergence
- Constructed rather than trained

Cons:

- Poor computational scaling

Isn't this just Eigenvector Continuation?

Eigenvector Continuation

Requires:

- Snapshots of full eigenvectors
- The evaluation of $H(c)$

Pros:

- Exponential convergence
- Constructed rather than trained

Cons:

- Poor computational scaling
- Limited in scope

Isn't this just Eigenvector Continuation?

Eigenvector Continuation

Requires:

- Snapshots of full eigenvectors
- The evaluation of $H(c)$

Pros:

- Exponential convergence
- Constructed rather than trained

Cons:

- Poor computational scaling
- Limited in scope
- Curse of dimensionality

Isn't this just Eigenvector Continuation?

Eigenvector Continuation

Requires:

- Snapshots of full eigenvectors
- The evaluation of $H(c)$

Pros:

- Exponential convergence
- Constructed rather than trained

Cons:

- Poor computational scaling
- Limited in scope
- Curse of dimensionality

$$H(c) = \frac{1}{2N}\sum_{i}^{N}(\sigma_i^z + c\sigma_i^x)$$

Isn't this just Eigenvector Continuation?

Eigenvector Continuation

Requires:

- Snapshots of full eigenvectors
- The evaluation of $H(c)$

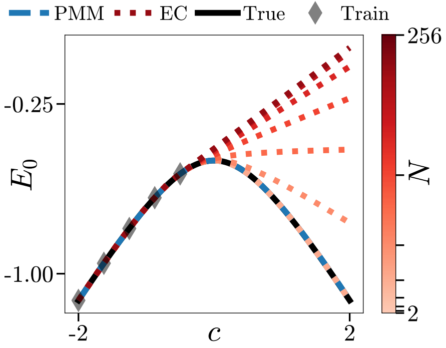
Pros:

- Exponential convergence
- Constructed rather than trained

Cons:

- Poor computational scaling
- Limited in scope
- Curse of dimensionality

$$H(c) = \frac{1}{2N}\sum_{i}^{N}(\sigma_i^z + c\sigma_i^x)$$



Time evolution on a quantum computer of a non-commuting Hamiltonian

$$H = H_0 + H_1 + \cdots + H_l$$

Time evolution on a quantum computer of a non-commuting Hamiltonian

$$H = H_0 + H_1 + \cdots + H_l$$

$$U(dt) = e^{-iHdt} \approx e^{-iH_0 dt} e^{-iH_1 dt} \cdots e^{-iH_l dt}$$

Time evolution on a quantum computer of a non-commuting Hamiltonian

$$H = H_0 + H_1 + \cdots + H_l$$

$$U(dt) = e^{-iHdt} \approx e^{-iH_0 dt} e^{-iH_1 dt} \cdots e^{-iH_l dt}$$

PMM

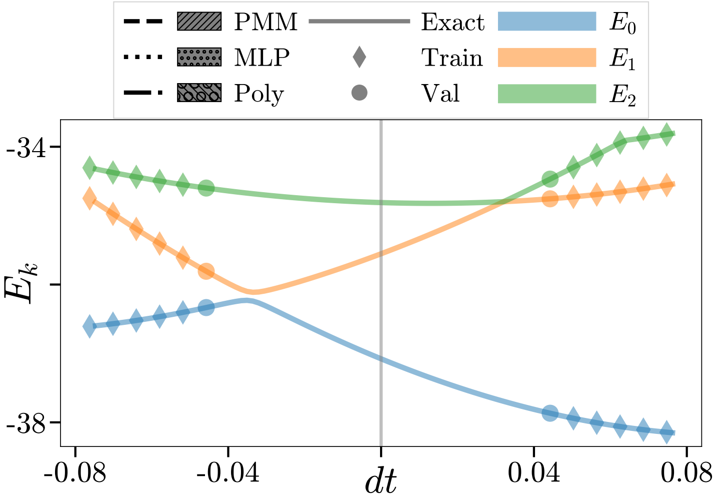
$$\tilde{U}(dt) = e^{-i\underline{M_0}dt} e^{-i\underline{M_1}dt} \cdots e^{-i\underline{M_l}dt}$$

1D Heisenberg Model with Dzyaloshinskii-Moriya Interaction

$$H = B\sum_{i}^{N} r_i\sigma_i^z + J\sum_{u\in\{x,y,z\}}\sum_{i}^{N}\sigma_i^u\sigma_{i+1}^u + D\sum_{i}^{N}\left(\sigma_i^x\sigma_{i+1}^y - \sigma_i^y\sigma_{i+1}^x\right)$$

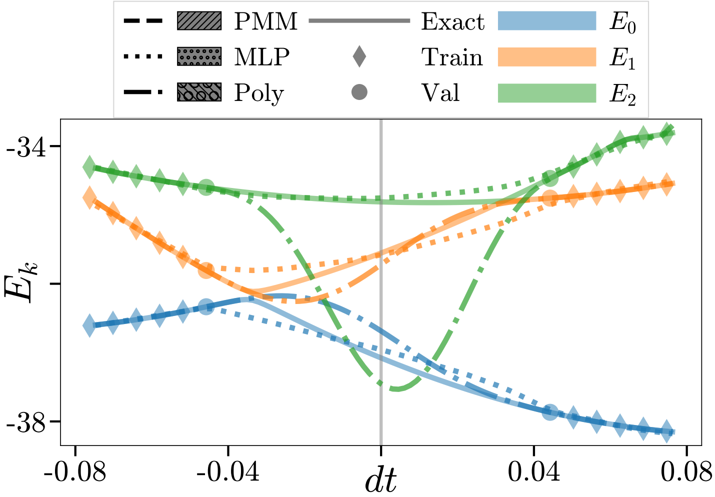
1D Heisenberg Model with Dzyaloshinskii-Moriya Interaction

$$H = B\sum_{i}^{N} r_i\sigma_i^z + J\sum_{u\in\{x,y,z\}}\sum_{i}^{N}\sigma_i^u\sigma_{i+1}^u + D\sum_{i}^{N}\left(\sigma_i^x\sigma_{i+1}^y - \sigma_i^y\sigma_{i+1}^x\right)$$



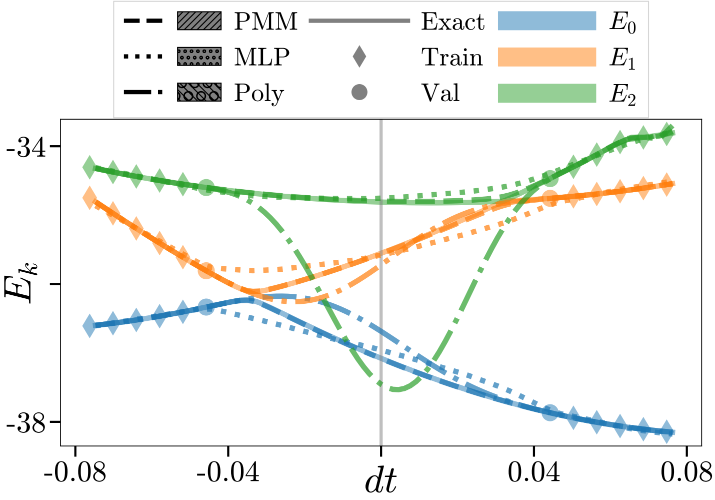
1D Heisenberg Model with Dzyaloshinskii-Moriya Interaction

$$H = B\sum_i^N r_i \sigma_i^z + J \sum_{u\in\{x,y,z\}} \sum_i^N \sigma_i^u \sigma_{i+1}^u + D \sum_i^N \left(\sigma_i^x \sigma_{i+1}^y - \sigma_i^y \sigma_{i+1}^x\right)$$



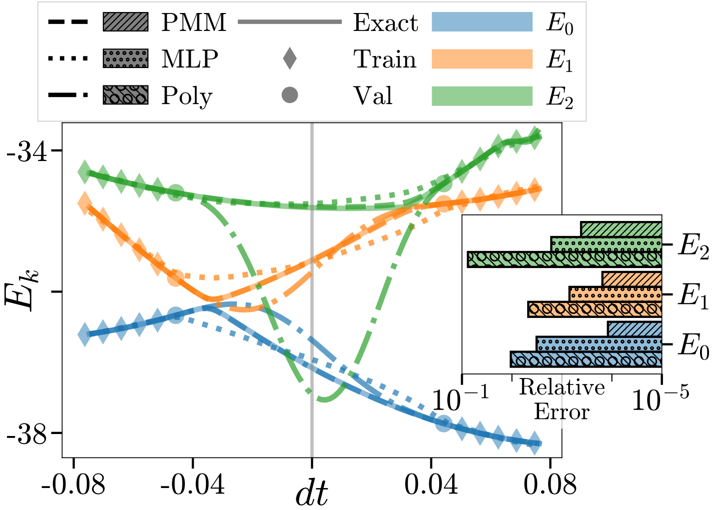
1D Heisenberg Model with Dzyaloshinskii-Moriya Interaction

$$H = B\sum_i^N r_i\sigma_i^z + J\sum_{u\in\{x,y,z\}}\sum_i^N \sigma_i^u\sigma_{i+1}^u + D\sum_i^N\left(\sigma_i^x\sigma_{i+1}^y - \sigma_i^y\sigma_{i+1}^x\right)$$



1D Heisenberg Model with Dzyaloshinskii-Moriya Interaction

$$H = B\sum_{i}^{N} r_i\sigma_i^z + J\sum_{u\in\{x,y,z\}}\sum_{i}^{N}\sigma_i^u\sigma_{i+1}^u + D\sum_{i}^{N}\left(\sigma_i^x\sigma_{i+1}^y - \sigma_i^y\sigma_{i+1}^x\right)$$



Practical Summary

Practical Summary

The PMM for your system *is* your system!

Practical Summary

The PMM for your system *is* your system!

- Use the underlying equations of your system

Practical Summary

The PMM for your system *is* your system!

- Use the underlying equations of your system
- The PMM will be a low-dimension analog of your system

Practical Summary

The PMM for your system *is* your system!

- Use the underlying equations of your system
- The PMM will be a low-dimension analog of your system
- Low-order equations are perfectly fine

Practical Summary

The PMM for your system *is* your system!

- Use the underlying equations of your system
- The PMM will be a low-dimension analog of your system
- Low-order equations are perfectly fine
- Due to low dimensionality, exact solutions are possible

Practical Summary

The PMM for your system *is* your system!

- Use the underlying equations of your system
- The PMM will be a low-dimension analog of your system
- Low-order equations are perfectly fine
- Due to low dimensionality, exact solutions are possible

Practical Summary

The PMM for your system *is* your system!

- Use the underlying equations of your system
- The PMM will be a low-dimension analog of your system
- Low-order equations are perfectly fine
- Due to low dimensionality, exact solutions are possible

Two outstanding questions:

Practical Summary

The PMM for your system *is* your system!

- Use the underlying equations of your system
- The PMM will be a low-dimension analog of your system
- Low-order equations are perfectly fine
- Due to low dimensionality, exact solutions are possible

Two outstanding questions:

- *What if I don't know the underlying equations?*

Practical Summary

The PMM for your system *is* your system!

- Use the underlying equations of your system
- The PMM will be a low-dimension analog of your system
- Low-order equations are perfectly fine
- Due to low dimensionality, exact solutions are possible

Two outstanding questions:

- *What if I don't know the underlying equations?*
- *What if my problem is nonlinear?*

arXiv:2404.11566

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

Rahul Somasundaram,[1,2,*] Cassandra L. Armstrong,[3]
Pablo Giuliani,[4,5] Kyle Godbey,[4] Stefano Gandolfi,[1] and Ingo Tews[1]

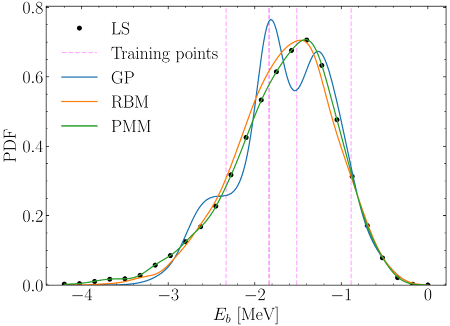
[1] *Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*
[2] *Department of Physics, Syracuse University, Syracuse, NY 13244, USA*
[3] *Intelligence and Space Research Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*
[4] *Facility for Rare Isotope Beams, Michigan State University, East Lansing, Michigan 48824, USA*
[5] *Department of Statistics and Probability, Michigan State University, East Lansing, Michigan 48824, USA*

arXiv:2404.11566

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

Rahul Somasundaram,[1,2,*] Cassandra L. Armstrong,[3]
Pablo Giuliani,[4,5] Kyle Godbey,[4] Stefano Gandolfi,[1] and Ingo Tews[1]

[1] *Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*
[2] *Department of Physics, Syracuse University, Syracuse, NY 13244, USA*
[3] *Intelligence and Space Research Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*
[4] *Facility for Rare Isotope Beams, Michigan State University, East Lansing, Michigan 48824, USA*
[5] *Department of Statistics and Probability, Michigan State University, East Lansing, Michigan 48824, USA*

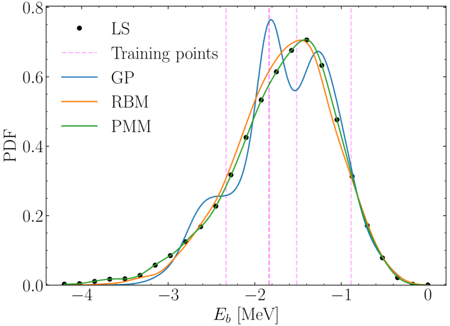


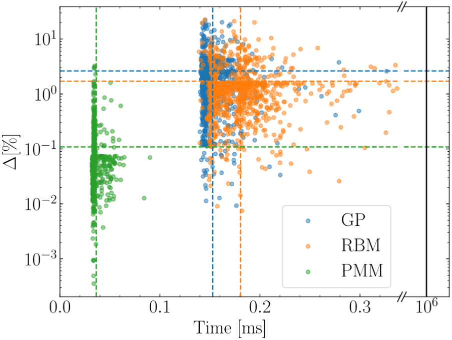
arXiv:2404.11566

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

Rahul Somasundaram,[1,2,*] Cassandra L. Armstrong,[3]
Pablo Giuliani,[4,5] Kyle Godbey,[4] Stefano Gandolfi,[1] and Ingo Tews[1]

[1] *Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*
[2] *Department of Physics, Syracuse University, Syracuse, NY 13244, USA*
[3] *Intelligence and Space Research Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*
[4] *Facility for Rare Isotope Beams, Michigan State University, East Lansing, Michigan 48824, USA*
[5] *Department of Statistics and Probability, Michigan State University, East Lansing, Michigan 48824, USA*





arXiv:2405.20558

Towards accelerated nuclear-physics parameter estimation from binary neutron star mergers: Emulators for the Tolman-Oppenheimer-Volkoff equations

Brendan T. Reed,[1] Rahul Somasundaram,[1,2] Soumi De,[1] Cassandra L. Armstrong,[3] Pablo Giuliani,[4] Collin Capano,[2,5] Duncan A. Brown,[2] and Ingo Tews[1]

[1] Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA
[2] Department of Physics, Syracuse University, Syracuse, NY 13244, USA
[3] Intelligence and Space Research Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA
[4] Facility for Rare Isotope Beams, Michigan State University, East Lansing, Michigan 48824, USA
[5] Physics Department, University of Massachusetts Dartmouth, North Dartmouth, MA 02747, USA

arXiv:2410.00247

Inferring three-nucleon couplings from multi-messenger neutron-star observations

Rahul Somasundaram[1,2], Isak Svensson[3,4,5], Soumi De[2], Andrew E. Deneris[6], Yannick Dietz[3,4], Philippe Landry[7,8], Achim Schwenk[3,4,5], and Ingo Tews[2]

[1]Department of Physics, Syracuse University, Syracuse, NY 13244, USA

[2]Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

[3]Technische Universität Darmstadt, Department of Physics, 64289 Darmstadt, Germany

[4]ExtreMe Matter Institute EMMI, GSI Helmholtzzentrum für Schwerionenforschung GmbH, 64291 Darmstadt, Germany

[5]Max-Planck-Institut für Kernphysik, Saupfercheckweg 1, 69117 Heidelberg, Germany

[6]Computer, Computational, and Statistical Sciences Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

[7]Canadian Institute for Theoretical Astrophysics, University of Toronto, Toronto, Ontario M5S 3H8, Canada

[8]Perimeter Institute for Theoretical Physics, Waterloo, Ontario N2L 2Y5, Canada

From Data to Discovery

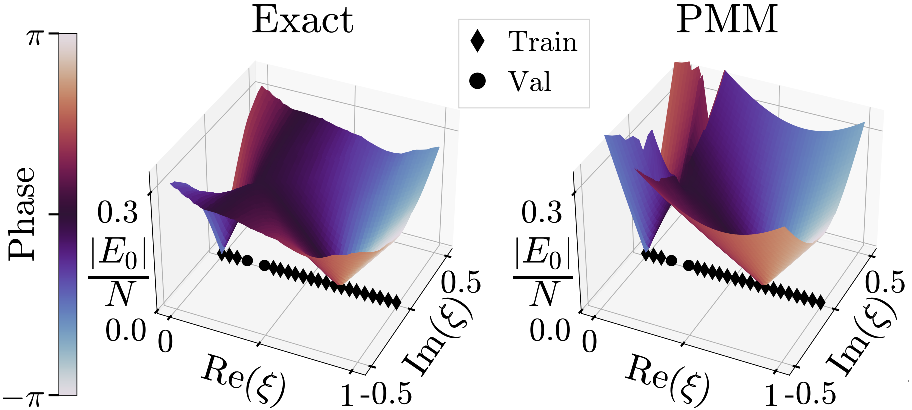
Anharmonic Lipkin Model

Anharmonic Lipkin Model

$$\begin{Bmatrix} H_\alpha(\xi) = H_0 + \xi H_1 \\ \left\langle E_0(\xi) \middle| \hat{n} \middle| E_0(\xi) \right\rangle \end{Bmatrix} \sim \begin{Bmatrix} M_\alpha(\xi) = \underline{M_0} + \xi \underline{M_1} \\ \left\langle \tilde{E}_0(\xi) \middle| \underline{X} \middle| \tilde{E}_0(\xi) \right\rangle \end{Bmatrix}$$

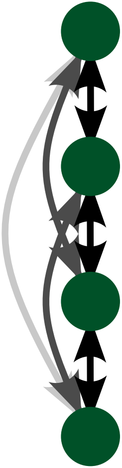
Anharmonic Lipkin Model

$$\begin{Bmatrix} H_\alpha(\xi) = H_0 + \xi H_1 \\ \langle E_0(\xi)|\hat{n}|E_0(\xi)\rangle \end{Bmatrix} \sim \begin{Bmatrix} M_\alpha(\xi) = \underline{M_0} + \xi \underline{M_1} \\ \langle \tilde{E}_0(\xi)|\underline{X}|\tilde{E}_0(\xi)\rangle \end{Bmatrix}$$

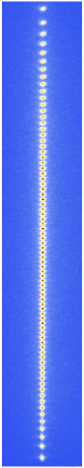


Generalized Long-Range Spin Model

$$H = -\sum_{i \neq j} J_{ij} \left(\gamma_x \sigma_i^x \sigma_j^x + \gamma_y \sigma_i^y \sigma_j^y \right) - B \sum_i \sigma_i^z$$

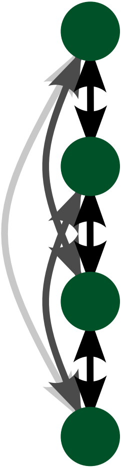


Generalized Long-Range Spin Model

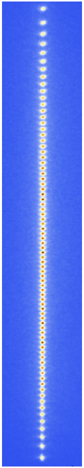


^{171}Yb^{+}

$$H = -\sum_{i \neq j} J_{ij} \left(\gamma_x \sigma_i^x \sigma_j^x + \gamma_y \sigma_i^y \sigma_j^y \right) - B \sum_i \sigma_i^z$$



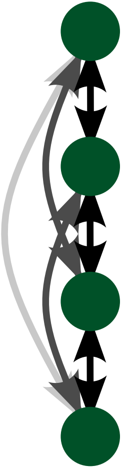
Generalized Long-Range Spin Model



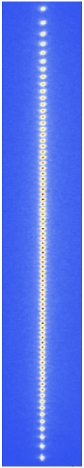
^{171}Yb^{+}

$$H = -\sum_{i\neq j} J_{ij}\left(\gamma_x \sigma_i^x \sigma_j^x + \gamma_y \sigma_i^y \sigma_j^y\right) - B\sum_i \sigma_i^z$$

$$M = -\sum_{i\neq j} \underline{\tilde{J}_{ij}}\left(\underline{\tilde{\gamma}_x} \sigma_i^x \sigma_j^x + \underline{\tilde{\gamma}_y} \sigma_i^y \sigma_j^y\right) - \underline{\tilde{B}}\sum_i \sigma_i^z$$



Generalized Long-Range Spin Model

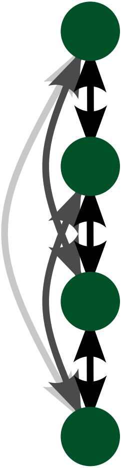


^{171}Yb^{+}

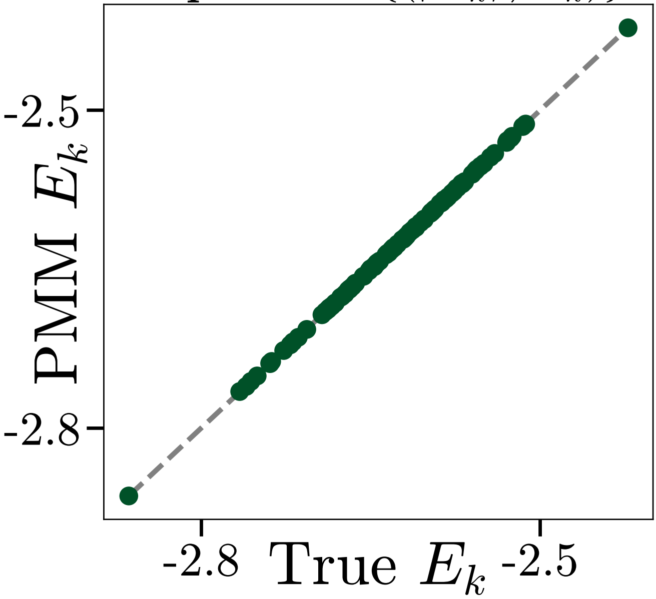
$$H = -\sum_{i\neq j} J_{ij}\left(\gamma_x \sigma_i^x \sigma_j^x + \gamma_y \sigma_i^y \sigma_j^y\right) - B\sum_i \sigma_i^z$$

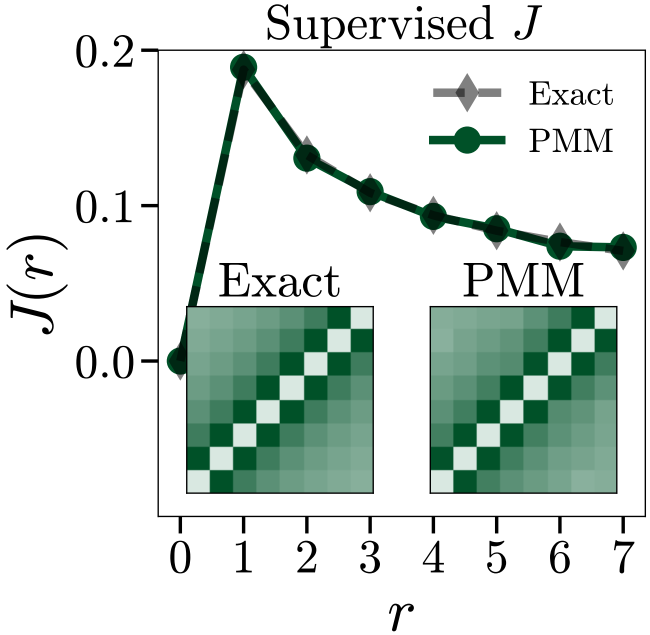
$$M = -\sum_{i\neq j} \underline{\tilde{J}_{ij}}\left(\underline{\tilde{\gamma}_x} \sigma_i^x \sigma_j^x + \underline{\tilde{\gamma}_y} \sigma_i^y \sigma_j^y\right) - \underline{\tilde{B}}\sum_i \sigma_i^z$$

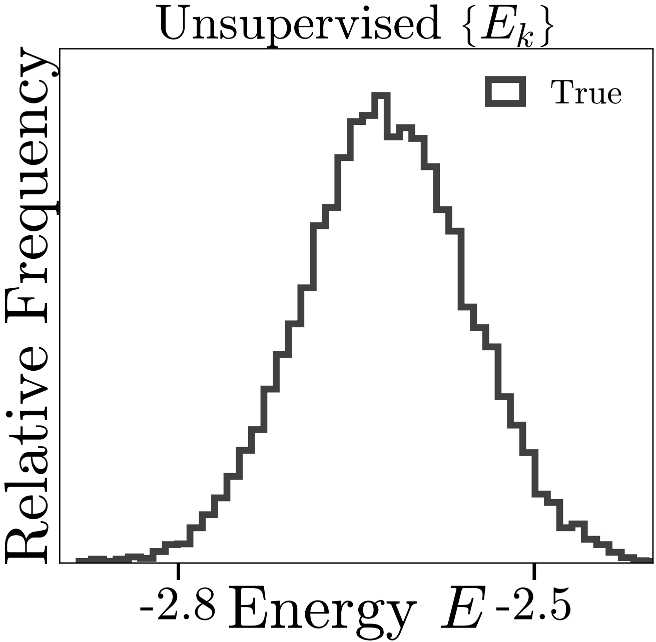
Supervised and **Unsupervised**

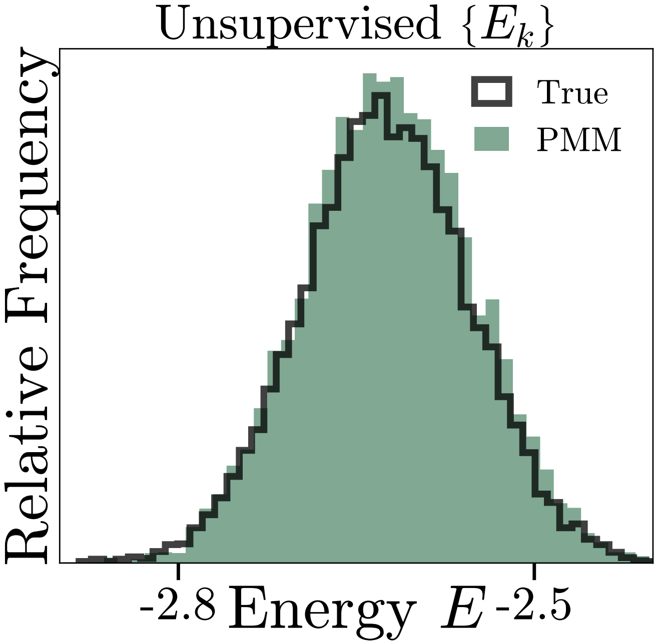


Supervised $\{(|\Psi_k\rangle, E_k)\}$

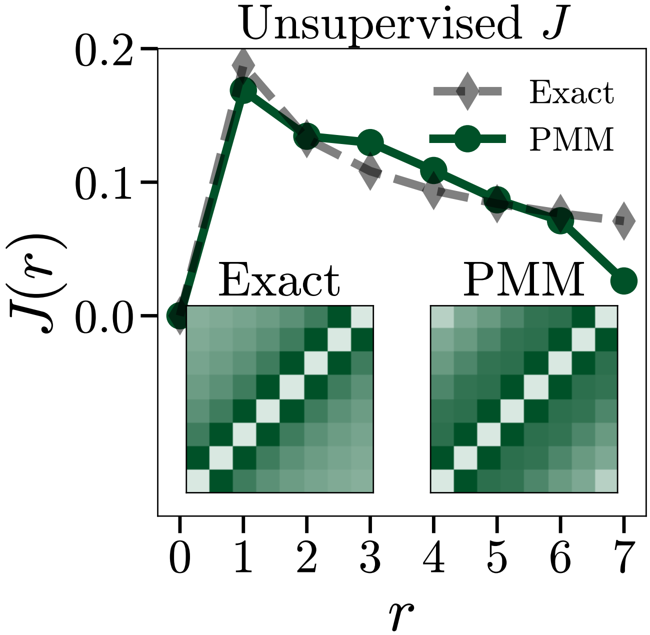


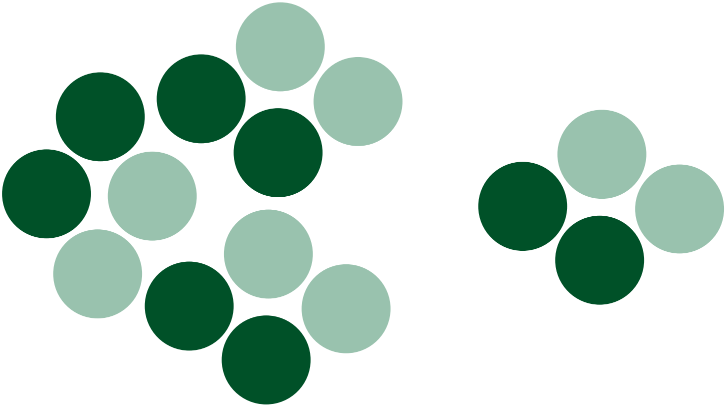

Supervised J
0.2
0.1
0.0
$J(r)$
Exact
PMM
Exact
PMM
0
1
2
3
4
5
6
7
r

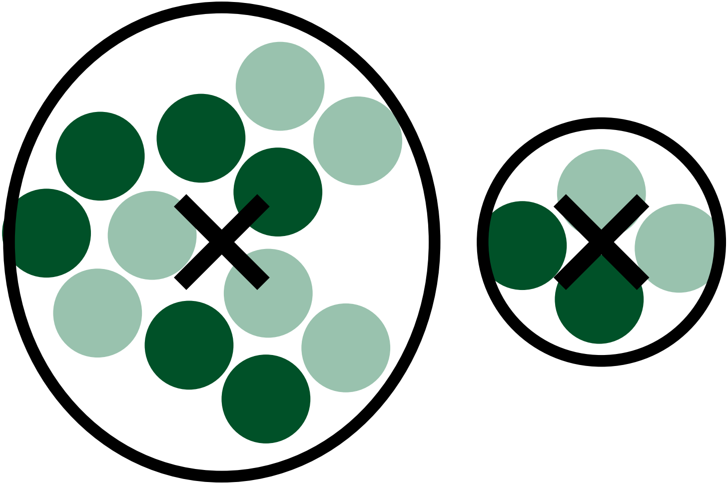

Unsupervised $\{E_k\}$
True
Relative Frequency
-2.8
-2.5
Energy E



Unsupervised $\{E_k\}$
Relative Frequency
True
PMM
-2.8
-2.5
Energy E


Unsupervised J
Exact
PMM
$J(r)$
r
0.2
0.1
0.0
0 1 2 3 4 5 6 7
Exact
PMM






^{12}C
α

$$M = T + \underline{V}\left(\left|\vec{r}_{^{12}\mathrm{C}} - \vec{r}_{\alpha}\right|\right)$$

$$M_2 = T + \underline{V}\left(\left|\vec{r}_{^{12}\mathsf{C}} - \vec{r}_{\alpha}\right|\right)$$

$$M_3 = T + \underline{V}\left(\vec{r}_{^{8}\mathsf{Be}}, \vec{r}_{\alpha_1}, \vec{r}_{\alpha_2}\right)$$

$$M_4 = T + \underline{V}\left(\vec{r}_{\alpha_1}, \vec{r}_{\alpha_2}, \vec{r}_{\alpha_3}, \vec{r}_{\alpha_4}\right)$$

Extending PMMs

Two main requests:

Extending PMMs

Two main requests:

- Predicting unseen observables

Extending PMMs

Two main requests:

- Predicting unseen observables
  - State-vector emulation

Extending PMMs

Two main requests:

- Predicting unseen observables
  - State-vector emulation
- Nonlinear problems

Extending PMMs

Two main requests:

- Predicting unseen observables
  - State-vector emulation
- Nonlinear problems
  - Density-dependent Hamiltonians

Extending PMMs

Two main requests:

- Predicting unseen observables
  - State-vector emulation
- Nonlinear problems
  - Density-dependent Hamiltonians

Extending PMMs

Two main requests:

- Predicting unseen observables
  - State-vector emulation
- Nonlinear problems
  - Density-dependent Hamiltonians

These are the same problem!

Proper Orthogonal Decomposition

Proper Orthogonal Decomposition

Given:

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$
- The governing equations for $\vec{v}$

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$
- The governing equations for $\vec{v}$

Example

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$
- The governing equations for $\vec{v}$

Example

$$\{v(t_0), v(t_1), \ldots, v(t_p)\}$$

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$
- The governing equations for $\vec{v}$

Example

$$\{v(t_0), v(t_1), \ldots, v(t_p)\}$$

$$dv/dt = Av + f(t, v)$$

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$
- The governing equations for $\vec{v}$

POD:

Example

$$\{v(t_0), v(t_1), \ldots, v(t_p)\}$$

$$dv/dt = Av + f(t, v)$$

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$
- The governing equations for $\vec{v}$

POD:

- r most important principal components of the snapshots

Example

$$\{v(t_0), v(t_1), \ldots, v(t_p)\}$$

$$dv/dt = Av + f(t, v)$$

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$
- The governing equations for $\vec{v}$

POD:

- r most important principal components of the snapshots

Example

$$\{v(t_0), v(t_1), \ldots, v(t_p)\}$$

$$dv/dt = Av + f(t, v)$$

$$V = \begin{bmatrix} v(t_0) & v(t_1) & \cdots & v(t_p) \end{bmatrix}$$

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$
- The governing equations for $\vec{v}$

POD:

- r most important principal components of the snapshots

Example

$$\{v(t_0), v(t_1), \ldots, v(t_p)\}$$

$$dv/dt = Av + f(t, v)$$

$$V = \begin{bmatrix} v(t_0) & v(t_1) & \cdots & v(t_p) \end{bmatrix}$$

$$V \approx P\Sigma_r Q^\dagger \quad \text{SVD}$$

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$
- The governing equations for $\vec{v}$

POD:

- r most important principal components of the snapshots
- Project the governing equations onto the POD basis

Example

$$\{v(t_0), v(t_1), \ldots, v(t_p)\}$$

$$dv/dt = Av + f(t, v)$$

$$V = \begin{bmatrix} v(t_0) & v(t_1) & \cdots & v(t_p) \end{bmatrix}$$

$$V \approx P\Sigma_r Q^\dagger \quad \text{SVD}$$

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$
- The governing equations for $\vec{v}$

POD:

- r most important principal components of the snapshots
- Project the governing equations onto the POD basis

Example

$$\{v(t_0), v(t_1), \ldots, v(t_p)\}$$

$$dv/dt = Av + f(t, v)$$

$$V = \begin{bmatrix} v(t_0) & v(t_1) & \cdots & v(t_p) \end{bmatrix}$$

$$V \approx P\Sigma_r Q^\dagger \quad \text{SVD}$$

$$u = P^\dagger v$$

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$
- The governing equations for $\vec{v}$

POD:

- r most important principal components of the snapshots
- Project the governing equations onto the POD basis

Example

$$\{v(t_0), v(t_1), \ldots, v(t_p)\}$$

$$dv/dt = Av + f(t, v)$$

$$V = \begin{bmatrix} v(t_0) & v(t_1) & \cdots & v(t_p) \end{bmatrix}$$

$$V \approx P\Sigma_r Q^\dagger \quad \text{SVD}$$

$$u = P^\dagger v$$

$$\tilde{A} = P^\dagger A P$$

$$du/dt = \tilde{A}u + P^\dagger f(t, Pu)$$

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$
- The governing equations for $\vec{v}$

POD:

- r most important principal components of the snapshots
- Project the governing equations onto the POD basis
- Solve the projected equations

Example

$$\{v(t_0), v(t_1), \ldots, v(t_p)\}$$

$$dv/dt = Av + f(t, v)$$

$$V = \begin{bmatrix} v(t_0) & v(t_1) & \cdots & v(t_p) \end{bmatrix}$$

$$V \approx P\Sigma_r Q^\dagger \quad \text{SVD}$$

$$u = P^\dagger v$$

$$\tilde{A} = P^\dagger A P$$

$$du/dt = \tilde{A}u + P^\dagger f(t, Pu)$$

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$
- The governing equations for $\vec{v}$

POD:

- r most important principal components of the snapshots
- Project the governing equations onto the POD basis
- Solve the projected equations
- Reconstruct the full solution

Example

$$\{v(t_0), v(t_1), \ldots, v(t_p)\}$$
$$dv/dt = Av + f(t,v)$$

$$V = \begin{bmatrix} v(t_0) & v(t_1) & \cdots & v(t_p) \end{bmatrix}$$
$$V \approx P\Sigma_r Q^\dagger \quad \text{SVD}$$

$$u = P^\dagger v$$
$$\tilde{A} = P^\dagger A P$$
$$du/dt = \tilde{A}u + P^\dagger f(t, Pu)$$

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$
- The governing equations for $\vec{v}$

POD:

- r most important principal components of the snapshots
- Project the governing equations onto the POD basis
- Solve the projected equations
- Reconstruct the full solution

Example

$$\{v(t_0), v(t_1), \ldots, v(t_p)\}$$

$$dv/dt = Av + f(t, v)$$

$$V = \begin{bmatrix} v(t_0) & v(t_1) & \cdots & v(t_p) \end{bmatrix}$$

$$V \approx P\Sigma_r Q^\dagger \quad \text{SVD}$$

$$\begin{aligned} u &= P^\dagger v \\ \tilde{A} &= P^\dagger A P \\ du/dt &= \tilde{A}u + P^\dagger f(t, Pu) \\ v &\approx Pu \end{aligned}$$

Proper Orthogonal Decomposition

Given:

- A set of state snapshots $\{\vec{v}_i\}$
- The governing equations for $\vec{v}$

POD:

- r most important principal components of the snapshots
- Project the governing equations onto the POD basis
- Solve the projected equations
- Reconstruct the full solution

Example

$$\{v(t_0), v(t_1), \ldots, v(t_p)\}$$
$$dv/dt = Av + f(t, v)$$

$$V = \begin{bmatrix} v(t_0) & v(t_1) & \cdots & v(t_p) \end{bmatrix}$$
$$V \approx P\Sigma_r Q^\dagger \quad \text{SVD}$$

$$u = P^\dagger v$$
$$\tilde{A} = P^\dagger A P$$
$$du/dt = \tilde{A}u + {\color{red} P^\dagger f(t, Pu)}$$
$$v \approx Pu$$

POD for Nonlinear Problems

POD for Nonlinear Problems

Consider the simplest nonlinearity:

POD for Nonlinear Problems

Consider the simplest nonlinearity:

$$f(t, v) = v^2 = v \odot v$$

POD for Nonlinear Problems

Consider the simplest nonlinearity:

$$f(t, v) = v^2 = v \odot v$$

POD:

POD for Nonlinear Problems

Consider the simplest nonlinearity:

$$f(t, v) = v^2 = v \odot v$$

POD:

$$P^\dagger f(t, Pu) = P^\dagger \left[(Pu) \odot (Pu)\right]$$

POD for Nonlinear Problems

Consider the simplest nonlinearity:

$$f(t, v) = v^2 = v \odot v$$

POD:

$$\begin{aligned} P^\dagger f(t, Pu) &= P^\dagger \left[(Pu) \odot (Pu)\right] \\ &\stackrel{i}{=} (P^\dagger)_{i\mu} P_{\mu j} P_{\mu k} u_j u_k \end{aligned}$$

POD for Nonlinear Problems

Consider the simplest nonlinearity:

$$f(t, v) = v^2 = v \odot v$$

POD:

$$\begin{aligned} P^\dagger f(t, Pu) &= P^\dagger \left[(Pu) \odot (Pu) \right] \\ &\overset{i}{=} (P^\dagger)_{i\mu} P_{\mu j} P_{\mu k} u_j u_k \\ &\equiv P^\dagger (P \otimes P)(u \otimes u) \end{aligned}$$

POD for Nonlinear Problems

Consider the simplest nonlinearity:

$$f(t, v) = v^2 = v \odot v$$

POD:

$$\begin{aligned} P^\dagger f(t, Pu) &= P^\dagger \left[(Pu) \odot (Pu)\right] \\ &\overset{i}{=} (P^\dagger)_{i\mu} P_{\mu j} P_{\mu k} u_j u_k \\ &\equiv P^\dagger (P \otimes P)(u \otimes u) \end{aligned}$$

Generalizing to polynomial nonlinearities:

$$v^n = v \odot^n v$$

$$\uparrow\downarrow$$

$$P^\dagger (P \otimes^n P)(u \otimes^n u) \overset{i}{=} (P^\dagger)_{i\mu} P_{\mu j_1} P_{\mu j_2} \cdots P_{\mu j_n} u_{j_1} u_{j_2} \cdots u_{j_n}$$

Back to PMMs

$$P \bigotimes^n P \rightarrow \underline{\mathbb{T}^n} \equiv \underline{T_{j_1 j_2 \cdots j_n}}$$

Gross Pitaevskii Equation

$$i\frac{\partial\Psi}{\partial t} = H_0\Psi + g\left|\Psi\right|^2\Psi$$

Gross Pitaevskii Equation

$$i\frac{\partial \Psi}{\partial t} = H_0 \Psi + g \left|\Psi\right|^2 \Psi$$

Gross Pitaevskii Equation

$$i\frac{\partial \Psi}{\partial t} = H_0 \Psi + g\Psi \odot \Psi^* \odot \Psi$$

Gross Pitaevskii Equation

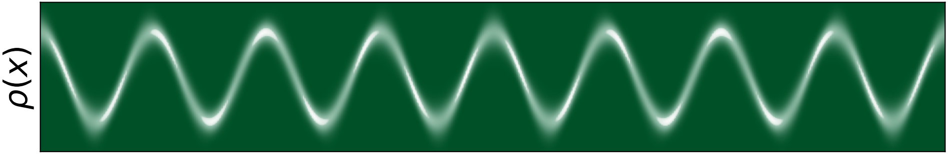
$$i\frac{\partial \Psi}{\partial t} = H_0\Psi + g\Psi \odot \Psi^* \odot \Psi$$

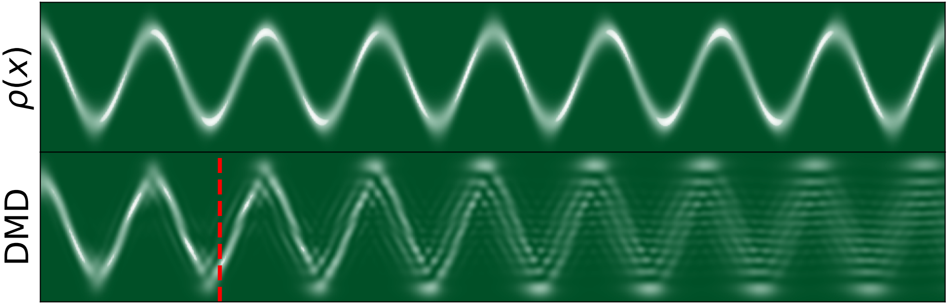
$$i\frac{\partial \psi}{\partial t} = \underline{\tilde{H}_0}\psi + g\underline{\mathbb{T}^3}\left(\psi \otimes \psi^* \otimes \psi\right)$$

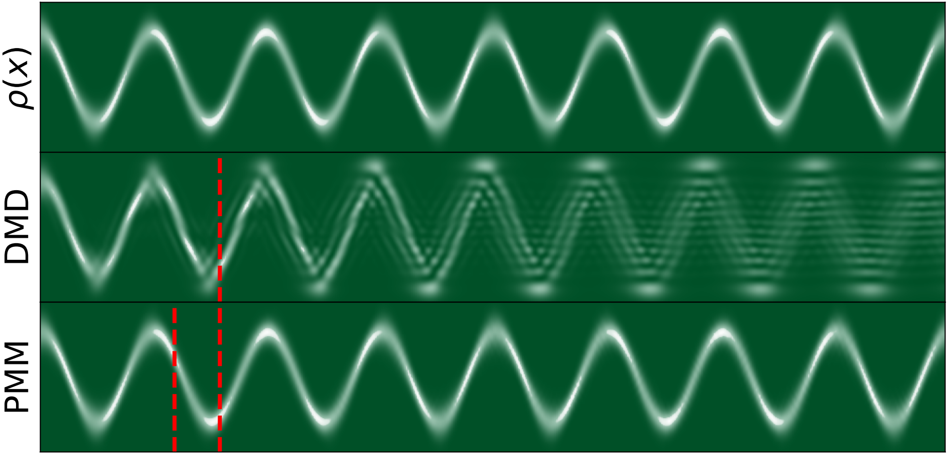
Gross Pitaevskii Equation

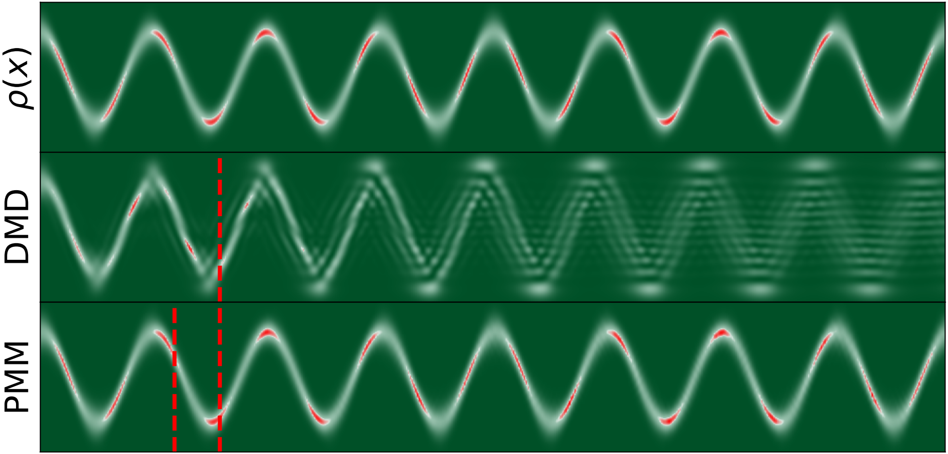
$$i\frac{\partial \Psi}{\partial t} = H_0\Psi + g\Psi \odot \Psi^* \odot \Psi$$

$$i\frac{\partial \psi}{\partial t} = \underline{\tilde{H}_0}\psi + g\underline{T_{jkl}}\psi_j\psi_k^*\psi_l$$


$\rho(x)$


$\rho(x)$
DMD


$\rho(x)$
DMD
PMM


$\rho(x)$
DMD
PMM

Beyond POD

What if the snapshots aren't of the state vector?

Beyond POD

What if the snapshots aren't of the state vector?

Taking Gross Pitaevskii as an example, with $\{\rho(t_i)\} \to P_\rho$

Beyond POD

What if the snapshots aren't of the state vector?

Taking Gross Pitaevskii as an example, with $\{\rho(t_i)\} \to P_\rho$

$$\psi = P_\Psi^\dagger \Psi$$

Beyond POD

What if the snapshots aren't of the state vector?

Taking Gross Pitaevskii as an example, with $\{\rho(t_i)\} \to P_\rho$

$$\psi = P_\Psi^\dagger \Psi$$
$$q = P_\rho^\dagger \rho$$

Beyond POD

What if the snapshots aren't of the state vector?

Taking Gross Pitaevskii as an example, with $\{\rho(t_i)\} \to P_\rho$

$$\psi = P_\Psi^\dagger \Psi$$

$$q = P_\rho^\dagger \rho = P_\rho^\dagger (\Psi \odot \Psi^*)$$

Beyond POD

What if the snapshots aren't of the state vector?

Taking Gross Pitaevskii as an example, with $\{\rho(t_i)\} \rightarrow P_\rho$

$$\psi = P_\Psi^\dagger \Psi$$

$$q = P_\rho^\dagger \rho = P_\rho^\dagger (\Psi \odot \Psi^*) \approx P_\rho^\dagger (P_\Psi \otimes P_\Psi^*)(\psi \otimes \psi^*)$$

Beyond POD

What if the snapshots aren't of the state vector?

Taking Gross Pitaevskii as an example, with $\{\rho(t_i)\} \to P_\rho$

$$
\begin{aligned}
\psi &= P_\Psi^\dagger \Psi \\
q &= P_\rho^\dagger \rho = P_\rho^\dagger (\Psi \odot \Psi^*) \approx P_\rho^\dagger (P_\Psi \otimes P_\Psi^*)(\psi \otimes \psi^*) \\
\tilde{H}_0 &= P_\Psi^\dagger H_0 P_\Psi
\end{aligned}
$$

Beyond POD

What if the snapshots aren't of the state vector?

Taking Gross Pitaevskii as an example, with $\{\rho(t_i)\} \to P_\rho$

$$
\begin{aligned}
\psi &= P_\Psi^\dagger \Psi \\
q &= P_\rho^\dagger \rho = P_\rho^\dagger (\Psi \odot \Psi^*) \approx P_\rho^\dagger (P_\Psi \otimes P_\Psi^*)(\psi \otimes \psi^*) \\
\tilde{H}_0 &= P_\Psi^\dagger H_0 P_\Psi \\
\frac{d\psi}{dt} &= \tilde{H}_0 \psi + g P_\Psi^\dagger (P_\Psi \otimes P_\Psi^* \otimes P_\Psi)(\psi \otimes \psi^* \otimes \psi)
\end{aligned}
$$

Beyond POD

What if the snapshots aren't of the state vector?

Taking Gross Pitaevskii as an example, with $\{\rho(t_i)\} \to P_\rho$

$$\begin{aligned}
\psi &= P_\Psi^\dagger \Psi \\
q &= P_\rho^\dagger \rho = P_\rho^\dagger (\Psi \odot \Psi^*) \approx P_\rho^\dagger (P_\Psi \otimes P_\Psi^*)(\psi \otimes \psi^*) \\
\tilde{H}_0 &= P_\Psi^\dagger H_0 P_\Psi \\
\frac{d\psi}{dt} &= \tilde{H}_0 \psi + g P_\Psi^\dagger (P_\Psi \otimes P_\Psi^* \otimes P_\Psi)(\psi \otimes \psi^* \otimes \psi) \\
\rho &\approx P_\rho q
\end{aligned}$$

Beyond POD

What if the snapshots aren't of the state vector?

Taking Gross Pitaevskii as an example, with $\{\rho(t_i)\} \to P_\rho$

$$\begin{aligned}
\psi &= P_\Psi^\dagger \Psi \\
q &= P_\rho^\dagger \rho = P_\rho^\dagger (\Psi \odot \Psi^*) \approx P_\rho^\dagger P_\Psi P_\Psi^\dagger (P_\Psi \otimes P_\Psi^*)(\psi \otimes \psi^*) \\
\tilde{H}_0 &= P_\Psi^\dagger H_0 P_\Psi \\
\frac{d\psi}{dt} &= \tilde{H}_0 \psi + g P_\Psi^\dagger (P_\Psi \otimes P_\Psi^* \otimes P_\Psi)(\psi \otimes \psi^* \otimes \psi) \\
\rho &\approx P_\rho q
\end{aligned}$$

Beyond POD

What if the snapshots aren't of the state vector?
Taking Gross Pitaevskii as an example, with $\{\rho(t_i)\} \to P_\rho$

$$\psi = P_\Psi^\dagger \Psi$$

$$q = P_\rho^\dagger \rho = P_\rho^\dagger (\Psi \odot \Psi^*) \approx \underbrace{P_\rho^\dagger P_\Psi}_{B_\rho^\Psi} \underbrace{P_\Psi^\dagger (P_\Psi \otimes P_\Psi^*)}_{\mathbb{T}_\Psi^2} (\psi \otimes \psi^*)$$

$$\tilde{H}_0 = P_\Psi^\dagger H_0 P_\Psi$$

$$\frac{d\psi}{dt} = \tilde{H}_0 \psi + g P_\Psi^\dagger (P_\Psi \otimes P_\Psi^* \otimes P_\Psi)(\psi \otimes \psi^* \otimes \psi)$$

$$\rho \approx P_\rho q$$

Beyond POD

What if the snapshots aren't of the state vector?
Taking Gross Pitaevskii as an example, with $\{\rho(t_i)\} \to P_\rho$

$$\psi(t_0) = \underline{\psi_0}$$
$$\frac{d\psi}{dt} = \underline{\tilde{H}_0}\psi + g\underline{\mathbb{T}^3_\Psi}(\psi \otimes \psi^* \otimes \psi)$$
$$q = P_\rho^\dagger \rho \approx \underline{B_\rho^\Psi \mathbb{T}^2_\Psi}(\psi \otimes \psi^*)$$
$$\rho \approx P_\rho q$$

Multiple snapshot sets:

Multiple snapshot sets:

Navier-Stokes as an example has three velocity fields

$$\mathbf{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$$

Multiple snapshot sets:

Navier-Stokes as an example has three velocity fields

$$\mathbf{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$$

and the advective term

$$(\mathbf{u}\cdot\nabla)\mathbf{u} = \begin{bmatrix} u_x \odot (\partial_x u_x) + u_y \odot (\partial_y u_x) + u_z \odot (\partial_z u_x) \\ u_x \odot (\partial_x u_y) + u_y \odot (\partial_y u_y) + u_z \odot (\partial_z u_y) \\ u_x \odot (\partial_x u_z) + u_y \odot (\partial_y u_z) + u_z \odot (\partial_z u_z) \end{bmatrix} \stackrel{i}{=} u_j \odot (\partial_j u_i)$$

Multiple snapshot sets:
Navier-Stokes as an example has three velocity fields

$$\mathbf{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$$

and the advective term

$$(\mathbf{u}\cdot\nabla)\mathbf{u} = \begin{bmatrix} u_x \odot (\partial_x u_x) + u_y \odot (\partial_y u_x) + u_z \odot (\partial_z u_x) \\ u_x \odot (\partial_x u_y) + u_y \odot (\partial_y u_y) + u_z \odot (\partial_z u_y) \\ u_x \odot (\partial_x u_z) + u_y \odot (\partial_y u_z) + u_z \odot (\partial_z u_z) \end{bmatrix} \stackrel{i}{=} u_j \odot (\partial_j u_i)$$

$$\xrightarrow{i} \underline{B_j^{i\dagger}}\, \underline{\mathbb{T}_j^2} \left(v_j \otimes \underline{D_j B_j^i} v_i \right)$$

Equation-Free Methods

Equation-Free Methods

- **D**ynamic **M**ode **D**ecomposition

Equation-Free Methods

- **D**ynamic **M**ode **D**ecomposition
- **S**parse **I**dentification of **N**onlinear **Dy**namics

Equation-Free Methods

- **D**ynamic **M**ode **D**ecomposition
- **S**parse **I**dentification of **N**onlinear **Dy**namics
- Neural Operators

Equation-Free Methods

- **D**ynamic **M**ode **D**ecomposition
- **S**parse **I**dentification of **N**onlinear **Dy**namics
- Neural Operators

Equation-Full Methods

Equation-Free Methods

- **D**ynamic **M**ode **D**ecomposition
- **S**parse **I**dentification of **N**onlinear **Dy**namics
- Neural Operators

Equation-Full Methods

- **P**roper **O**rthogonal **D**ecomposition

Equation-Free Methods

- **D**ynamic **M**ode **D**ecomposition
- **S**parse **I**dentification of **N**onlinear **Dy**namics
- Neural Operators

Equation-Full Methods

- **P**roper **O**rthogonal **D**ecomposition
- Eigenvector Continuation

Parametric Matrix Models

Equation-Free Methods

- **D**ynamic **M**ode **D**ecomposition
- **S**parse **I**dentification of **N**onlinear **Dy**namics
- Neural Operators

Equation-Full Methods

- **P**roper **O**rthogonal **D**ecomposition
- Eigenvector Continuation

Parametric Matrix Models

Equation-Free Methods

- **D**ynamic **M**ode **D**ecomposition
- **S**parse **I**dentification of **N**onlinear **Dy**namics
- Neural Operators

Equation-Informed Methods

Equation-Full Methods

- **P**roper **O**rthogonal **D**ecomposition
- Eigenvector Continuation

Parametric Matrix Models

Equation-Free Methods

- **D**ynamic **M**ode **D**ecomposition
- **S**parse **I**dentification of **N**onlinear **Dy**namics
- Neural Operators

Equation-Informed Methods

Form-Full Methods

Equation-Full Methods

- **P**roper **O**rthogonal **D**ecomposition
- Eigenvector Continuation

Summary

Summary

- Practical high-level overview to making a PMM

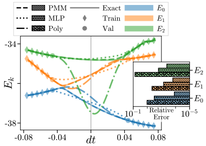
$$H(c) = H_0 + cH_1$$
$$\downarrow$$
$$M(c) = \underline{M_0} + c\underline{M_1}$$

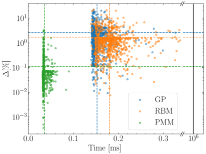
Summary

- Practical high-level overview to making a PMM

$$H(c) = H_0 + cH_1$$
$$\downarrow$$
$$M(c) = \underline{M_0} + c\underline{M_1}$$

- Highlighted previous results in emulation

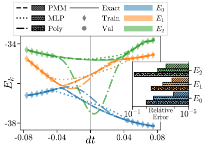


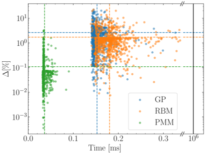


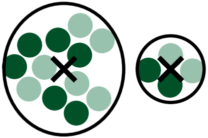
Summary

- Practical high-level overview to making a PMM
- Highlighted previous results in emulation
- Teased ongoing work in model discovery

$$H(c) = H_0 + cH_1$$
$$\downarrow$$
$$M(c) = \underline{M_0} + c\underline{M_1}$$





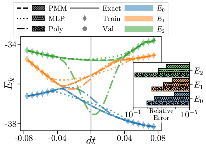


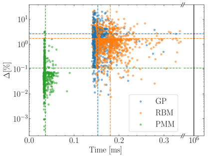
Summary

- Practical high-level overview to making a PMM

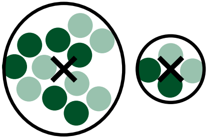
$$H(c) = H_0 + cH_1$$
$$\downarrow$$
$$M(c) = \underline{M_0} + c\underline{M_1}$$

- Highlighted previous results in emulation





- Teased ongoing work in model discovery



- Detailed new theory for extending PMMs

$$\underline{\mathbb{T}^3_\Psi} \quad \underline{T_{jkl}} \quad \underline{B^\Psi_\rho} \quad \cdots$$

References

Cohen, Gregory, Saeed Afshar, et al. (2017). arXiv: 1702.05373 [cs.CV].

Cook, Patrick, Danny Jammooa, et al. (2024). https://arxiv.org/abs/2401.11694. arXiv: 2401.11694v5 [cs.LG].

De, Arinjoy, Patrick Cook, et al. (2023). arXiv: 2309.10856 [quant-ph].

Fein-Ashley, Jacob, Tian Ye, et al. (2024). arXiv: 2402.00564v1 [cs.CV].

Frame, Dillon, Rongzheng He, et al. (2018). In: *Phys. Rev. Lett.* 121 (3), p. 032501. DOI: 10.1103/PhysRevLett.121.032501.

Gamito, J., J. Khalouf-Rivera, et al. (2022). In: *Physical Review E* 106.4. DOI: 10.1103/physreve.106.044125.

Granmo, Ole-Christoffer, Sondre Glimsdal, et al. (2019). arXiv: 1905.09688 [cs.LG].

Jin, Lauren, Ante Ravlić, et al. (2024). In: *Bulletin of the American Physical Society*.

Jin, Yingyezhe, Wenrui Zhang, et al. (2018). In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Curran Associates Inc., pp. 7005–7015. DOI: 10.5555/3327757.3327804.

Kabir, H. M. D., Moloud Abdar, et al. (2020). In: *IEEE Transactions on Artificial Intelligence* 4, pp. 1165–1177.

König, S., A. Ekström, et al. (2020). In: *Physics Letters B* 810, p. 135814. DOI: https://doi.org/10.1016/j.physletb.2020.135814.

Kramer, Boris and Karen E. Willcox (2019). In: *AIAA Journal* 57.6, pp. 2297–2307. DOI: 10.2514/1.j057791.

Lecun, Y., L. Bottou, et al. (1998). In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. DOI: 10.1109/5.726791.

Mazzia, Vittorio, Francesco Salvetti, et al. (2021). In: *Scientific Reports* 11.1. DOI: 10.1038/s41598-021-93977-0.

Nøkland, Arild and Lars Hiller Eidnes (2019). arXiv: 1901.06656 [stat.ML].

Pishchik, Evgenii (2023). DOI: 10.20944/preprints202301.0463.v1.

Reed, Brendan T., Rahul Somasundaram, et al. (2024). arXiv: 2405.20558 [astro-ph.HE].

Sarkar, Avik and Dean Lee (2021). In: *Phys. Rev. Lett.* 126 (3), p. 032501. DOI: 10.1103/PhysRevLett.126.032501.

Somasundaram, Rahul, Cassandra L. Armstrong, et al. (2024). arXiv: 2404.11566 [nucl-th].

Somasundaram, Rahul, Isak Svensson, et al. (2024). arXiv: 2410.00247 [nucl-th].

Tanveer, M., M. Karim Khan, et al. (2021). In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE Computer Society, pp. 4789–4796. DOI: 10.1109/ICPR48806.2021.9412221.

Xiao, Han, Kashif Rasul, et al. (2017). arXiv: 1708.07747 [cs.LG].

General Machine Learning

General Machine Learning

Underlying Equations

???

General Machine Learning

Underlying Equations

???

PMM

$$H(\vec{c}) \quad = \quad \underline{H_0} + c_1\underline{H_1} + \cdots + c_p\underline{H_p}$$

General Machine Learning

Underlying Equations

???

PMM

$$
\begin{aligned}
H(\vec{c}) &= \underline{H_0} + c_1 \underline{H_1} + \cdots + c_p \underline{H_p} \\
H(\vec{c})\,|E_k(\vec{c})\rangle &= E_k(\vec{c})\,|E_k(\vec{c})\rangle
\end{aligned}
$$

General Machine Learning

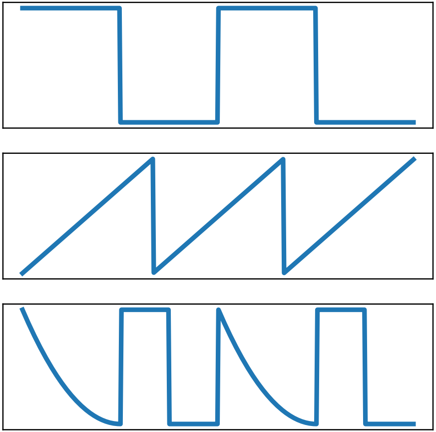
Underlying Equations

???

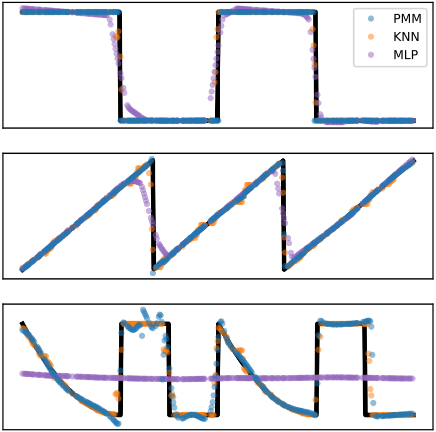
PMM

$$\begin{aligned} H(\vec{c}) &= \underline{H_0} + c_1 \underline{H_1} + \cdots + c_p \underline{H_p} \\ H(\vec{c}) \left|E_k(\vec{c})\right\rangle &= E_k(\vec{c}) \left|E_k(\vec{c})\right\rangle \\ z_i(\vec{c}) &\sim \left|\left\langle E_a(\vec{c}) \middle| \underline{O_i} \middle| E_b(\vec{c}) \right\rangle\right|^2 \end{aligned}$$

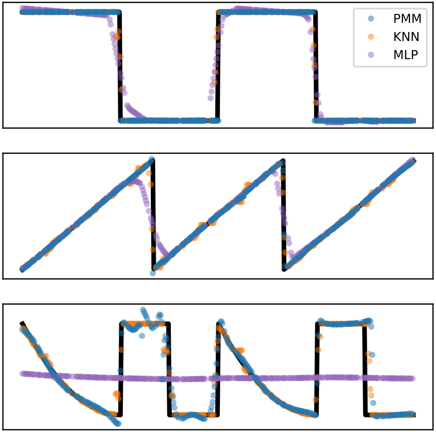
Discontinuous Regression

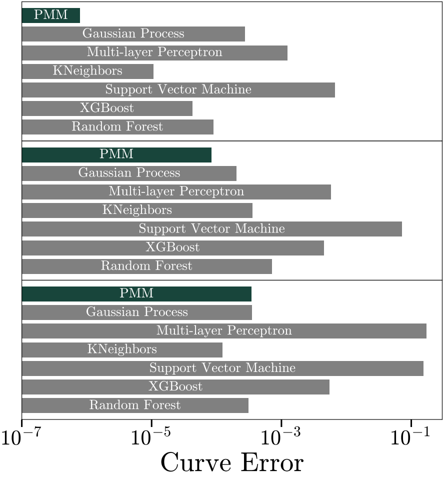


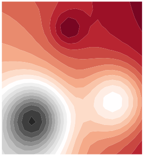
Discontinuous Regression

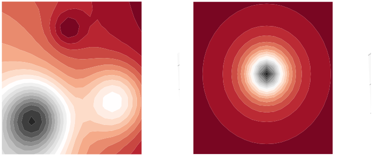


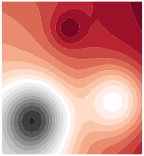
Discontinuous Regression

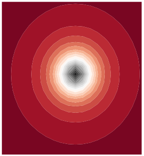




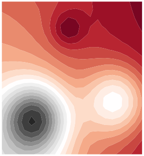


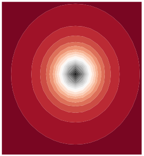




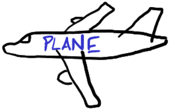


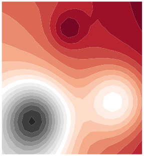
$$\sum_{nm} a_{nm} \sin{(nx)} \sin{(my)}$$

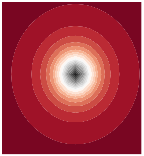




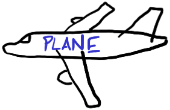
$$\sum_{nm} a_{nm} \sin(nx) \sin(my)$$

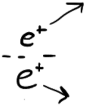


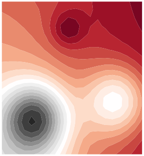


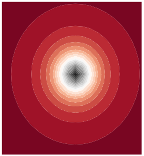


$$\sum_{nm} a_{nm} \sin(nx) \sin(my)$$

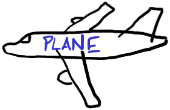


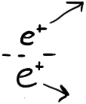






$$\sum_{nm} a_{nm} \sin(nx) \sin(my)$$





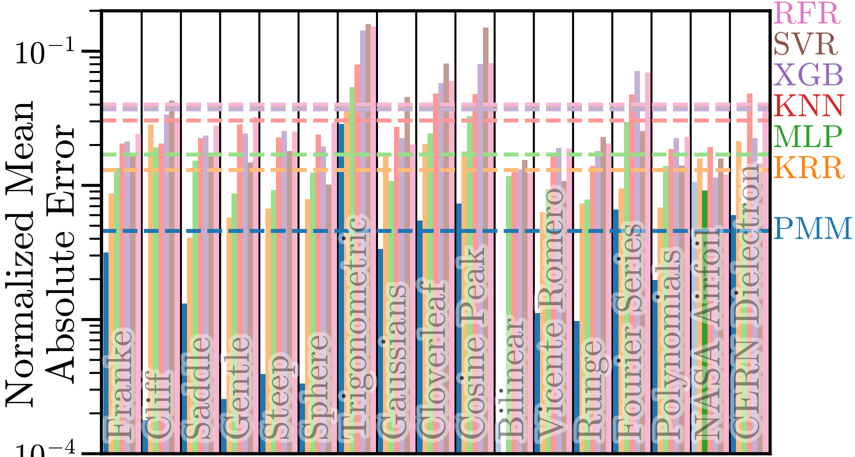
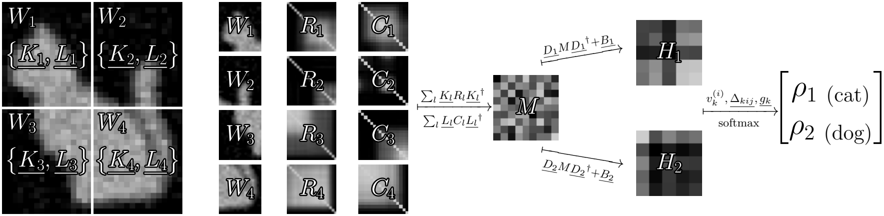
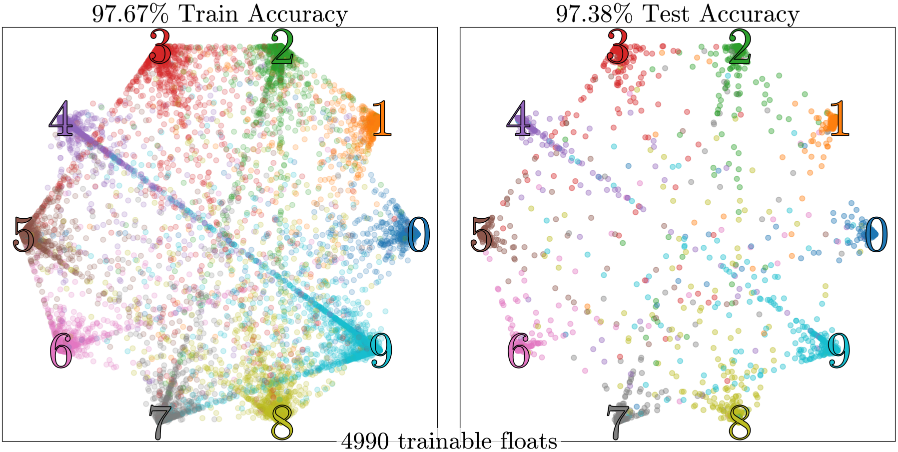


Image Classification

Image Classification

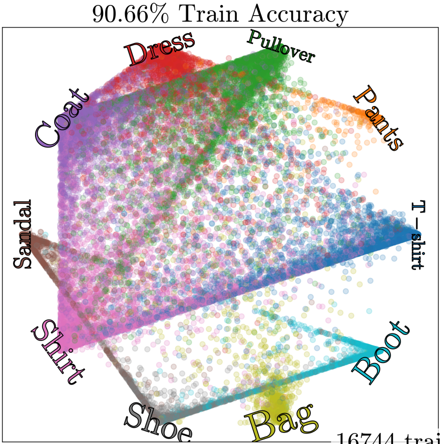


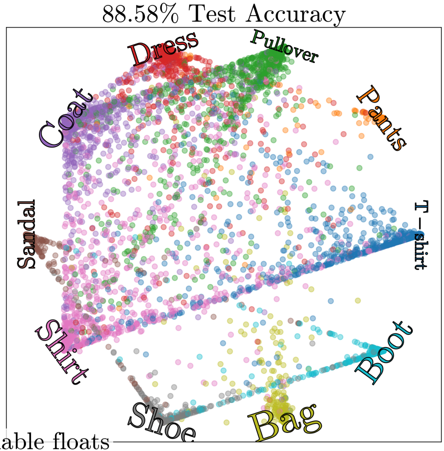

97.67% Train Accuracy
97.38% Test Accuracy
3
2
1
4
0
5
6
9
7
8
4990 trainable floats

MNIST Digits

Model	**Acc. (%)**	`float`
PMM†	97.38	4990
ConvPMM	99.10	129 416
DNN-2† [Pishchik 2023]	96.5	$\sim$ 311 650
DNN-3† [Pishchik 2023]	97.0	$\sim$ 386 718
DNN-5† [Pishchik 2023]	97.2	$\sim$ 575 050
GECCO [Fein-Ashley, Ye, et al. 2024]	98.04	$\sim$ 19 000
CTM-250 [Granmo, Glimsdal, et al. 2019]	98.82	31 750
CTM-8000 [Granmo, Glimsdal, et al. 2019]	99.4	527 250
Eff.-CapsNet [Mazzia, Salvetti, et al. 2021]	99.84	161 824

† non-convolutional



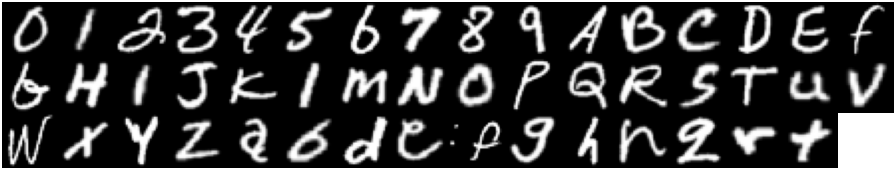


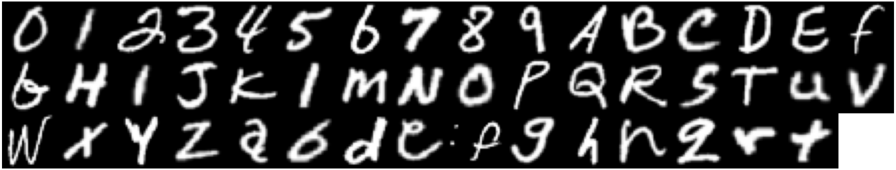
16744 trainable floats

Fashion MNIST

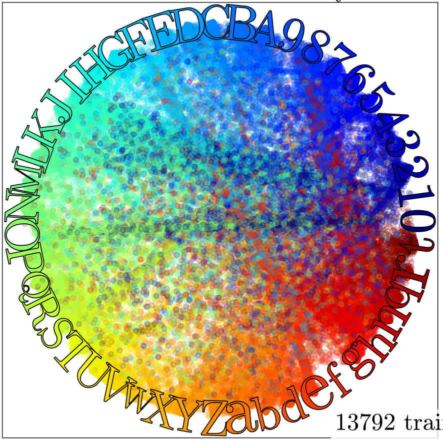
Model	**Acc. (%)**	`float`
PMM†	88.58	16 744
ConvPMM	90.94	278 280
GECCO [Fein-Ashley, Ye, et al. 2024]	88.09	$\sim$ 19 000
CTM-250 [Granmo, Glimsdal, et al. 2019]	88.25	31 750
CTM-8000 [Granmo, Glimsdal, et al. 2019]	91.5	527 250
MLP† [Nøkland and Eidnes 2019]	91.63	2 913 290
VGG8B [Nøkland and Eidnes 2019]	95.47	$\sim$ 7 300 000
F-T DARTS [Tanveer, Khan, et al. 2021]	96.91	$\sim$ 3 200 000

† non-convolutional

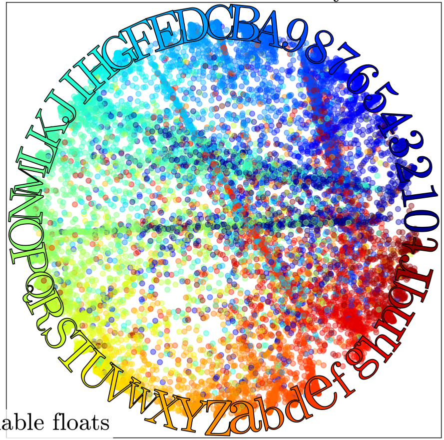




83.45% Train Accuracy



81.57% Test Accuracy

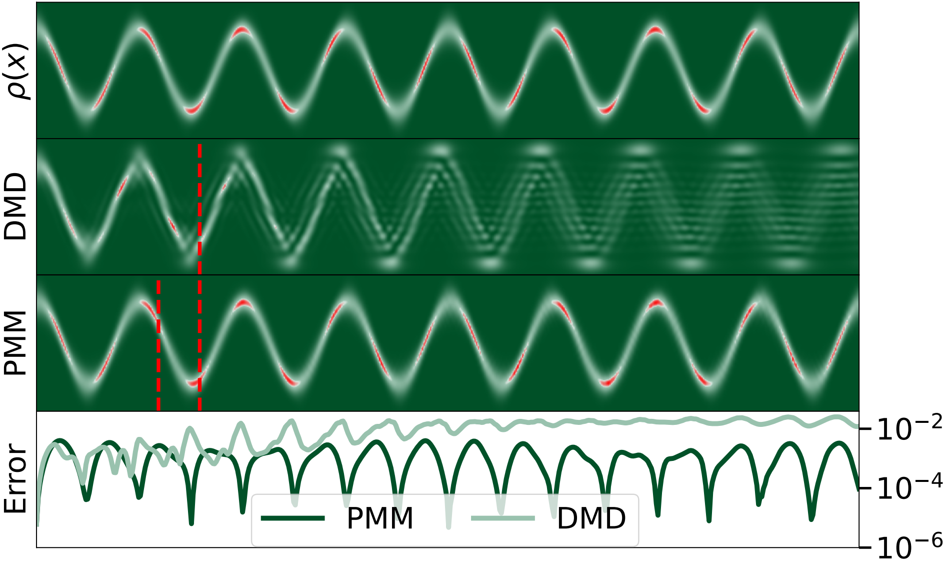


13792 trainable floats

EMNIST

Model	**Acc. (%)**	`float`
PMM†	81.57	13 792
ConvPMM	85.95	349 172
CNN [Kabir, Abdar, et al. 2020]	79.61	21 840
CNN (S-FC) [Kabir, Abdar, et al. 2020]	82.77	13 820
CNN (S-FC) [Kabir, Abdar, et al. 2020]	83.21	16 050
HM2-BP† [Y. Jin, Zhang, et al. 2018]	85.57	665 647

† non-convolutional



$\rho(x)$
DMD
PMM
Error
PMM
DMD
10^{-2}
10^{-4}
10^{-6}