ATPESC 2022

CALCULATION OF NUCLEAR GROUND STATES USING ARTIFICIAL NEURAL NETWORKS.

Corey Adams
Physicist, Computer Scientist

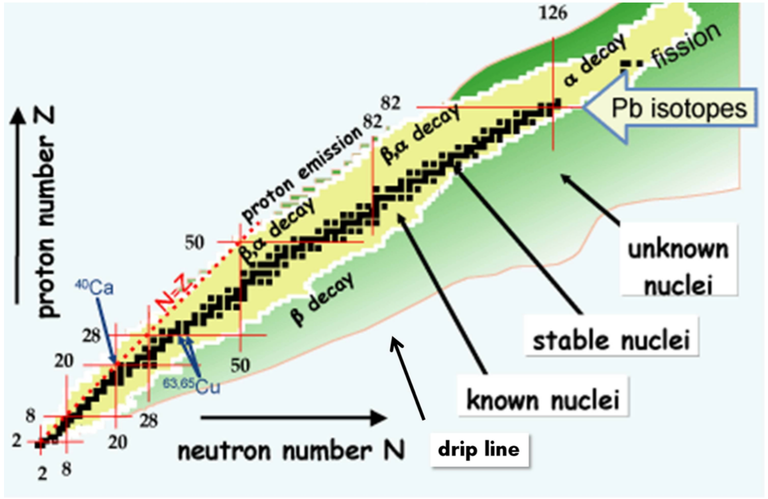
COLLABORATORS
A. Lovato (ANL)
A. Gnech (JLAB)
N. Brawand (ANL)
G. Carleo (EPFL)
N. Rocco (FNAL)

8/12/22

INTRODUCTION

Why nuclear physics?

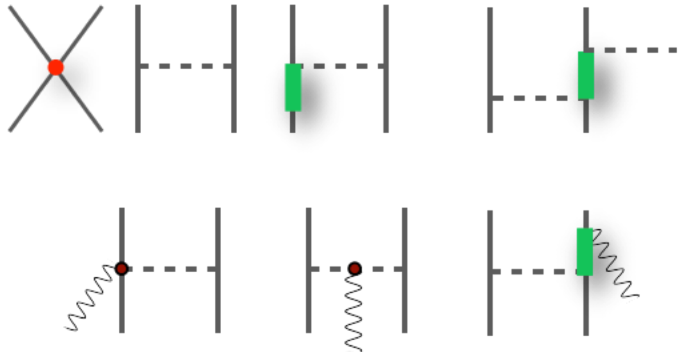
Atomic Nuclei are many-body systems governed by the strong interaction, which exhibit emergent properties such as: shell structure, pairing and superfluidity, deformation, and self-emerging clusters.



Understanding how the properties of nuclei emerge from QCD is a long-standing goal of nuclear physics.

NUCLEAR MANY BODY PHYSICS

At low energies, the quarks and gluons are **confined** within the hadrons: protons, neutrons and pions.

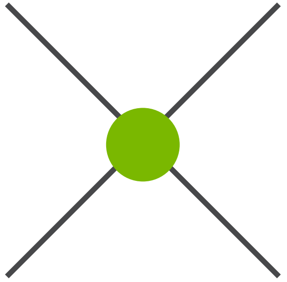


$$H = \sum_i \frac{\vec{p}_i^{\,2}}{2m} + \sum_{i<j} v_{ij} + \sum_{i<j<k} V_{ijk} + \ldots$$

We can approximate QCD through **effective field theories,** allowing us to compute observables

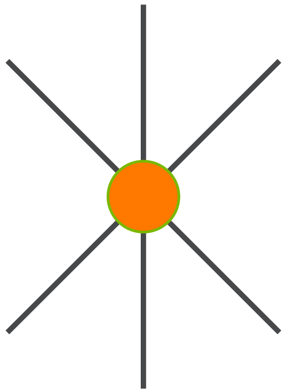
PION-LESS NUCLEAR HAMILTONIAN

An Effective Field Theory with 2- and 3- body interactions



$$v_{12} = C_1 v_\Lambda(r_{12}) + C_2 v_\Lambda(r_{12})\sigma_{12}$$

C_1 and C_2 fit to nucleon-nucleon scattering data



$$v_{123} = D_0 \sum_{cyc} v_\Lambda(r_{12}) v_\Lambda(r_{13})$$

D_0 fixed with the binding energy of ^{3}H

THE NUCLEAR MANY-BODY PROBLEM

- The non-relativistic many body theory is solving the Schrodinger equation:

$$H\psi_n(R) = E_n\psi_n(R) \qquad R = (\vec{x}_1, s_{1,z}, \tau_{1,z} \ldots)$$

$$H \equiv V(R) - \frac{\hbar^2}{2m}\nabla^2$$

- The exact solution of this is **exponentially hard.**

- The methods described in this talk solve this equation approximately, and while we target Nuclear many-body systems it is broadly applicable to many-body quantum systems.

VARIATIONAL MONTE CARLO

- The Variational Principle of Quantum Mechanics guarantees that for any trial wavefunction, the expectation of the energy of that wavefunction is greater than the ground state:

$$\psi_T = \psi_T(R, \vec{\theta}) \qquad E_T = \frac{\langle \psi_T | H | \psi_T \rangle}{\langle \psi_T | \psi_T \rangle} \geq E_0$$

- Trial wavefunctions are parametrized in some way, and so you may optimize the trial wavefunction to reduce the expectation of the energy.

$$min(E_T) : \theta_j \rightarrow \theta_j - \eta \frac{\partial}{\partial \theta_j} E_T$$

- Ultimately, the lowest energy found represents the best approximation of the ground state.

COMPUTING EXPECTATION VALUES

- The trial wavefunction, in just one dimension, is simple to compute numerically. But with many-body problems in 3 dimensions, the number of dimensions in the integral scales as $3xN_{\text{particles}}$.

$$E_t = \frac{\int dr \Psi_T^*(r,\vec{\theta}) H \Psi_T(r,\vec{\theta})}{\int dr \Psi_T^*(r,\vec{\theta}) \Psi_T(r,\vec{\theta})} \qquad r \equiv (\vec{r_1}, \vec{r_2}, ... \vec{r_N})$$

- Sampling this integral in a dense or even adaptive way is computationally very very hard!
- The **central limit theorem** provides a way to approximate this multi-dimensional integral.

CENTRAL LIMIT ESTIMATES

- Let P(x) be a probability distribution, and (x_1, … x_N) be drawn from P(x). For the function f(x), you can define a new random variable:

$$S_N = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

- By the central limit theorem:

$$\bar{S}_N = \int dx P(x) f(x) \qquad \sigma_N = \sqrt{\frac{1}{N}\left[\int P(x) f(x)^2 dx - \bar{S}_N\right]}$$

$$I = \int dx f(x) = \int dx P(x) \frac{f(x)}{P(x)}$$

VARIATIONAL MEASUREMENTS

- The integral to estimate the energy of a trial wavefunction is:

$$E_T = \frac{\langle \psi_T | \, H \, | \psi_T \rangle}{\langle \psi_T | \psi_T \rangle} = \frac{\int dR \, \langle \psi_T | R \rangle \, \langle R | H | \psi_T \rangle}{\int dR \, \langle \psi_T | R \rangle \, \langle R | \psi_T \rangle}$$

- Define a quantity $E_L(R)$: $E_L(R) \equiv \frac{H\psi_T(R)}{\psi_T(R)}$

$$E_T = \frac{\int dR |\psi_T(R)|^2 E_L(R)}{\int dR |\psi_T(R)|^2}$$

TRIAL ENERGY ESTIMATE

- Numerically approximate the integral by sampling R from the probability distribution P(R):

$$P(R) = \frac{|\psi(R)|^2}{\int dR|\psi_T(R)|^2}$$

$$\langle E_T \rangle = \frac{1}{N}\sum_n E_L(R_n)$$

- And, the integration error can be estimated just as easily:

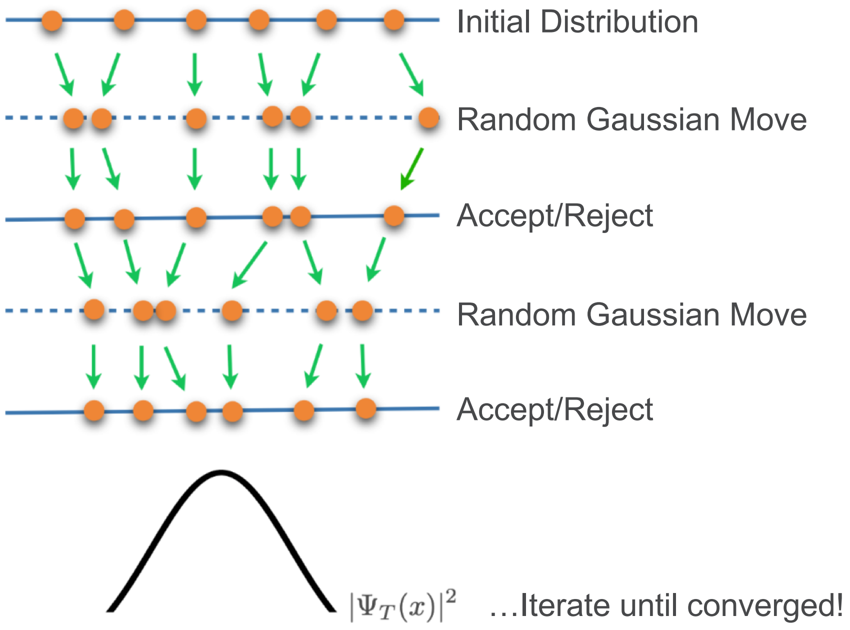
$$\langle E_T^2 \rangle = \frac{1}{N}\sum_n E_L^2(R_n)$$

$$E_L(R) \equiv \frac{H\psi_T(R)}{\psi_T(R)}$$

M(RT)^2 SAMPLING

- Now that we have a tool for computing integrals in high dimensionality, we can compute the energy for any trial wavefunction as long as we sample x_i from the probability distribution $P(x_i)$.
- The M(RT)^2 algorithm* provides a technique to sample from any arbitrary probability distribution under general conditions.
- Referring to each sample as a "walker."

*named for N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller



https://github.com/Nuclear-Physics-with-Machine-Learning/AI4NP_School/blob/main/Lectures/MLNP_school_I.pdf

PRACTICAL CONSIDERATIONS

- The $M(RT)^2$ algorithm has some nice properties:
  - We can sample nearly any function;
  - It is numerically and analytically fairly simple;
  - It is easily parallelized up to however many configurations we want
- Also: The $M(RT)^2$ algorithm has some unfortunate convergence properties:
  - It takes a large number of steps to converge to the target distribution, especially initially.
  - Subsequent samples are often frequently correlated with each other, requiring intermediate steps to re-thermalize.
  - Discarding sampled configurations initially and with each re-thermalization is quite wasteful.

ENERGY MINIMIZATION

- Recall the wavefunction, and the values we must compute:

$$\psi_T = \psi_T(R, \vec{\theta}) \qquad E_L(R) \equiv \frac{H\psi_T(R)}{\psi_T(R)}$$

- So,

$$\frac{\partial \langle E_T \rangle}{\partial \theta_i} = 2\left(\frac{\langle \partial_i \psi_T | H | \psi_T \rangle}{\langle \psi_T | \psi_T \rangle} - E_T \frac{\langle \partial_i \psi_T | \psi_T \rangle}{\langle \psi_T | \psi_T \rangle} \right)$$

- Define:

$$O^i \psi_T(R, \vec{\theta}) \equiv \frac{\partial}{\partial \theta_i} \psi_T(R, \vec{\theta}) \qquad G^i \equiv \frac{\partial \langle E_T \rangle}{\partial \theta_i} = 2\left(\langle O^i H \rangle - \langle E_T \rangle \langle O^i \rangle \right)$$

CALCULUS INTERLUDE

- So far, we've encountered a number of derivatives:
  - The Hamiltonian operator requires a second derivative to compute the energy of the trial model, as a function of the inputs.
  - The Gradient Calculation requires derivatives of the trial model as a function of the parameters.
- We can either figure out these derivates analytically (hard), numerically (slow), or leverage a machine learning framework that has automatic differentiation.
  - Which one?
- In short: represent our "trial wavefunction" with a machine learning neural network.

ANTI-SYMMETRY

- A wavefunction of many fermions must be anti-symmetric under the exchange of any two particles. We enforce this directly in the network with the Slater determinant, in combination with a fully-symmetric DeepSets based correlator (U)

$$S = \begin{pmatrix} \langle x_1|\zeta_1\rangle & \langle x_2|\zeta_1\rangle & \dots & \langle x_N|\zeta_1\rangle \\ \langle x_1|\zeta_2\rangle & \langle x_2|\zeta_2\rangle & \dots & \langle x_N|\zeta_2\rangle \\ \vdots & & \ddots & \vdots \\ \langle x_1|\zeta_N\rangle & \langle x_2|\zeta_N\rangle & \dots & \langle x_N|\zeta_N\rangle \end{pmatrix}$$

$$S_{\text{deuteron}} = \begin{pmatrix} \langle x_1|R_1 p\uparrow\rangle & \langle x_2|R_1 p\uparrow\rangle \\ \langle x_1|R_2 n\uparrow\rangle & \langle x_2|R_2 n\uparrow\rangle \end{pmatrix}$$

x_i is a generalized coordinate of spatial position, spin, and isospin.

$$|\zeta_i\rangle = |R_i\rangle\,|s_i\rangle\,|\tau_i\rangle$$

NEURAL NETWORK QUANTUM STATES

- In general, we need a wavefunction of the form (S is matrix):

$$\psi(\vec{r}_1, ...\vec{r}_N) = \boxed{e^{U(\vec{r}_1,...\vec{r}_N)}} \det(S)$$

- In practice, we enforce full symmetry of the correlator under exchange of particles using the **DeepSets** formalism:

$$U(\vec{r}_1, ..., \vec{r}_A) = \rho_U\left(\sum_{\vec{r}_i} \phi_U(\vec{r}_i)\right) \qquad \phi, \rho = ANN$$

- Each particle's location is mapped to a latent space, and the latent space of all particles is summed to destroy individual interactions, then mapped to a single value.

NEURAL NETWORK PHYSICALITY

- The neural network implementation must also obey physical constraints: must be twice differentiable, continuous in the first derivative, and for a bound state must go to 0 at infinity.
- In practice, we enforce this with select activation functions (yes to tanh/sigmoid, no to ReLU!). A correlator function U is also augmented with a confinement term (goes to 0 at infinity):

$$U(\vec{r}_1, ..., \vec{r}_A) = \rho_U \left(\sum_{\vec{r}_i} \phi_U(\vec{r}_i) \right) - \alpha \sum_i \vec{r}_i^2$$

STOCHASTIC RECONFIGURATION

- The gradients computed above can be improved via “Stochastic Reconfiguration”
  – https://journals.aps.org/prb/abstract/10.1103/PhysRevB.71.241103

$$S_R^{mn} \equiv \langle O^m O^n \rangle - \langle O^m \rangle \langle O^n \rangle \qquad S_{R,\epsilon}^{-1} \equiv (S_R + \mathbb{I}\epsilon)^{-1}$$

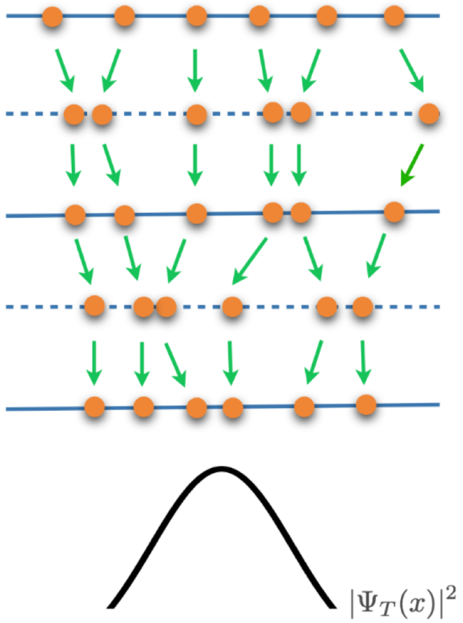
- Effectively, this flattens the space of optimization and is a 2nd order approach

$$\theta_j \to \theta_j - \eta \sum_j S_{R,\epsilon}^{-1} \frac{\partial}{\partial \theta_j} E_T$$

- But, this requires the jacobian matrix of the network!

ALGORITHM SUMMARY 1

- For a trial wavefunction, create sets of $N_{walkers}$ to use for a numerical integration.
- Thermalize the walkers for N_{therm} iterations at the start; between each measurement use N_{void} steps to remove correlations in measurements.
- For each set of thermalized, de-correlated walkers, compute the observable properties:
  - E_T, it's variational derivatives, the reconfiguration matrix S_{ij}.



ALGORITHM SUMMARY 2

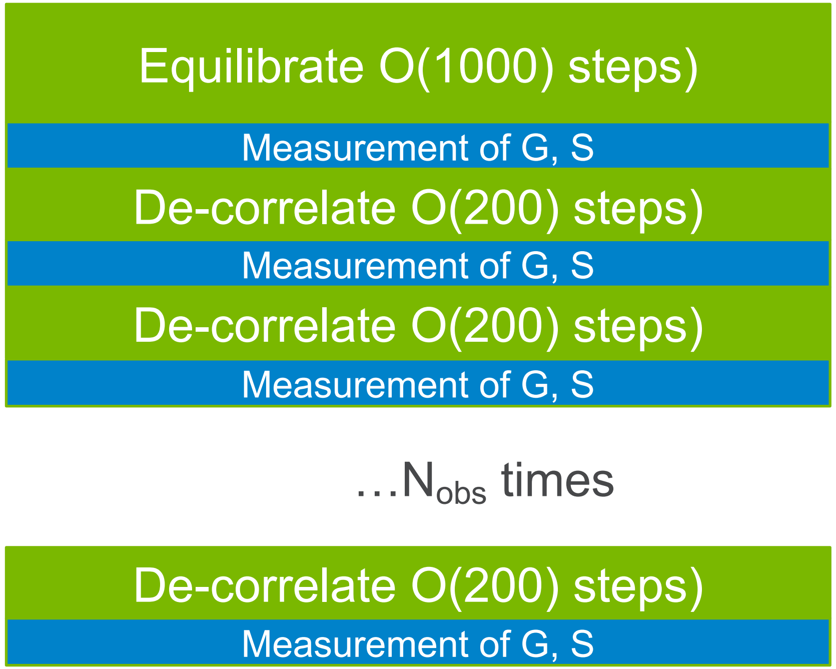
- Accumulate the **observables** for N_{obs} iterations;

$$G^i \equiv \frac{\partial \langle E_T \rangle}{\partial \theta_i} = 2\left(\langle O^i H \rangle - \langle E_T \rangle \langle O^i \rangle\right)$$

$$S_R^{mn} \equiv \langle O^m O^n \rangle - \langle O^m \rangle \langle O^n \rangle$$

- Update the wave function according to the accumulated observables and the update rule:

$$\theta_j \to \theta_j - \eta \sum_j S_{R,\epsilon}^{-1} \frac{\partial}{\partial \theta_j} E_T$$



Equilibrate O(1000) steps)
Measurement of G, S
De-correlate O(200) steps)
Measurement of G, S
De-correlate O(200) steps)
Measurement of G, S
…N_{obs} times
De-correlate O(200) steps)
Measurement of G, S

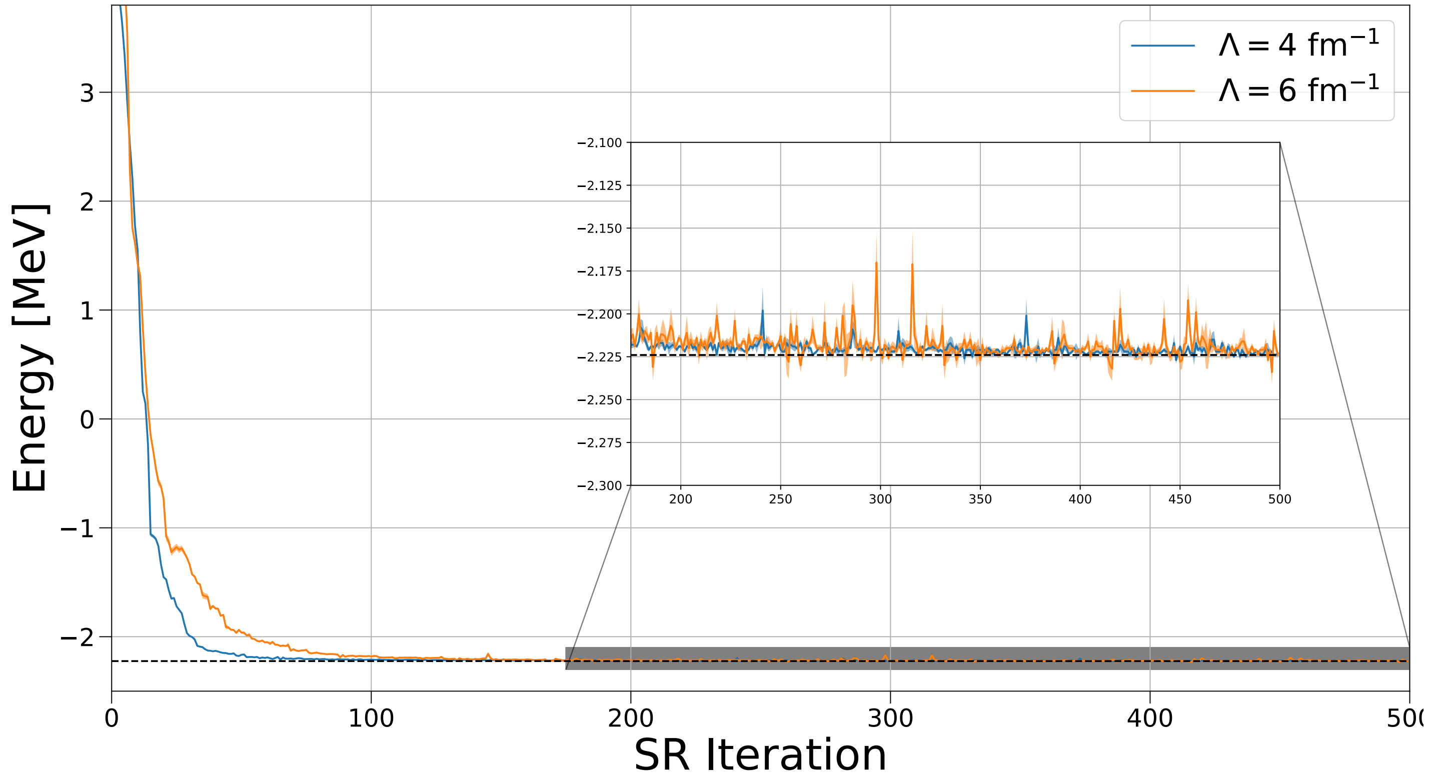
COMPUTATIONAL EFFICIENCY

- This algorithm can (and has been) implemented in there DL frameworks (TF, Torch, Jax). Jax is the clear winner for computational efficiency.
- Torch is imperative: the ”walk” algorithm is too slow, and makes terrible use of the GPU.
  – LibTorch is better, but has concurrency issues when computing the Jacobian matrix.
  – Generally torch is great when each GPU op is Big. It falls over when there are many many small ops.
- Tensorflow is better, but it’s graph compilation stage can be tedious and detrimental to start-up times as the problem size scales up.
  – Has excellent scaling properties, though!

COMPUTATIONAL EFFICIENCY (2)

- The non-traditional derivatives of this algorithm also are a challenge:
  - Need a second derivative with respect to input variables, AND a jacobian.
  - No simple vectorization and poor performance with both TF (jacobian) and Torch (both!)
- Jax offers a solution to all of this:
  - Easy to compile the many-small-ops Metropolis algorithm
  - Easy to vectorize the gradient of the wavefunction over all parameters (Jacobian)
  - Easy to vectorize the 2nd derivatives.
- In short: if you have a “weird” algorithm using machine learning, Jax is awesome.

SOLVING THE DEUTERON

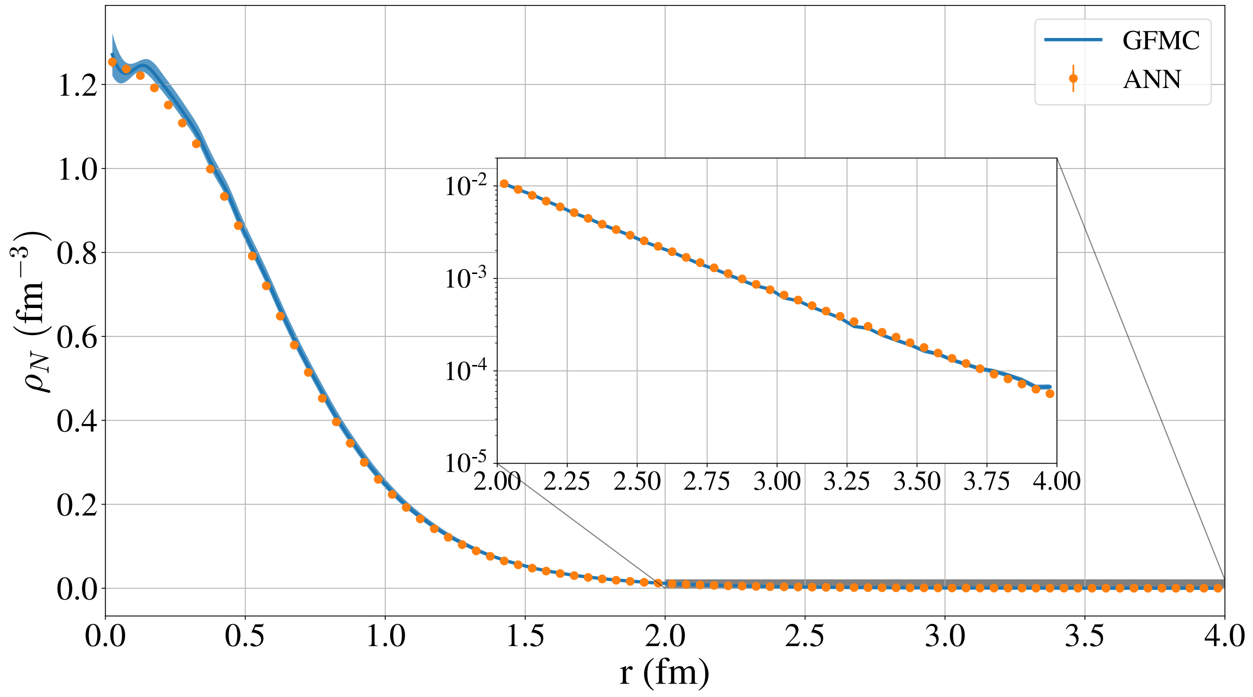


NUCLEI UP TO A=6

Nucleus	Potential	ANN		HH		Exp.	
		E (MeV)	r_{ch} (fm)	E (MeV)	r_{ch} (fm)	E (MeV)	r_{ch} (fm)
^{2}H	NN	$-2.242(1)$	$2.120(5)$	-2.242	$2.110(2)$	-2.225	2.128
^{3}H	NN	$-9.511(1)$	$1.658(4)$	-9.744	$1.656(4)$	-8.475	$1.755(86)$
	$3N$	$-8.232(1)$	$1.750(3)$	-8.475	$1.747(6)$		
^{3}He	NN	$-8.800(1)$	$1.845(3)$	-9.035	$1.848(6)$	-7.718	$1.964(1)$
	$3N$	$-7.564(1)$	$1.961(3)$	-7.811	$1.969(8)$		
^{4}He	NN	$-36.841(1)$	$1.484(3)$	-37.06	$1.485(4)$	-28.30	1.678
	$3N$	$-27.903(1)$	$1.643(2)$	-28.17	$1.646(4)$		
^{6}He	NN	$-37.25(4)$	$1.895(2)$	$-37.96(8)$	$1.71(1)$	-29.27	$2.05(1)$
	$3N$	$-27.46(2)$	$> 4.89(1)$	$-27.41(8)$	> 2.73		
^{6}Li	NN	$-42.04(1)$	$2.248(3)$	$-42.51(5)$	$2.09(2)$	-31.99	$2.54(3)$
	$3N$	$-30.82(3)$	$3.049(2)$	$-31.00(8)$	> 2.74		

Table 1 from https://link.springer.com/article/10.1007/s00601-021-01706-0

CONVERGENCE OF HELIUM



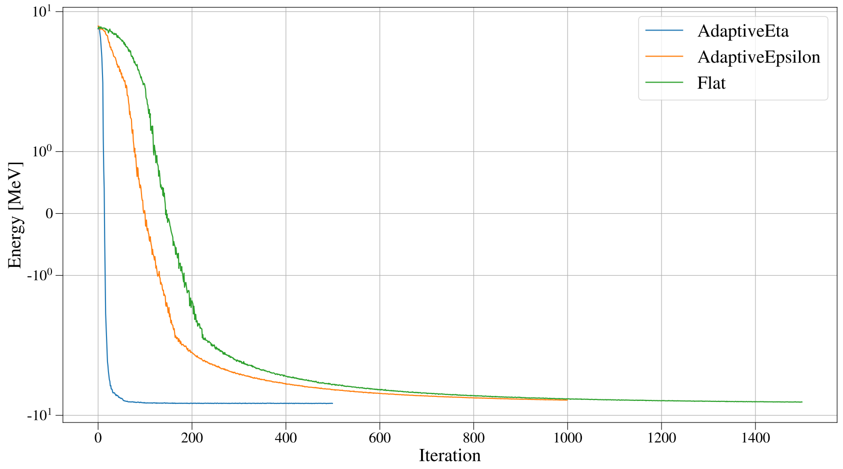
The point-nucleon density of ^{4}He compared to the classical, Green's field Monte Carlo Technique – accurate over 4 orders of magnitude.

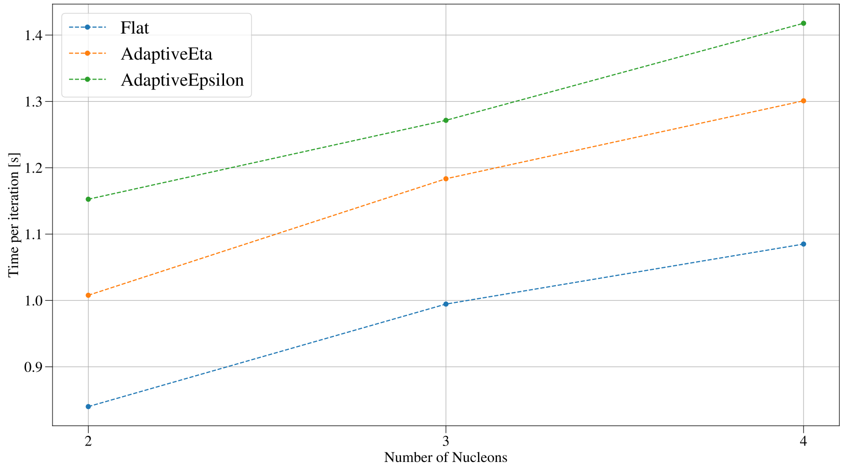
Figure 2 from https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.127.022502

OPTIMIZATION TRICKS

- The iterative update of parameters requires several hyperparameters, particularly the learning rate and the regularization parameter for the inversion of the Stochastic Reconfiguration matrix.
- Instead of picking hyperparameters, we can experiment:
  - Set values for the parameters, ensure the changes in the wavefunction are small.
  - update the wavefunction, recompute the energy
    - Because updates are constrained to be **small,** the previous walk can be reused by rescaling the probability (psi_new^2 / psi_old^2)
  - Choose the step with the best "next" energy

OPTIMIZATION TRICKS

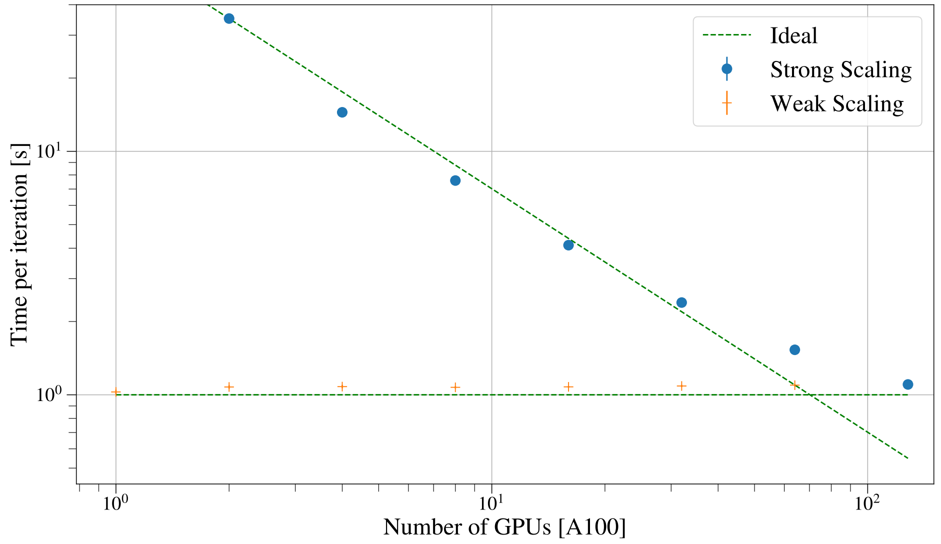




The “AdaptiveEta” (learning rate) quickly out-performs the standard algorithm with minimal additional computational cost (10% slower).

SCALABLE MACHINE LEARNING

- The algorithm, as designed, is easily scalable to multiple compute systems:
  - We compute the observables a total number of N_{obs} times
  - This can be trivially distributed across M GPUs, as long as N_{obs} / M is an integer.
- Nearly perfect weak scaling up to hundreds of A100 GPUs.



ONGOING WORK

- We continue to develop these techniques with the aim of solving bigger and bigger systems.
  – We intend to solve the Calcium nucleus on Polaris this year.
- Our software is open source and available on github with minimial software requirements (tensorflow) and no input requirements:
  – https://github.com/Nuclear-Physics-with-Machine-Learning
  – This also includes a tutorial session on these numerical techniques presented at the 2021 AI-in-Nuclear-Physics winter School
    - (Including hands-on exercises in Tensorflow, if you want to try this out – we solve the hydrogen atom with machine learning)

THANK YOU!