

# Variational principles & efficient subspace emulation

---

Jordan Melendez<sup>1, 2</sup>

December 13, 2021

<sup>1</sup>Root Insurance, Data Scientist

<sup>2</sup>The Ohio State University



THE OHIO STATE UNIVERSITY

There is **code** to accompany these slides!

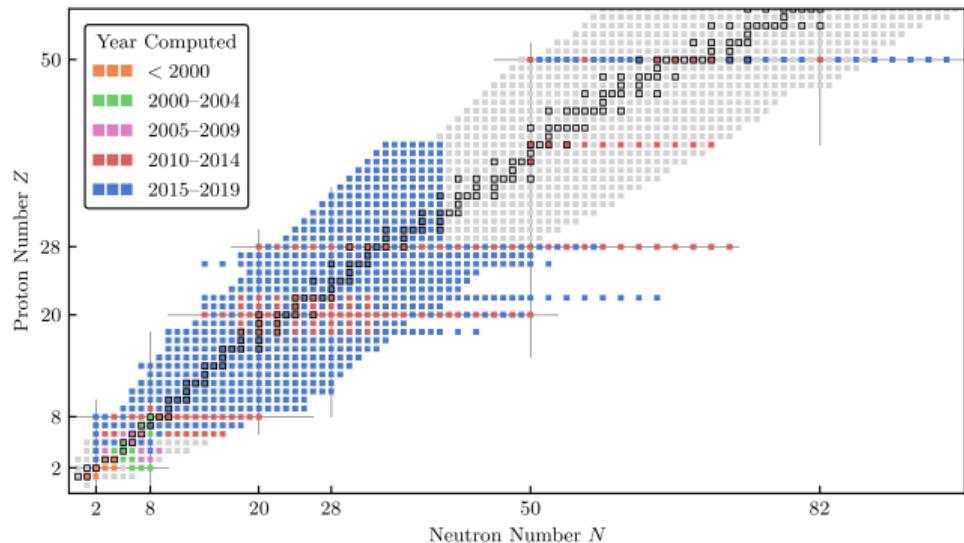
[github.com/jordan-melendez/quantum-emulator-examples](https://github.com/jordan-melendez/quantum-emulator-examples)

## Background

---

# Progress in heavy nuclei

- Great progress has been made
- Growth in computing power and algorithms is pushing to heavier systems
- But statistics requires more than one prediction



# Why we need emulators

Complexity:

1. 🤔 Forward UQ
2. 😰 Inverse UQ
3. 😱 Experimental Design

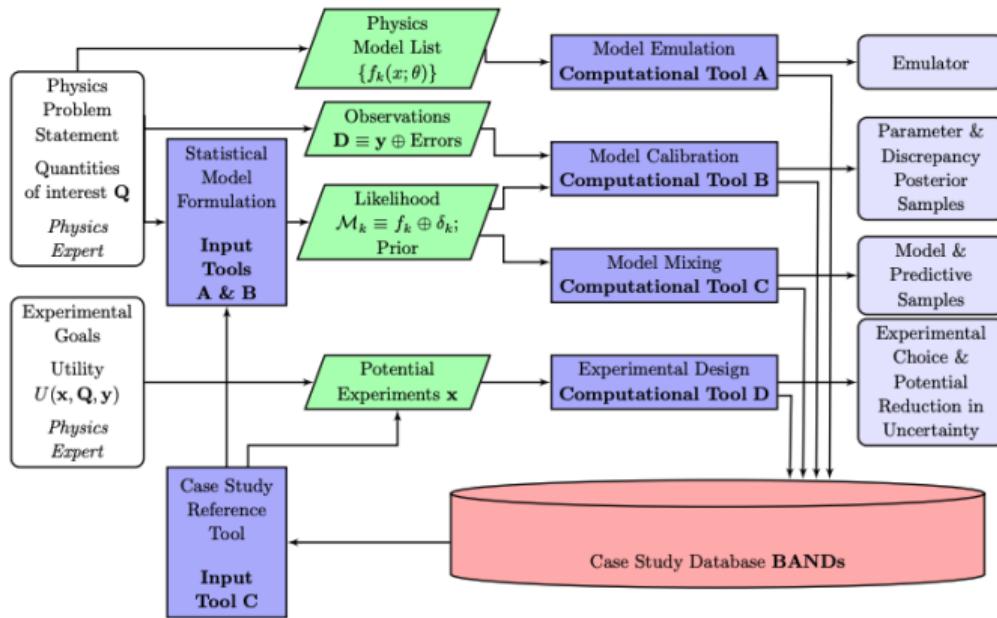
- $r$ -process,  $0\nu\beta\beta$  simulations, & optimizing FRIB experiments can each be compute intensive — but important!
- Emulator: An algorithm capable of accurately approximating the exact solution while requiring only a fraction of the computational resources

# Relation to BAND

## BAND

The goal of BAND is to translate novel statistical methods of UQ into software tools that address prominent current problems in nuclear physics.

- Subspace emulation could play a key role in **Tool A**.
- An emulator can feed into all subsequent tools



## Emulators

---

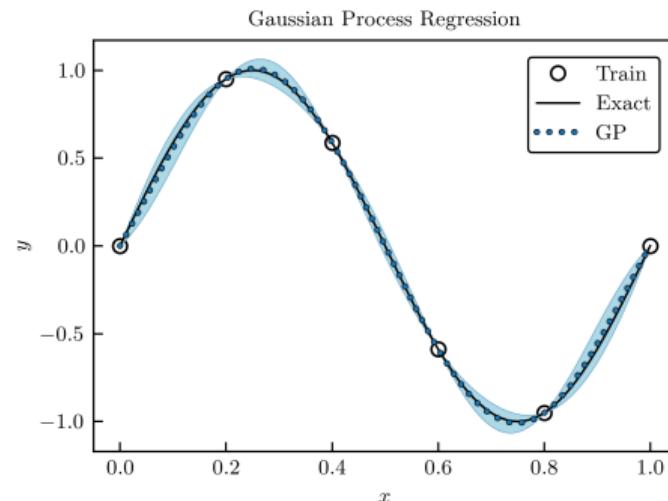
# Gaussian processes as emulators

```
import numpy as np
from sklearn.gaussian_process \
    import GaussianProcessRegressor
from sklearn.gaussian_process.kernels \
    import RBF, ConstantKernel as C

x_train = np.arange(0, 1.1, 0.2)
x_valid = np.linspace(0, 1, 101)
y_train = np.sin(2 * np.pi * x_train)
y_valid = np.sin(2 * np.pi * x_valid)

kernel = C(1) * RBF(length_scale=0.2)
gp = GaussianProcessRegressor(kernel)
gp.fit(x_train[:, None], y_train)
y_pred, y_stdv = gp.predict(
    x_valid[:, None], return_std=True)
```

- Forward Problem: Train at “kinematic” points
- Inverse Problem: + train at parameter values
- Modern libraries make it easy



## Gaussian processes: pros & cons

### Pros

- Non-intrusive (can leave legacy code untouched)
- Easy to use
- Flexible (non-parametric)
- Error bands for free
- ...

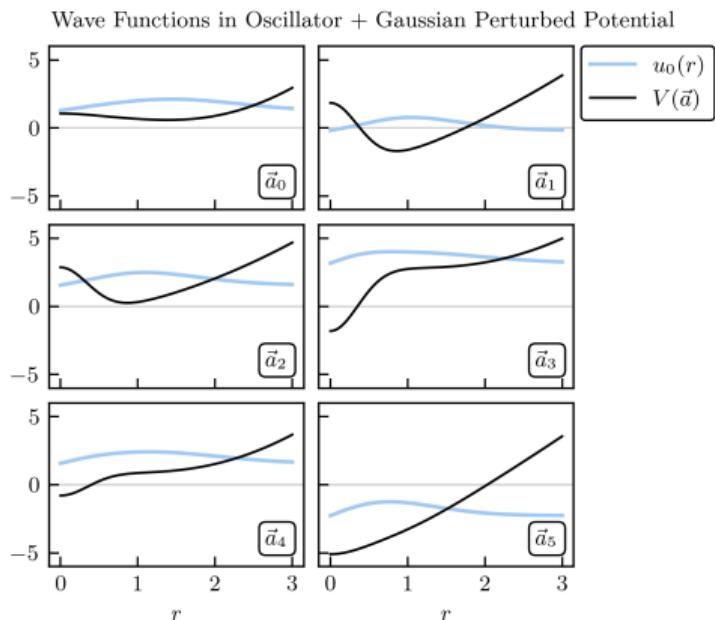
### Cons

- Not great at extrapolating
- Choosing a kernel – not always straightforward
- Numerical instabilities can arise
- Parameter space can be large!
- Does not necessarily take advantage of structure of the system (more on this later...)

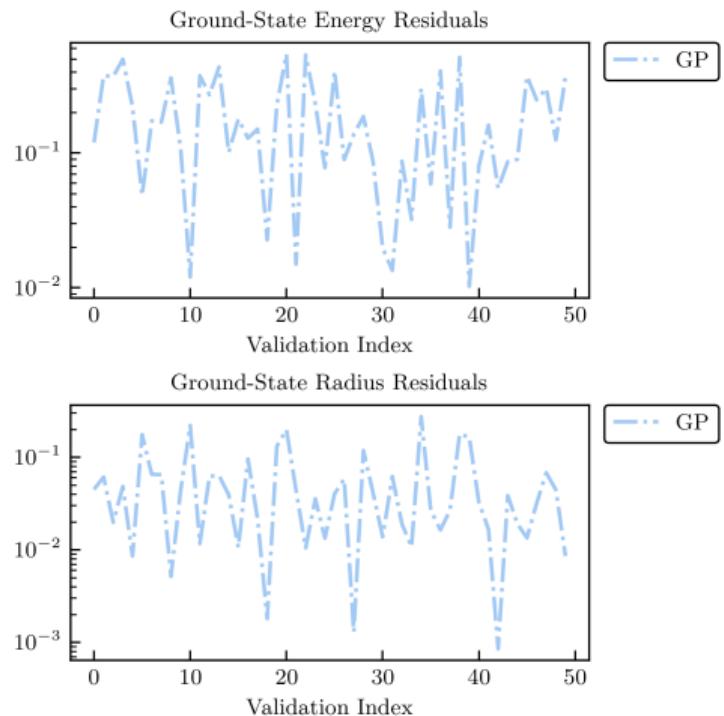
# Example: GPs for the Schrödinger equation

$$V(\vec{a}) = V_{h.o.}(\omega) + \sum a_i \exp[-(r/b_i)^2]$$

for fixed  $\{b_i\} = \{0.5, 2, 4\}$ .



Fit separate GP to  $E$  and radius  $R = \mathbb{E}[r]$ .



## Ritz subspace method: the basics

Instead of solving the Schrödinger eq. (an eigenvalue problem for bound states)

$$H(\vec{a}) |\psi(\vec{a})\rangle = E(\vec{a}) |\psi(\vec{a})\rangle$$

write down a **trial wave function** as a linear combination

$$|\tilde{\psi}\rangle = \sum_i \beta_i |\psi_i\rangle$$

and use a **variational method** to determine the best  $\beta_i$ :

### Bound state variational method

Minimize  $\langle \tilde{\psi} | H(\vec{a}) | \tilde{\psi} \rangle$  such that  $\langle \tilde{\psi} | \tilde{\psi} \rangle = 1$ .

The problem has then been reduced from determining an infinite-dimensional (or, at least large)  $|\psi(\vec{a})\rangle$  to determining a couple coefficients  $\beta_i$ .

# Ritz subspace method for eigenvalue problems (arXiv:2104.04441)

Problem:  $H(\vec{a})$  is  $N \times N$  and  $N \gg 1$

$$H(\vec{a}) |\psi(\vec{a})\rangle = E(\vec{a}) |\psi(\vec{a})\rangle \quad (17)$$

such that  $H(\vec{a}) = H_0 + \sum a_j H_j$

Solution: Choose a basis with  $N_b \ll N$ :

$$X \equiv \begin{pmatrix} | & | & & | \\ |\psi_1\rangle & |\psi_2\rangle & \dots & |\psi_{N_b}\rangle \\ | & | & & | \end{pmatrix} \quad (18)$$

$$\tilde{H}(\vec{a}) = X^\dagger H(\vec{a}) X, \quad \mathcal{N} = X^\dagger X \quad (19)$$

Finally, solve smaller problem:

$$\tilde{H}(\vec{a}) \beta(\vec{a}) = \tilde{E}(\vec{a}) \mathcal{N} \beta(\vec{a}) \quad (20)$$

```
def setup_projections(self, X):
    # Project matrices once
    H0_sub = X.T @ self.H0 @ X # const.
    H1_sub = X.T @ self.H1 @ X # linear
    # Store for later
    self.X = X; self.N = X.T @ X
    self.H0_sub = H0_sub
    self.H1_sub = H1_sub
    return self

def solve_subspace(self, a):
    H = self.H0_sub + self.H1_sub @ a
    from scipy.linalg import eigh
    E, beta = eigh(H, self.N)
    return E[0], beta[:, 0] # g.s.
```

# Ritz subspace method for eigenvalue problems (arXiv:2104.04441)

What is gained?

$$H(\vec{a}) |\psi(\vec{a})\rangle = E(\vec{a}) |\psi(\vec{a})\rangle \quad (17)$$

v.s.

$$\tilde{H}(\vec{a}) \beta(\vec{a}) = \tilde{E}(\vec{a}) N \beta(\vec{a}) \quad (20)$$

Eq. (20) is much smaller, and

$$E(\vec{a}) \approx \tilde{E}(\vec{a}) \quad |\psi(\vec{a})\rangle \approx X \beta(\vec{a})$$

Emulator for both  $E(\vec{a})$  and  $|\psi(\vec{a})\rangle$ ! Thus:

$$\begin{aligned} \langle \hat{O}(\vec{a}) \rangle &= \langle \psi(\vec{a}) | \hat{O}(\vec{a}) | \psi(\vec{a}) \rangle \\ &\approx \beta(\vec{a})^\dagger [X^\dagger \hat{O}(\vec{a}) X] \beta(\vec{a}) \end{aligned} \quad (21)$$

```
def solve_subspace(self, a):
    H = self.H0_sub + self.H1_sub @ a
    from scipy.linalg import eigh
    E, beta = eigh(H, self.N)
    # Get ground states:
    return E[0], beta[:, 0]

def predict(self, a):
    E, beta = self.solve_subspace(a)
    psi = self.X @ beta
    return E, psi

def expectation_value(self, a):
    op = self.op0_sub + self.op1_sub @ a
    E, beta = self.solve_subspace(a)
    return beta.T @ op @ beta
```

## Subspace methods: pros & cons

### Pros

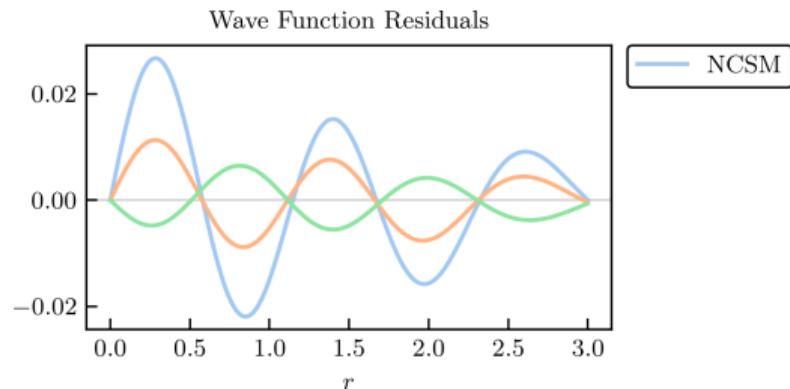
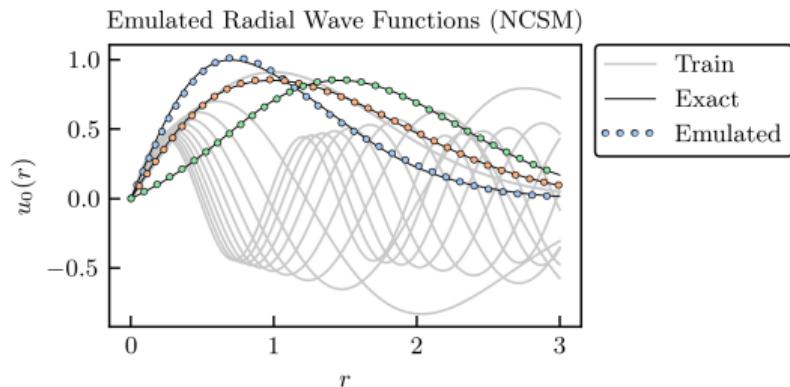
- Can radically reduce the size of the eigenvector problem
- If the basis is well chosen, one can get very accurate results
- Can emulate both the energy and the wave function
- Gets emulator for downstream observables  $\langle \hat{O}(\vec{a}) \rangle$  for free

### Cons

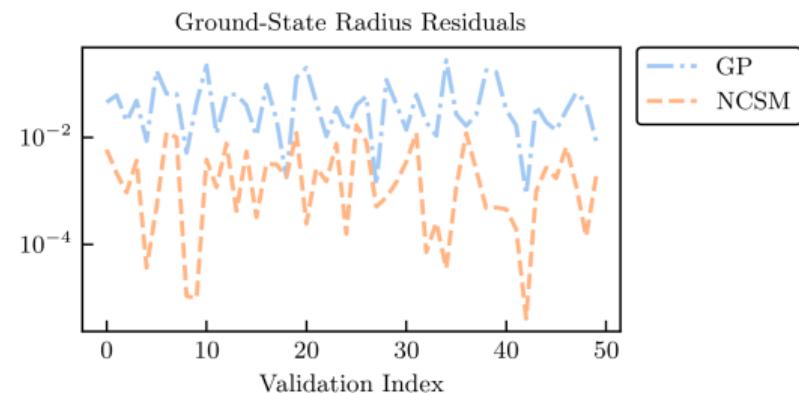
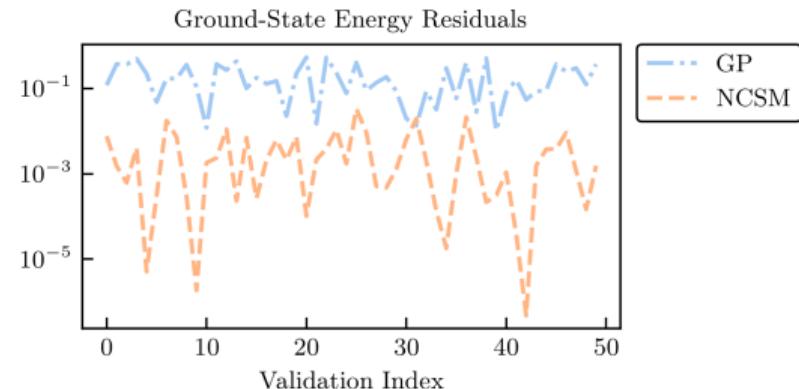
- Intrusive (Requires writing **new** solver code, but it fits in these slides!)
- Not clear how to choose the basis  $\{ |\psi_i\rangle\}$ . (**Will fix right now.**)
- No free uncertainty quantification
- Emulating excited states will require enlarging basis

# A comparison: GPs vs Ritz (NCSM) subspace emulators

Use 6 lowest oscillator states as basis.



Fit a separate GP to  $E$  and the radius  $R$ .



Efficient Subspace Emulators  
aka Eigenvector Continuation  
aka Reduced Basis Method

---

# Efficient subspace emulators for bound states: the basics

Remember, write down a **trial wave function**:  $|\tilde{\psi}\rangle = \sum \beta_i |\psi_i\rangle$

## Bound state variational method

Minimize  $\langle \tilde{\psi} | H(\vec{a}) | \tilde{\psi} \rangle$  such that  $\langle \tilde{\psi} | \tilde{\psi} \rangle = 1$ .

But how to choose the basis  $\{ |\psi_i\rangle \}$ ? For parameter-dependent problems:

## Efficient Subspace Emulation

The insight: Use exact solutions  $|\psi(\vec{a}_i)\rangle$  at a set of training parameters  $\{\vec{a}_i\}$  as the basis for the variational calculation.

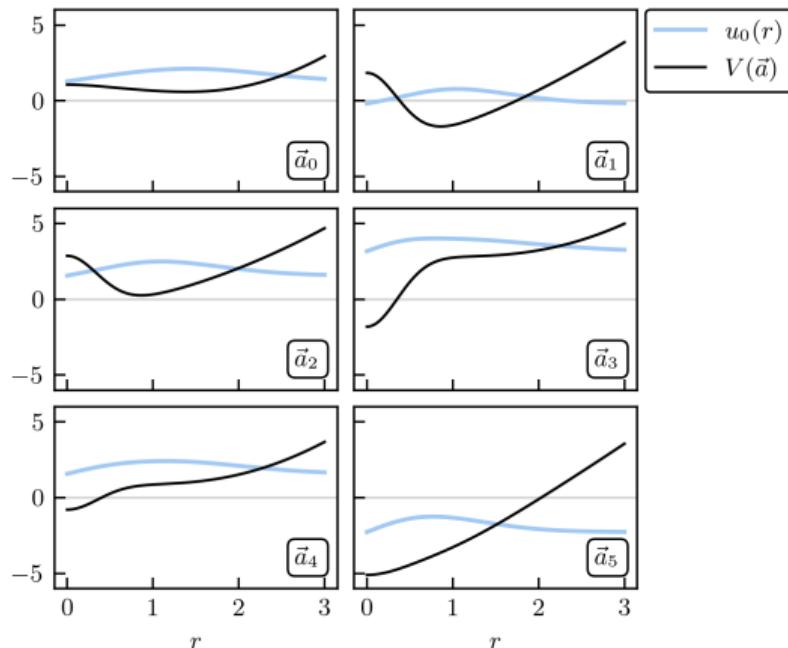
The intuition: As the  $\vec{a}$  are varied, the eigenvectors only trace a small subspace compared to the full Hilbert space. Using exact solutions thus automatically finds an **incredibly** effective basis for subsequent emulation.

# Efficient subspace emulators for bound states: the code

$$V(\vec{a}) = V_{h.o.}(\omega) + \sum a_i \exp[-(r/b_i)^2]$$

for fixed  $\{b_i\} = \{0.5, 2, 4\}$ .

Wave Functions in Oscillator + Gaussian Perturbed Potential



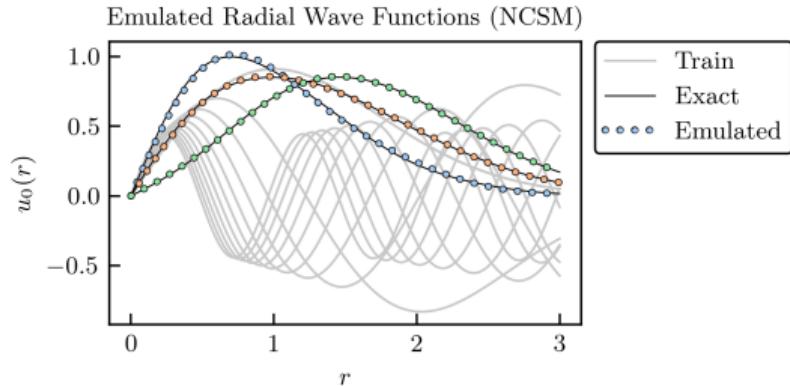
```
import numpy as np

def fit(self, a_train):
    # Create subspace from exact |ψ(̄a)⟩
    X = []
    for a in a_train:
        E, psi = self.solve_exact(a)
        X.append(psi)
    # Stack them as columns
    X = np.stack(X, axis=1)

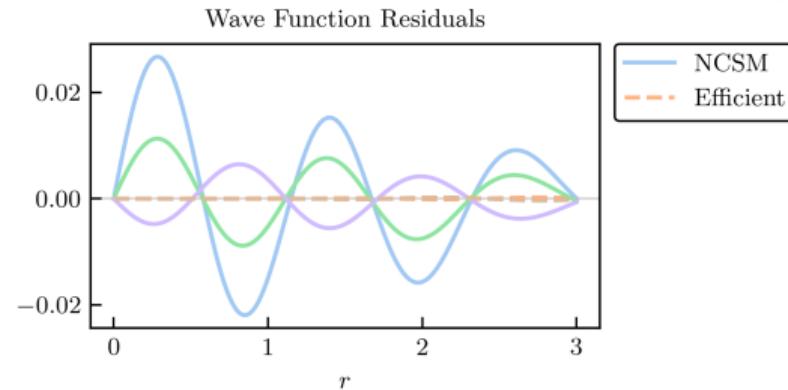
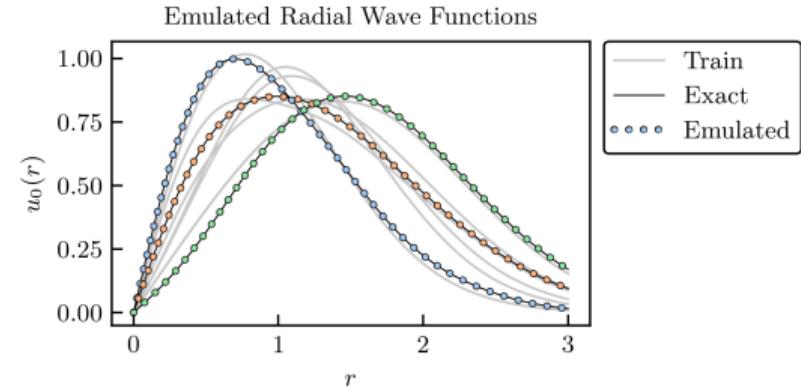
    # The same function as before:
    self.setup_projections(X)
    # Store the training points
    self.a_train = a_train
    return self
```

## Another comparison: GPs vs NCSM vs efficient emulators

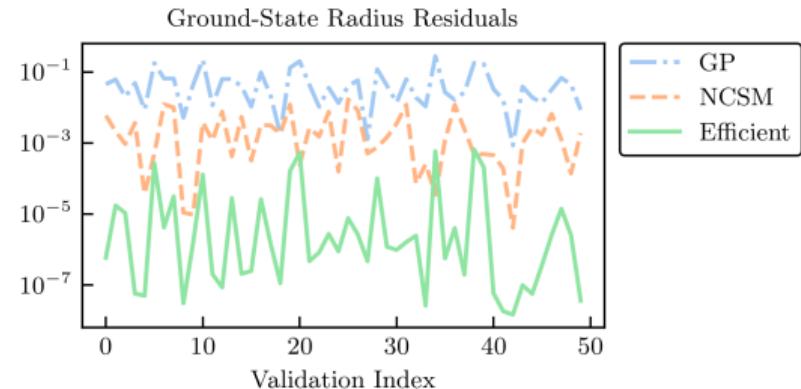
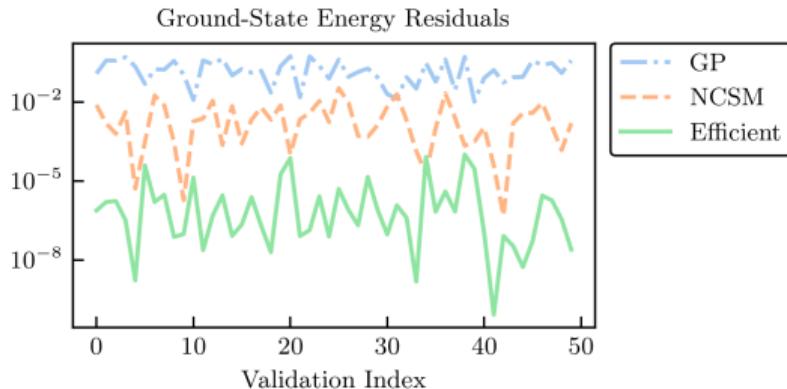
Inefficient! Most training wave functions do not look like the emulated states



Much smaller basis span, but much more efficient!

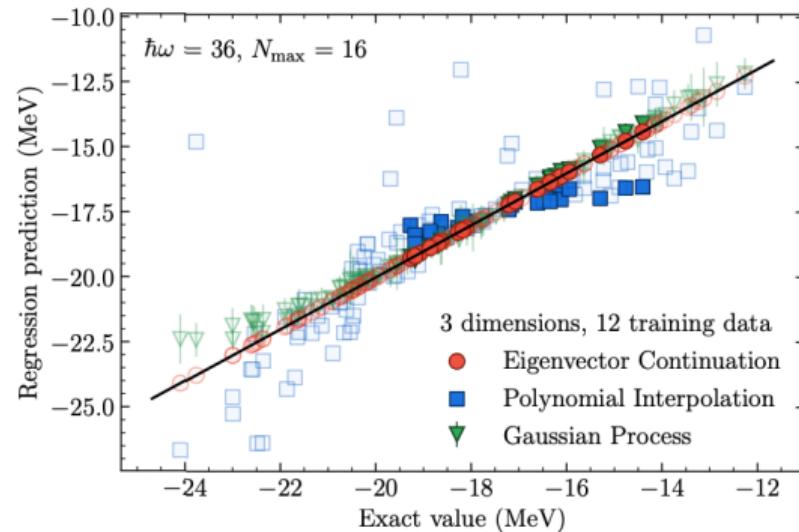
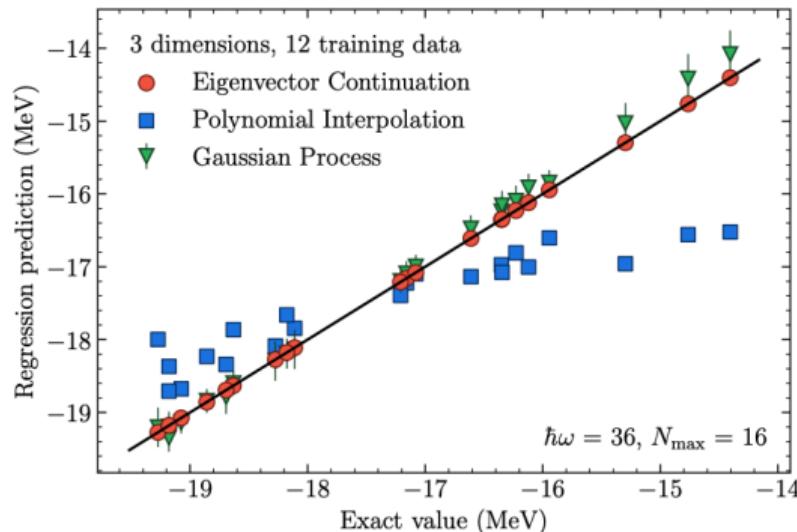


## Another comparison: GPs vs NCSM vs efficient emulators



Comparing GP, NCSM, and the efficient emulators: the efficient emulator wins!

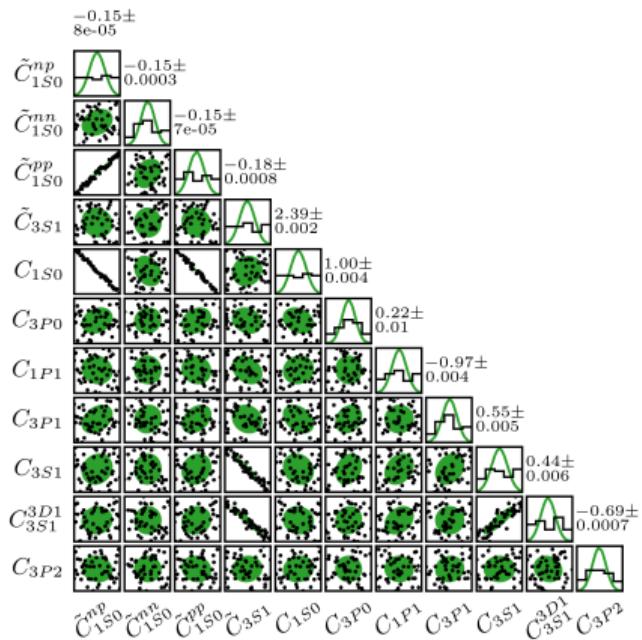
# Bound state emulators in the wild (König *et al.* arXiv:1909.08446)



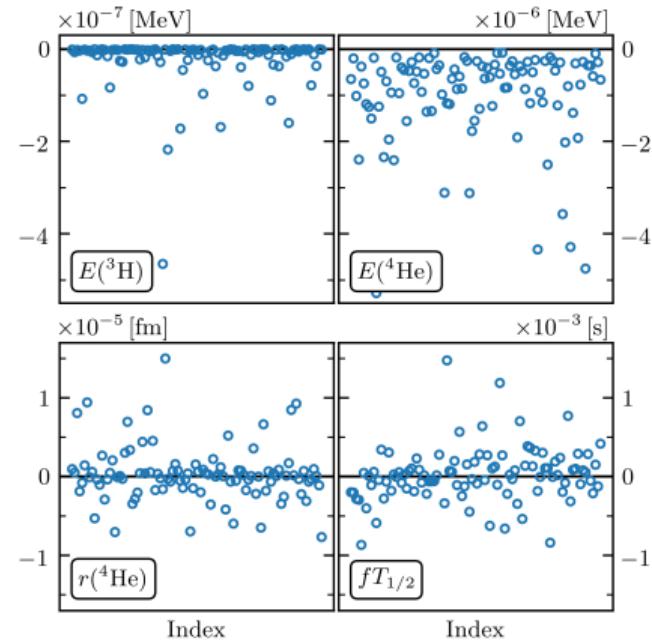
Eigenvector continuation (the efficient subspace emulator) beats both polynomials and GPs on interpolation and extrapolation in parameter space.

Observable:  ${}^4\text{He}$  g.s. energy. Uses 12 training points with 3 parameters.

# Bound state emulators in the wild: inverse problem ([arXiv:2104.04441](https://arxiv.org/abs/2104.04441))



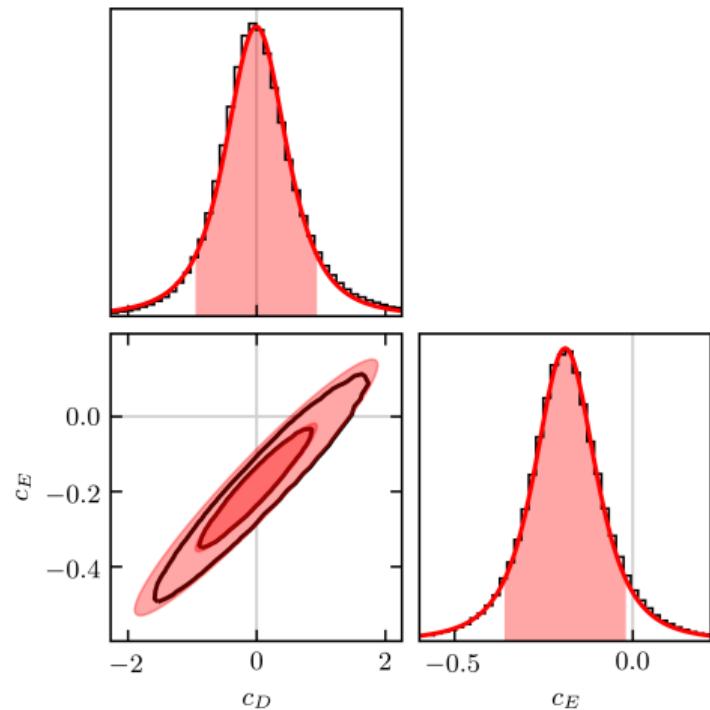
Sampling 15 different parameters (some not shown). 50 training points.



Negligible residuals on validation data

# Bound state emulators in the wild: inverse problem ([arXiv:2104.04441](https://arxiv.org/abs/2104.04441))

Able to rapidly perform sampling on laptop in minutes, rather than on supercomputer for hours



# Bound state emulators in the wild: uncertainties (Sarkar, Lee: arXiv:2107.13449)

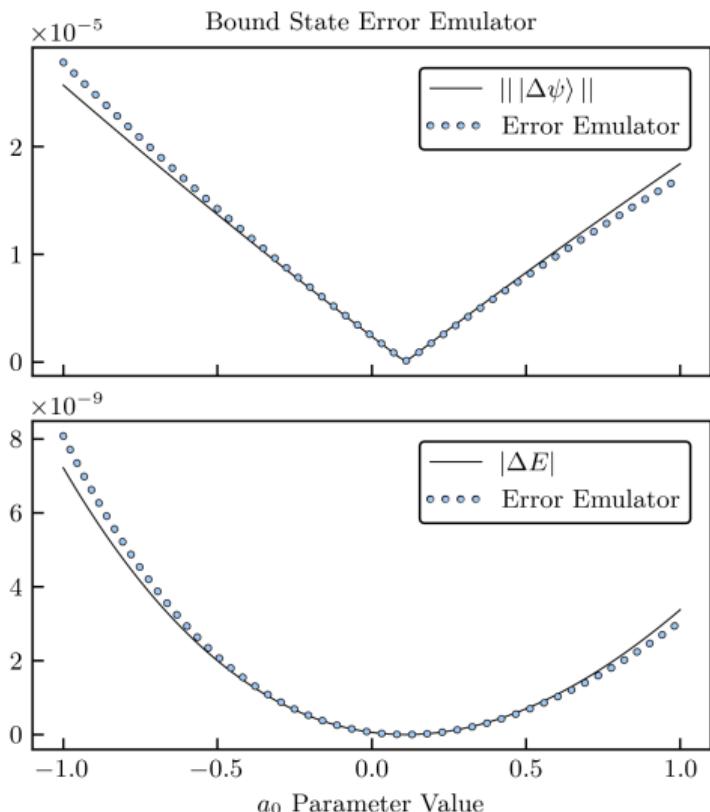
Uncertainty in the emulator:

$$\text{Error}[|\tilde{\psi}\rangle] \propto \left[ \frac{\langle \tilde{\psi} | [H - \tilde{E}]^2 | \tilde{\psi} \rangle}{\langle \tilde{\psi} | H^2 | \tilde{\psi} \rangle} \right]^{1/2}$$

If  $H$  is linear in  $\vec{a}$ , this can be computed in the small space.

Can obtain proportionality constant if needed

$$\text{Error}[E] \sim \text{Error}[|\tilde{\psi}\rangle]^2$$



## Beyond Bound States

---

## Efficient trial scattering wave functions: the basics (arXiv:2007.03635)

Again, write down a trial wave function:  $|\tilde{\psi}\rangle = \sum \beta_i |\psi_i\rangle$ .

For scattering, it is convenient to work with the radial wave function  $u_\ell(r)$  for partial wave  $\ell$ , which in asymptotic form is

$$u_\ell(r) \xrightarrow[r \rightarrow \infty]{} \sin\left(pr - \frac{1}{2}\ell\pi\right) + K_\ell \cos\left(pr - \frac{1}{2}\ell\pi\right)$$

### Kohn variational principle (KVP)

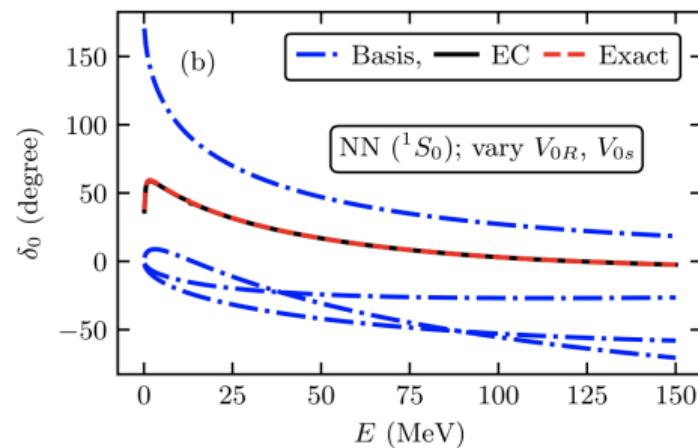
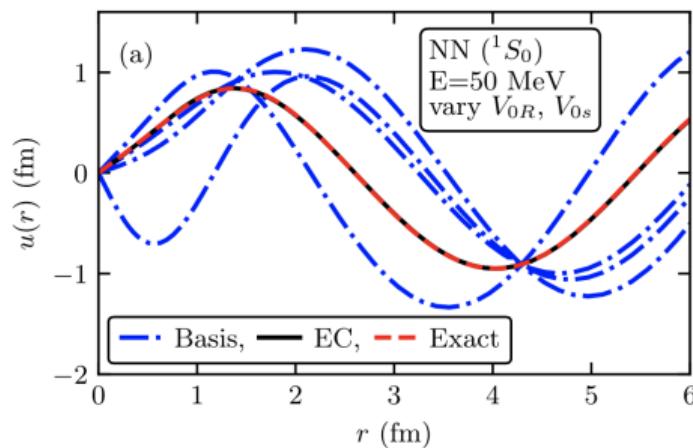
Minimize  $\mathcal{K}_{KVP}[\tilde{\psi}_\ell] = K_\ell - \langle \tilde{\psi}_\ell | H(\vec{a}) - E | \tilde{\psi}_\ell \rangle$  such that  $\langle \tilde{\psi}_\ell | \tilde{\psi}_\ell \rangle = 1$ .

Again, use exact  $u_\ell(r)$  at training points  $\{\vec{a}_i\}$  as the basis.

It is a linear problem, can solve for  $\beta$  analytically and quickly.

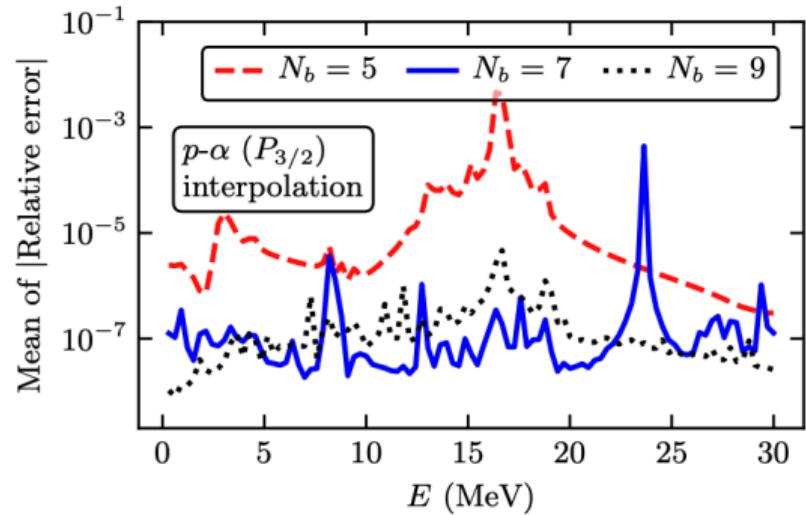
# Trial scattering wave functions in the wild (arXiv:2007.03635)

Can emulate entire scattering wave function and its phase shifts



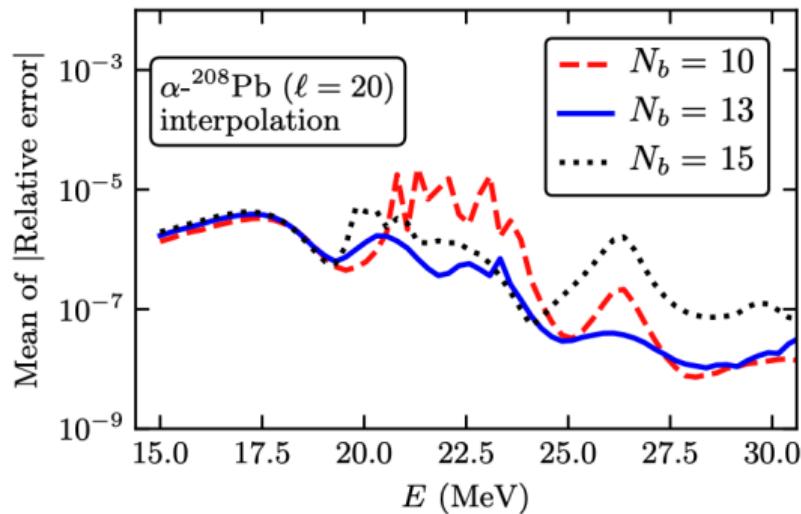
# Trial scattering wave functions in the wild (arXiv:2007.03635)

Can accurately emulate with Coulomb



# Trial scattering wave functions in the wild (arXiv:2007.03635)

Also works with optical potentials



## Efficient trial $K$ or $T$ matrices (arXiv:2106.15608)

Rather than solve the Schrödinger equation, use the Lippmann–Schwinger (LS) equation

$$K = V + VG_0K$$

Propose trial  $K$  matrix

$$\tilde{K} = \sum \beta_i K_i$$

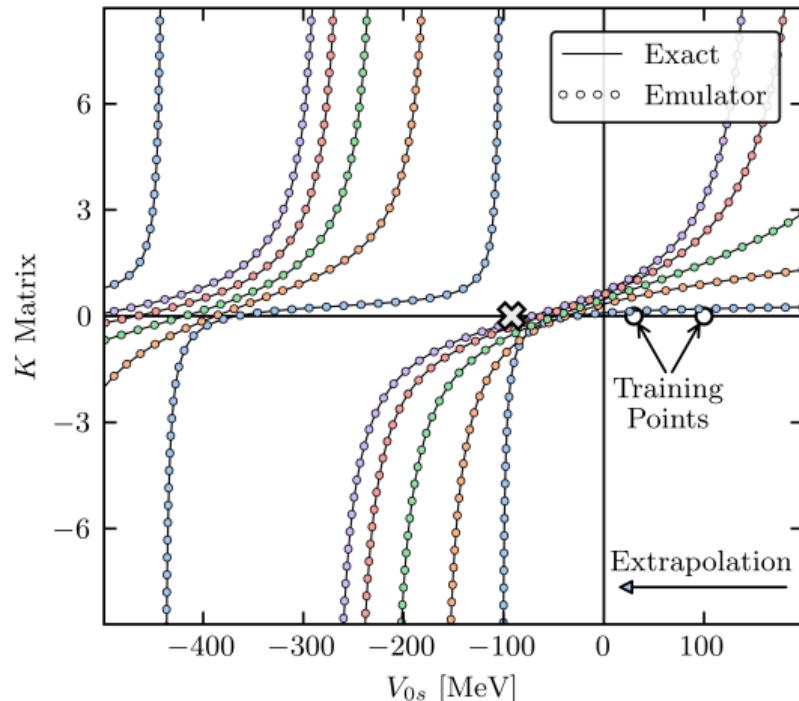
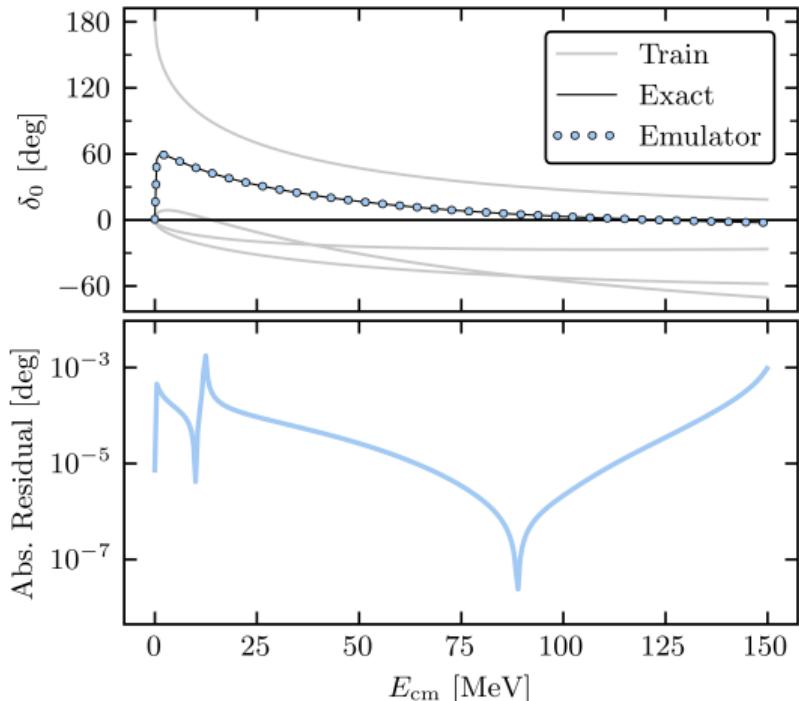
Newton variational principle (NVP)

$$\text{Minimize } \mathcal{K}_{NVP}[\tilde{K}] = V + VG_0\tilde{K} + \tilde{K}G_0V - \tilde{K}G_0\tilde{K} + \tilde{K}G_0VG_0\tilde{K} \quad (\text{no constraints!})$$

As usual, take the **matrix basis**  $\{K_i\}$  from exact solutions of the LS equation.

It is a linear problem, can solve for  $\beta$  analytically and quickly.

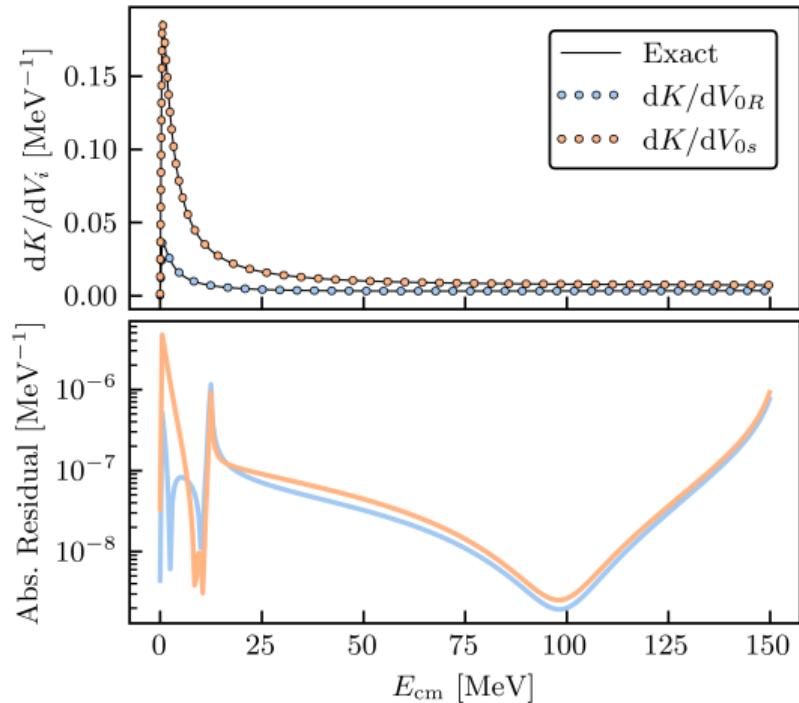
# Trial $K$ matrices in the wild (arXiv:2106.15608)



Can extrapolate very far from support of data, even across singularities

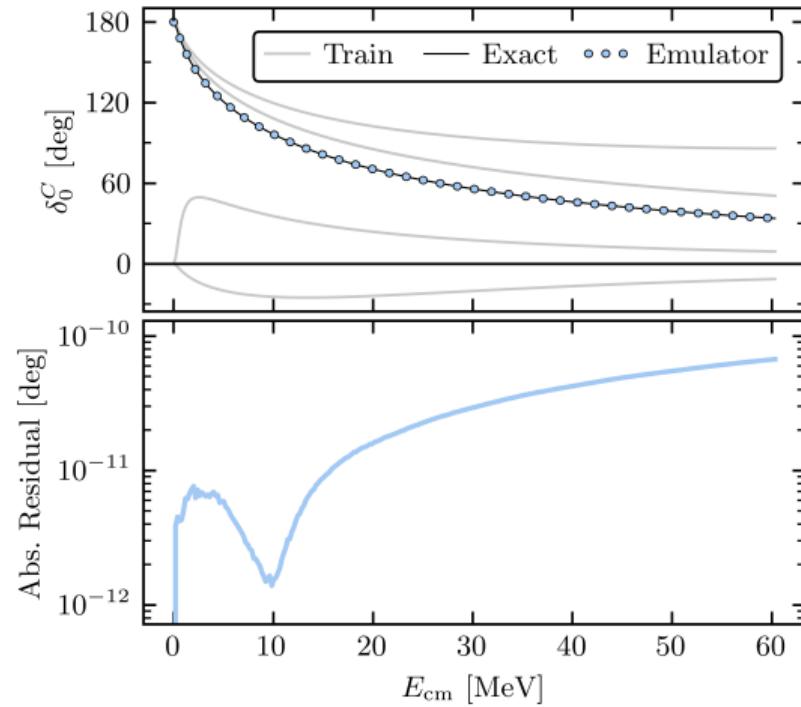
# Trial $K$ matrices in the wild (arXiv:2106.15608)

Can accurately & efficiently emulate  
**gradients** with respect to parameters



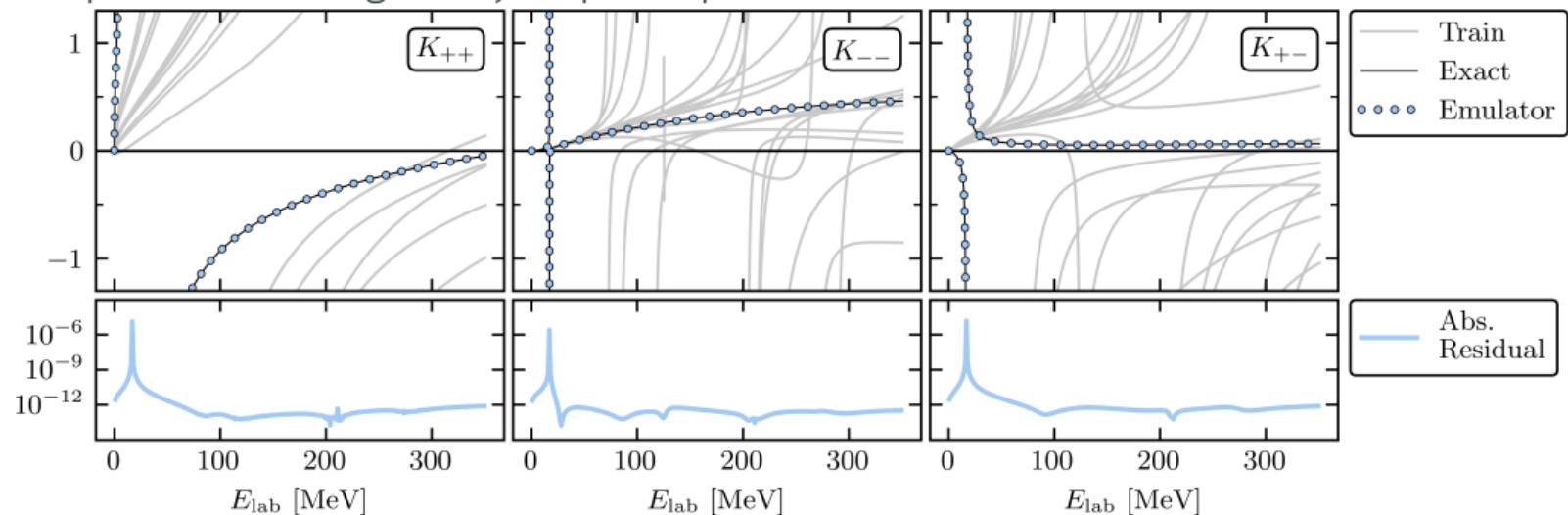
# Trial K matrices in the wild (arXiv:2106.15608)

Can easily handle the Coulomb interaction



# Trial K matrices in the wild (arXiv:2106.15608)

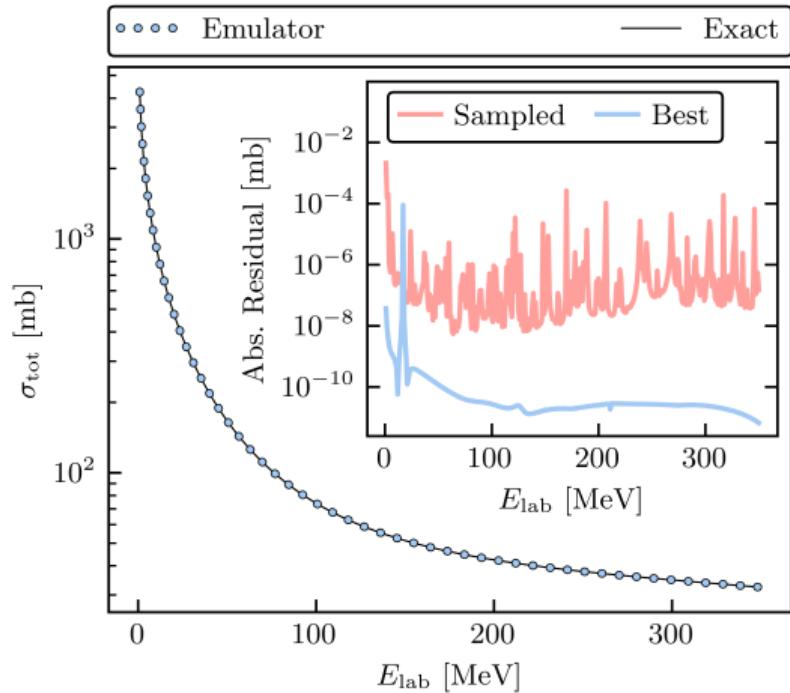
Coupled channels get major speedups with emulation



# Trial $K$ matrices in the wild (arXiv:2106.15608)

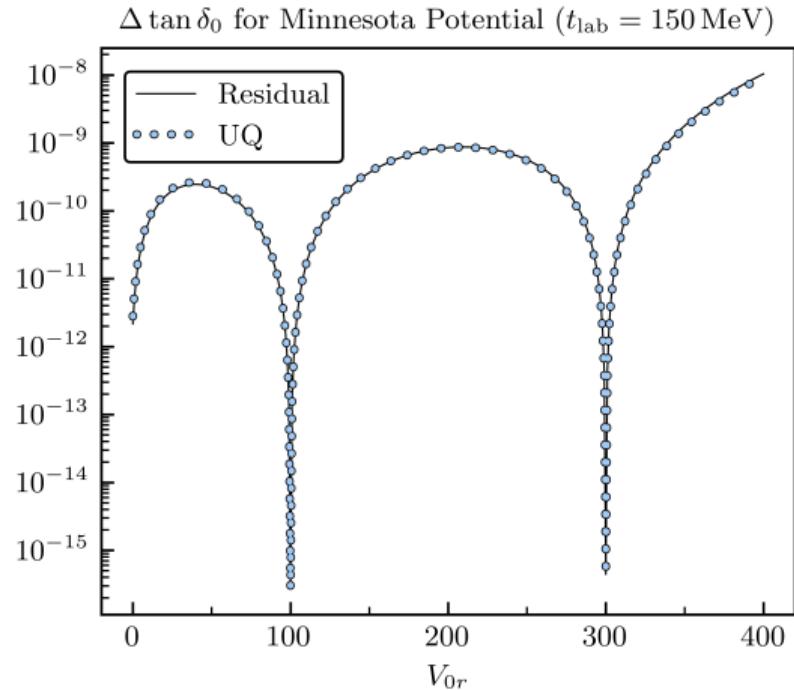
$$\sigma_{\text{tot}}(q) = -\frac{\pi}{2q^2} \sum_{j=0}^{j_{\max}} (2j+1) \operatorname{Re}\left\{ \operatorname{Tr}[S_j(q) - \mathbb{1}] \right\}$$

Multiple emulators across partial waves can be combined to emulate scattering observables. Over 300x improvement in CPU time.



## Trial $K$ matrices in the wild: uncertainties (In progress)

- By following reduced basis approach, UQ for  $\tilde{K}$  is possible
- Determined up to proportionality constant
- Can help propose next training point
- Can be added to error budget if using emulator in the wild



# Density functional theory?! (In progress)

Two reasonable approaches

1. The Kohn-Sham formalism requires self-consistently solving Schrödinger equations for orbitals. One could emulate this step.
2. The ground state energy is minimized as a functional of the density. Just write down a trial density and turn the crank.

I took the latter approach

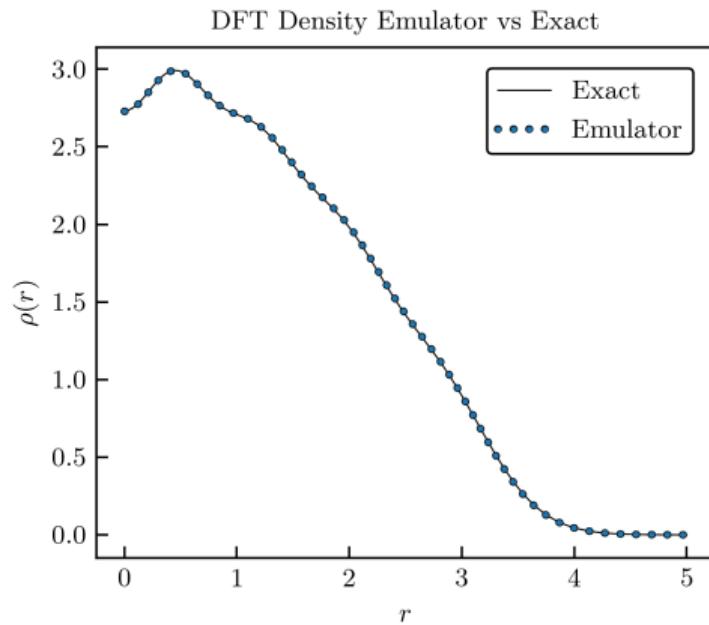
$$\tilde{\rho} = \sum \beta_i \rho_i$$

and minimized

$$E[\tilde{\rho}] = g \sum \epsilon - a \int d^3x [\tilde{\rho}(x)]^2 - b \int d^3x [\tilde{\rho}(x)]^{7/3} - c \int d^3x [\tilde{\rho}(x)]^{8/3}$$

Non-linear means there is no “nice” solution for  $\beta$ . Use an optimizer.

# Density functional theory?! (In progress)



# Emulators for generic differential equations

1. Derive the **Galerkin weak formulation** of the PDE
2. Apply the **reduced basis method**

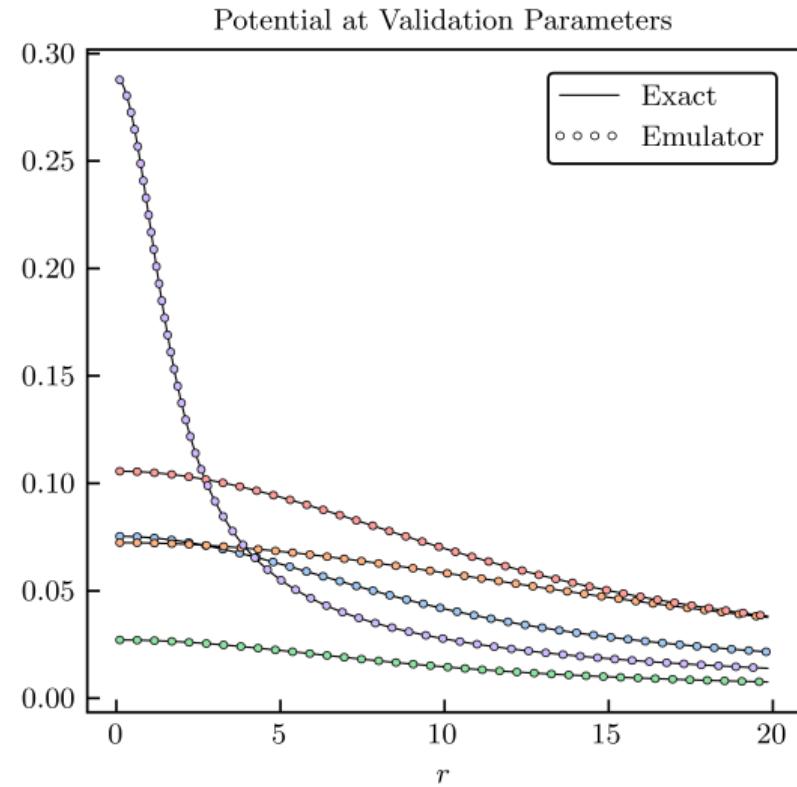
Example [Gaussian  $\rho(\vec{a})$  for variable  $\vec{a}$ ]:

- Poisson equation

$$\nabla^2 \phi = -\rho$$

- Weak form:

$$\int_{\Omega} \nabla \phi \cdot \nabla \psi \, dx = - \int_{\Omega} \rho \psi \, dx$$



## Concluding Remarks

---

# Conclusions

## BAND

The goal of BAND is to translate novel statistical methods of UQ into software tools that address prominent current problems in nuclear physics.

Subspace emulation could play a key role in this effort

## Benefits

- Can radically reduce the size of the problem (not just **eigenvectors!**)
- An extremely effective basis can be chosen **automatically**
- Gets emulator for downstream observables  $\langle \hat{O}(\vec{a}) \rangle$  **for free**

- Promising directions: heavy systems, beyond eigenvalues, DFT, PDEs, etc.
- Error bands are in the works!

# Thank you!

---

[buqeye.github.io](http://buqeye.github.io)