Third part, quantum mechanical studies

Morten Hjorth-Jensen[1]

Department of Physics and Center for Computing in Science Education,
University of Oslo, Norway[1]

Geilo Winter School, March 10-20, 2025

Many-body physics, Quantum Monte Carlo and deep learning

Given a hamiltonian H and a trial wave function Ψ_T, the variational principle states that the expectation value of $\langle H \rangle$, defined through

$$\langle E \rangle = \frac{\int d\boldsymbol{R} \Psi_T^*(\boldsymbol{R}) H(\boldsymbol{R}) \Psi_T(\boldsymbol{R})}{\int d\boldsymbol{R} \Psi_T^*(\boldsymbol{R}) \Psi_T(\boldsymbol{R})},$$

is an upper bound to the ground state energy E_0 of the hamiltonian H, that is

$$E_0 \leq \langle E \rangle.$$

In general, the integrals involved in the calculation of various expectation values are multi-dimensional ones. Traditional integration methods such as the Gauss-Legendre will not be adequate for say the computation of the energy of a many-body system. **Basic philosophy: Let a neural network find the optimal wave function**

Quantum Monte Carlo Motivation

Basic steps

Choose a trial wave function $\psi_T(\boldsymbol{R})$.

$$P(\boldsymbol{R},\boldsymbol{\alpha}) = \frac{|\psi_T(\boldsymbol{R},\boldsymbol{\alpha})|^2}{\int |\psi_T(\boldsymbol{R},\boldsymbol{\alpha})|^2 \, d\boldsymbol{R}}.$$

This is our model, or likelihood/probability distribution function (PDF). It depends on some variational parameters $\boldsymbol{\alpha}$. The approximation to the expectation value of the Hamiltonian is now

$$\langle E[\boldsymbol{\alpha}]\rangle = \frac{\int d\boldsymbol{R}\Psi_T^*(\boldsymbol{R},\boldsymbol{\alpha})H(\boldsymbol{R})\Psi_T(\boldsymbol{R},\boldsymbol{\alpha})}{\int d\boldsymbol{R}\Psi_T^*(\boldsymbol{R},\boldsymbol{\alpha})\Psi_T(\boldsymbol{R},\boldsymbol{\alpha})}.$$

Quantum Monte Carlo Motivation

Define a new quantity

$$E_L(\boldsymbol{R},\boldsymbol{\alpha}) = \frac{1}{\psi_T(\boldsymbol{R},\boldsymbol{\alpha})} H\psi_T(\boldsymbol{R},\boldsymbol{\alpha}),$$

called the local energy, which, together with our trial PDF yields

$$\langle E[\boldsymbol{\alpha}]\rangle = \int P(\boldsymbol{R}) E_L(\boldsymbol{R},\boldsymbol{\alpha}) d\boldsymbol{R} \approx \frac{1}{N}\sum_{i=1}^{N} E_L(\boldsymbol{R_i},\boldsymbol{\alpha})$$

with N being the number of Monte Carlo samples.

Energy derivatives

The local energy as function of the variational parameters defines now our **objective/cost** function.
To find the derivatives of the local energy expectation value as function of the variational parameters, we can use the chain rule and the hermiticity of the Hamiltonian.
Let us define (with the notation $\langle E[\boldsymbol{\alpha}]\rangle = \langle E_L\rangle$)

$$\bar{E}_{\alpha_i} = \frac{d\langle E_L\rangle}{d\alpha_i},$$

as the derivative of the energy with respect to the variational parameter α_i We define also the derivative of the trial function (skipping the subindex T) as

$$\bar{\Psi}_i = \frac{d\Psi}{d\alpha_i}.$$

Derivatives of the local energy

The elements of the gradient of the local energy are

$$\bar{E}_i = 2\left(\langle\frac{\bar{\Psi}_i}{\Psi}E_L\rangle - \langle\frac{\bar{\Psi}_i}{\Psi}\rangle\langle E_L\rangle\right).$$

From a computational point of view it means that you need to compute the expectation values of

$$\langle\frac{\bar{\Psi}_i}{\Psi}E_L\rangle,$$

and

$$\langle\frac{\bar{\Psi}_i}{\Psi}\rangle\langle E_L\rangle$$

These integrals are evaluted using MC integration (with all its possible error sources). Use methods like stochastic gradient or other minimization methods to find the optimal parameters.

Monte Carlo methods and Neural Networks

Machine Learning and the Deuteron by Kebble and Rios and Variational Monte Carlo calculations of $A \leq 4$ nuclei with an artificial neural-network correlator ansatz by Adams et al.
Adams et al:

$$H_{LO} = -\sum_i \frac{\vec{\nabla}_i^2}{2m_N} + \sum_{i<j} \left(C_1 + C_2\, \vec{\sigma_i} \cdot \vec{\sigma_j}\right) e^{-r_{ij}^2 \Lambda^2/4} + D_0 \sum_{i<j<k} \sum_{\text{cyc}} e^{-\left(r_{ik}^2 + r_{ij}^2\right)\Lambda^2/4}, \quad (1)$$

where m_N is the mass of the nucleon, $\vec{\sigma_i}$ is the Pauli matrix acting on nucleon i, and $\sum_{\text{cyc}}$ stands for the cyclic permutation of i, j, and k. The low-energy constants C_1 and C_2 are fit to the deuteron binding energy and to the neutron-neutron scattering length

Deep learning neural networks, Variational Monte Carlo calculations of $A \leq 4$ nuclei with an artificial neural-network correlator ansatz by Adams et al.

An appealing feature of the neural network ansatz is that it is more general than the more conventional product of two- and three-body spin-independent Jastrow functions

$$|\Psi_V^J\rangle = \prod_{i<j<k} \left(1 - \sum_{\text{cyc}} u(r_{ij})u(r_{jk})\right) \prod_{i<j} f(r_{ij})|\Phi\rangle\,, \qquad (2)$$

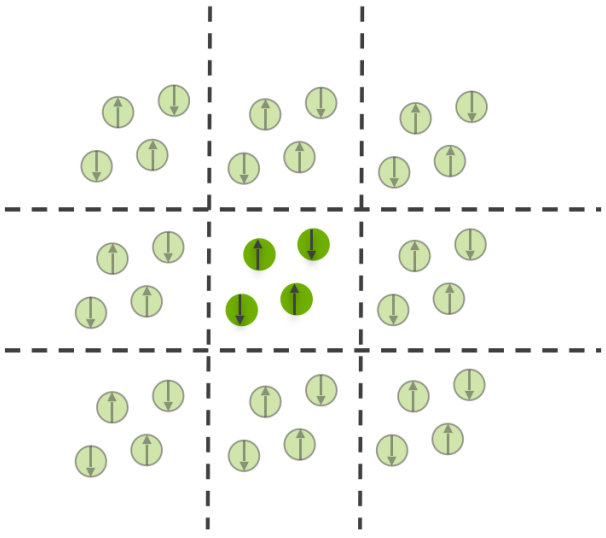
which is commonly used for nuclear Hamiltonians that do not contain tensor and spin-orbit terms. The above function is replaced by a four-layer Neural Network.

Ansatz for a fermionic state function, Jane Kim et al, Commun Phys 7, 148 (2024)

$$\Psi_T(\boldsymbol{X}) = \exp U(\boldsymbol{X})\Phi(\boldsymbol{X}).$$

1. Build in fermion antisymmetry for network compactness
2. Permutation-invariant Jastrow function improves ansatz flexibility
3. Build U and Φ functions from fully connected, deep neural networks
4. Use Slater determinant (or Pfaffian) Φ to enforce antisymmetry with single particle wavefunctions represented by neural networks

Nuclear matter setup

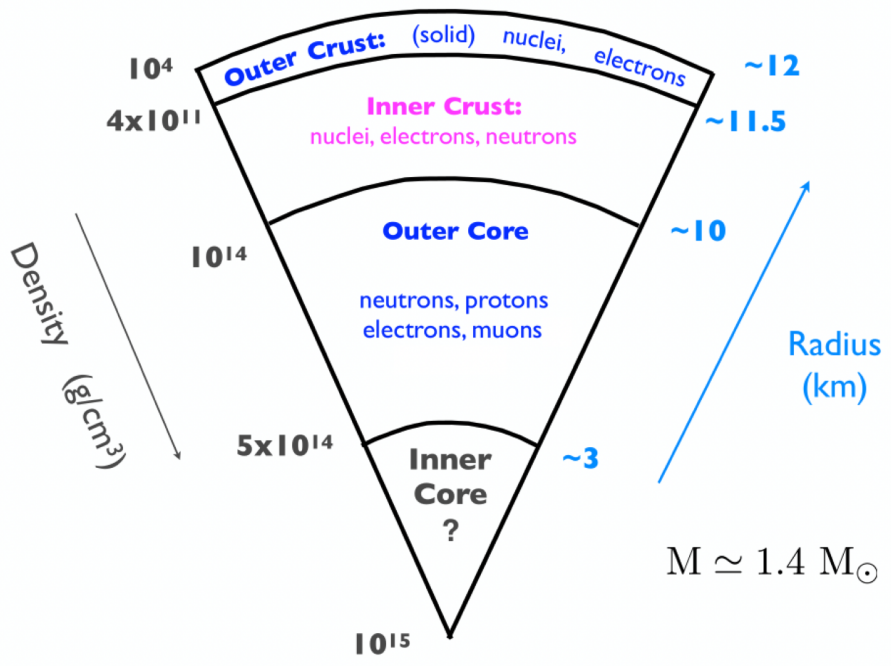


- Periodic boundary conditions and coordinate system

$$\mathbf{r}_i \longrightarrow \tilde{\mathbf{r}}_i = \left\{ \sin\left(\frac{2\pi}{L}\mathbf{r}_i\right), \cos\left(\frac{2\pi}{L}\mathbf{r}_i\right) \right.$$

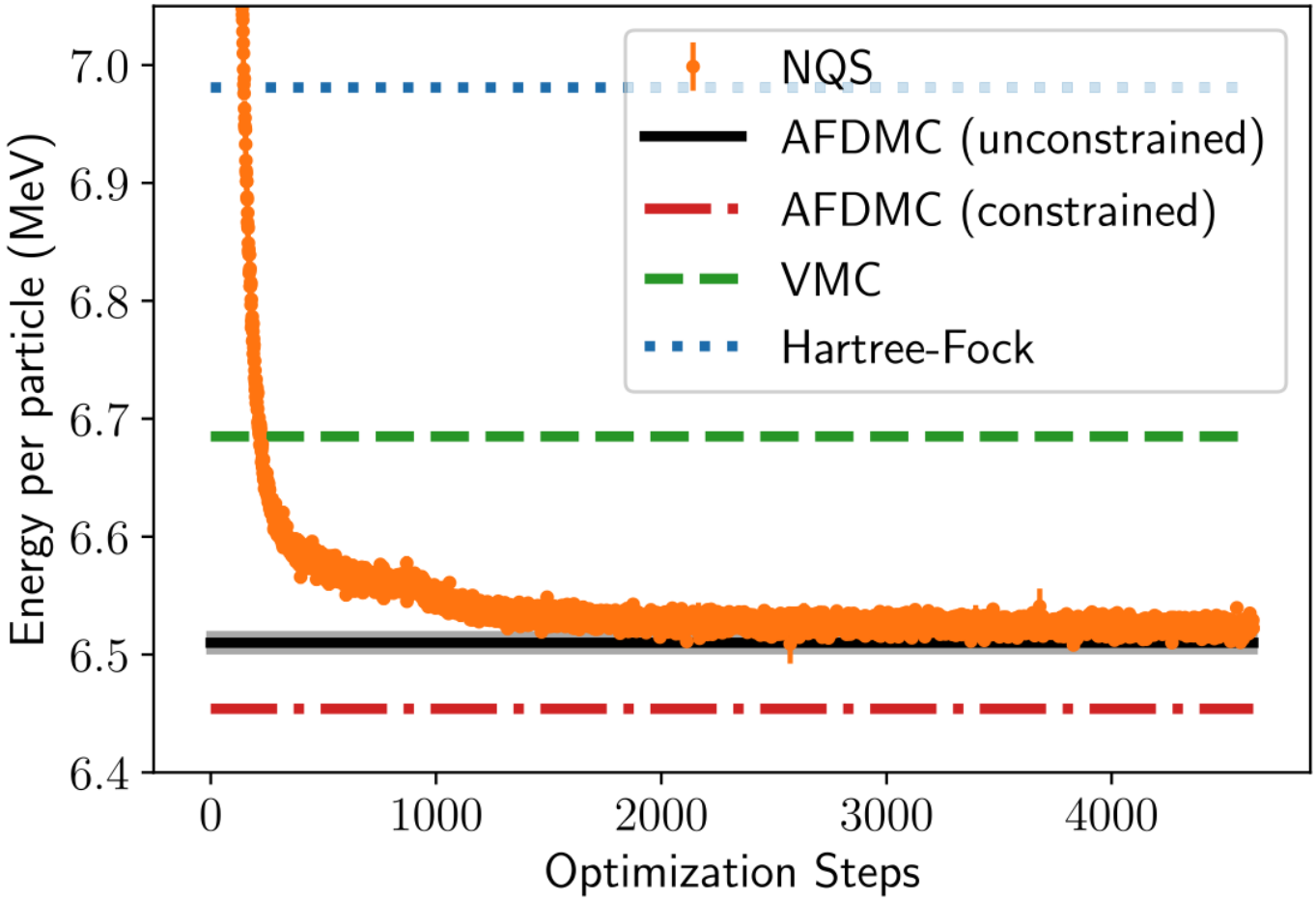
- Potential energy contribution from particle images
- Remove Coulomb potential

Neutron star structure

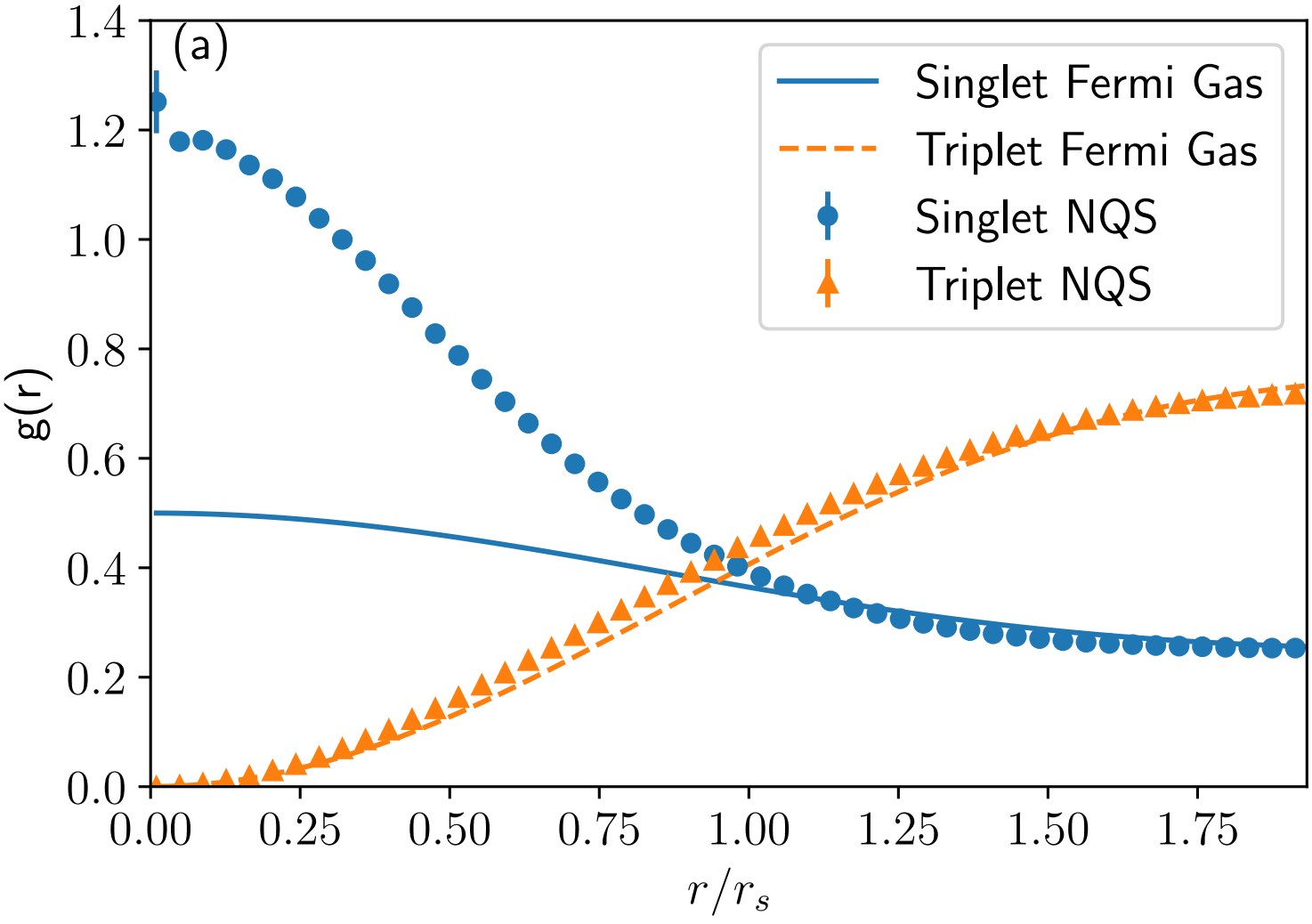


- Mostly neutrons but composition varies with density
- Nuclei in crust are squeezed into uniform matter in core
- Likely neutron superfluid in inner crust and outer core
- Calculations currently focus on inner crust

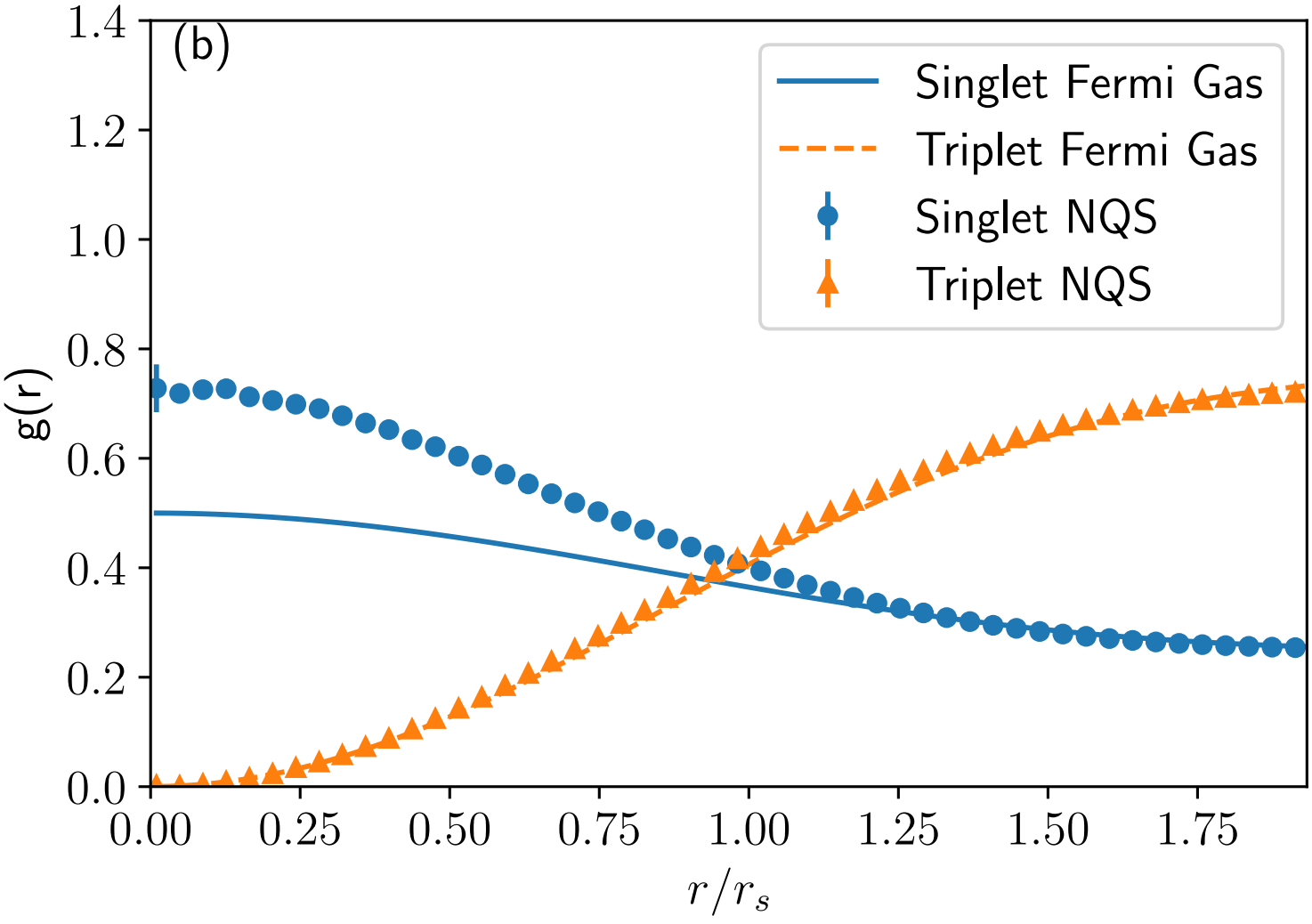
Dilute neutron star matter from neural-network quantum states by Fore et al, Physical Review Research 5, 033062 (2023) at density $\rho = 0.04$ fm^{-3}



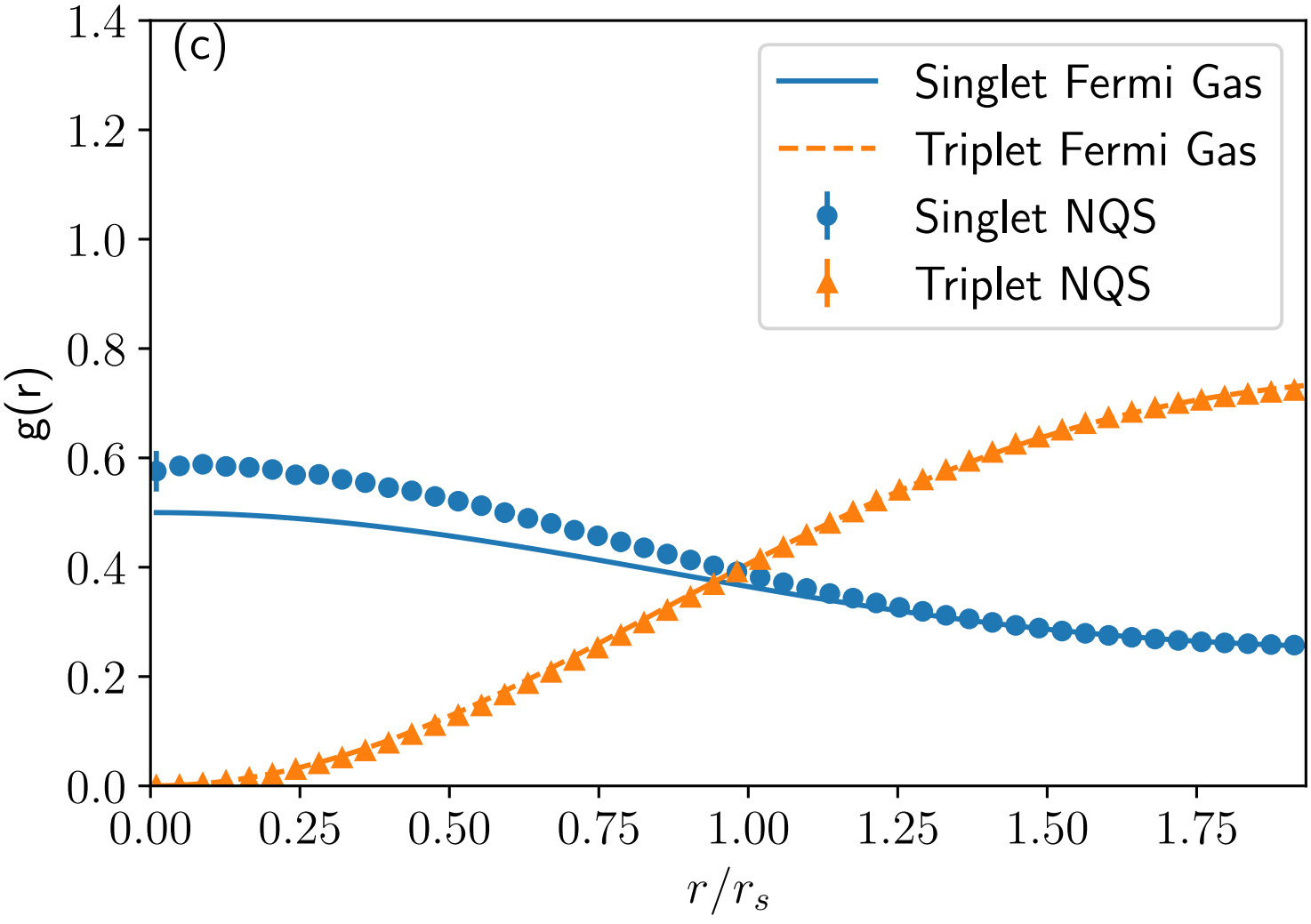
Pairing and Spin-singlet and triplet two-body distribution functions at $\rho = 0.01$ fm^{-3}



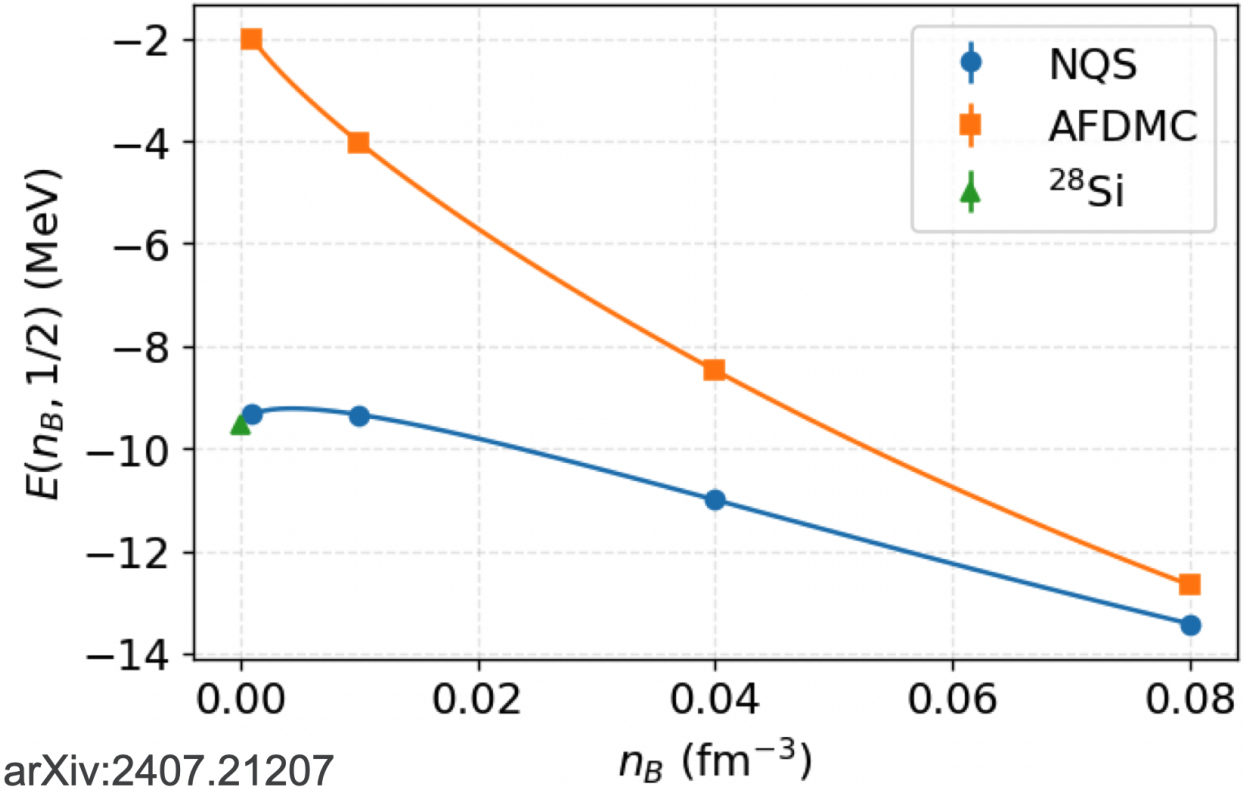
Pairing and Spin-singlet and triplet two-body distribution functions at $\rho = 0.04$ fm^{-3}



Pairing and Spin-singlet and triplet two-body distribution functions at $\rho = 0.08$ fm^{-3}

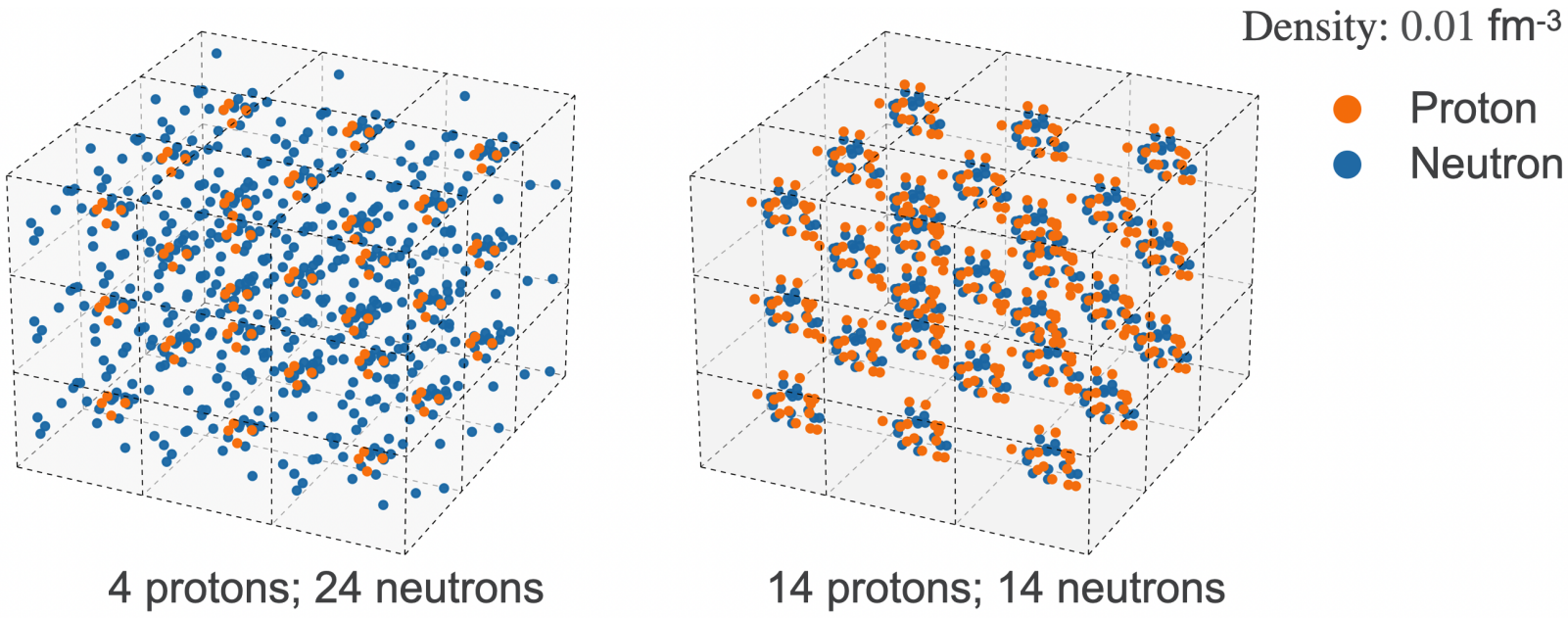


Symmetric nuclear matter



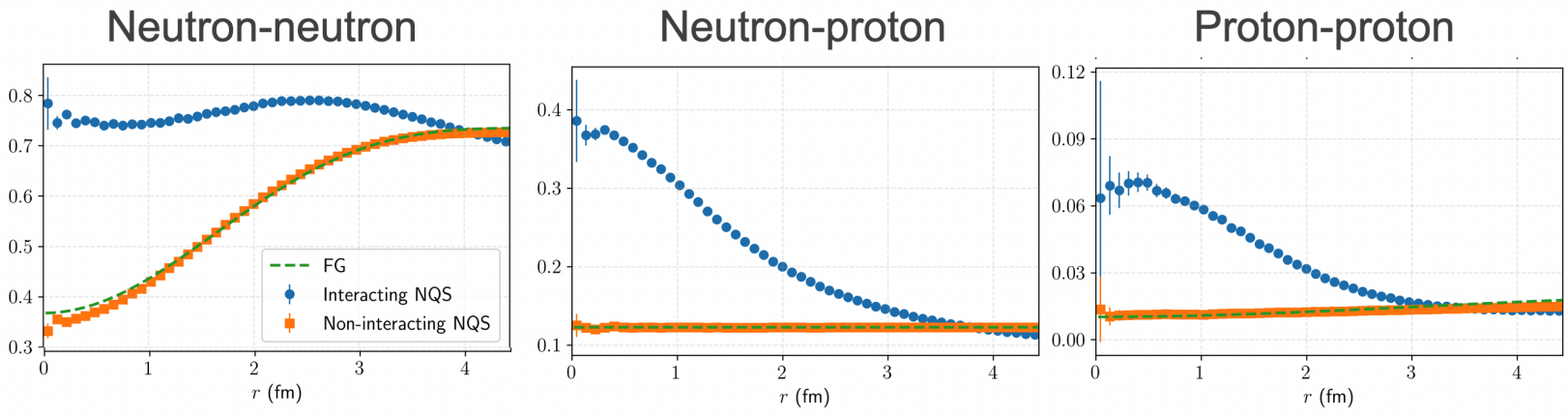
B. Fore, arXiv:2407.21207

Self-emerging clustering



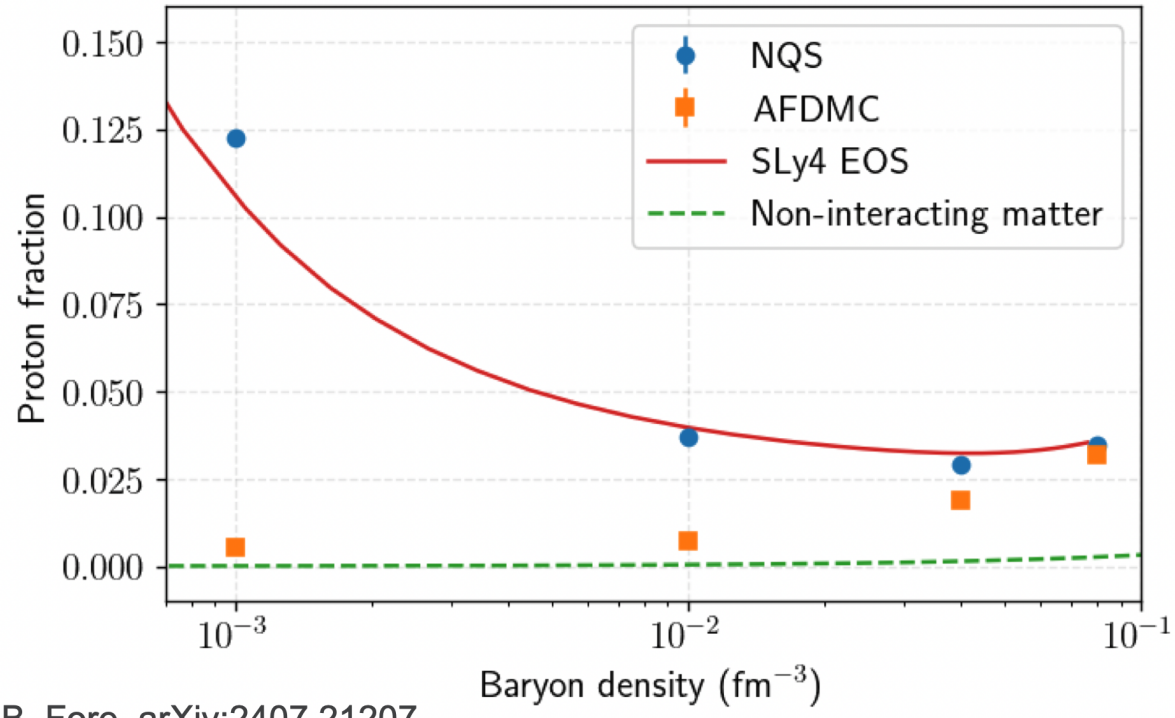
B. Fore, arXiv:2407.21207

Clustering: Two-body pair distributions



B. Fore, arXiv:2407.21207

Nuclear matter proton fraction

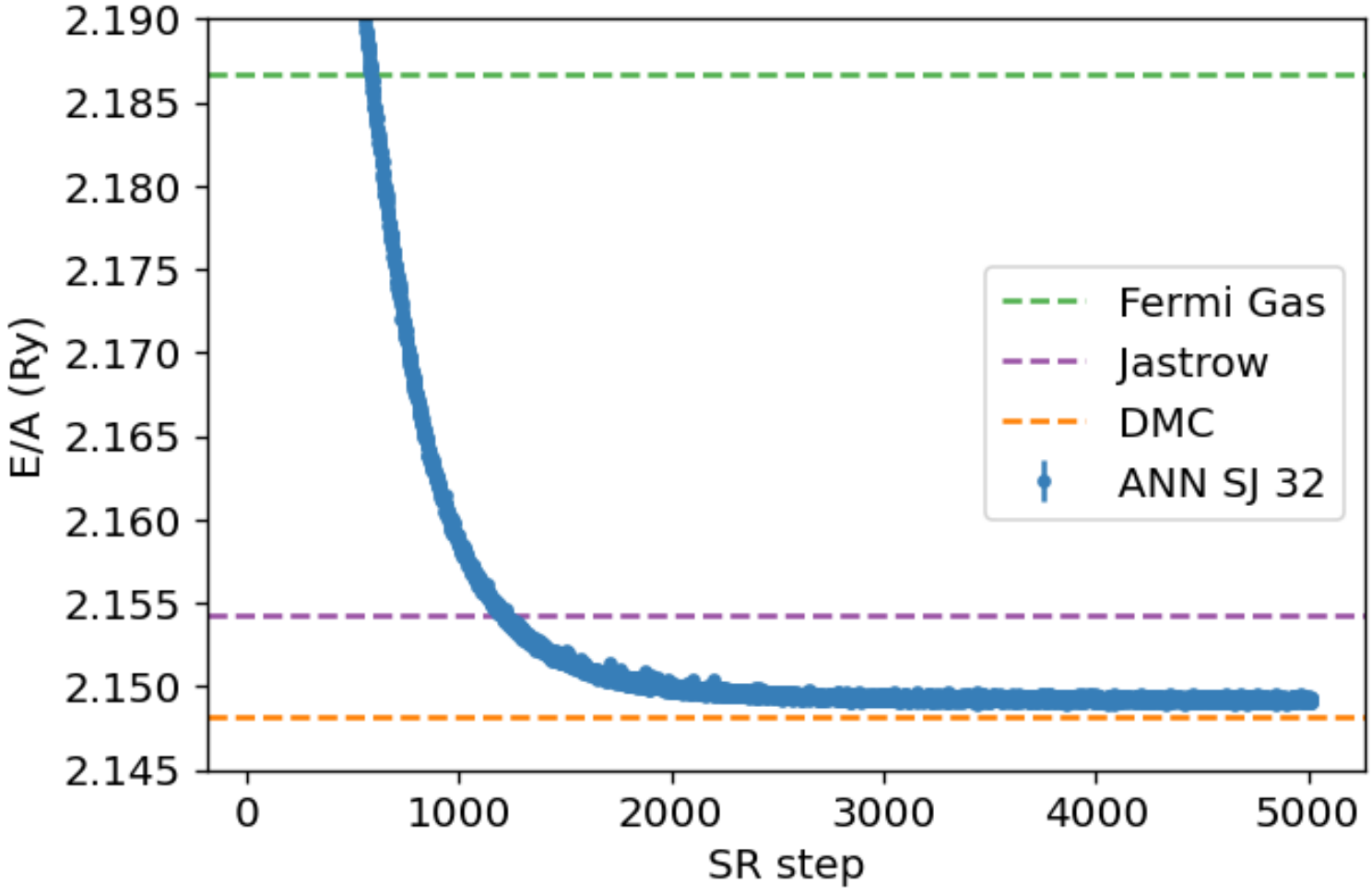


B. Fore, arXiv:2407.21207

Assumptions:

- Charge neutrality
  $n_p = n_e$
- Beta equilibrium
  $\mu_e = \mu_n - \mu_p$

The electron gas in three dimensions with $N = 14$ electrons (Wigner-Seitz radius $r_s = 2$ a.u.), Gabriel Pescia, Jane Kim et al. arXiv.2305.07240,



Efficient solutions of fermionic systems using artificial neural networks, Nordhagen et al, Frontiers in Physics 11, 2023

The Hamiltonian of the quantum dot is given by

$$\hat{H} = \hat{H}_0 + \hat{V},$$

where $\hat{H}_0$ is the many-body HO Hamiltonian, and $\hat{V}$ is the inter-electron Coulomb interactions. In dimensionless units,

$$\hat{V} = \sum_{i<j}^{N} \frac{1}{r_{ij}},$$

with $r_{ij} = \sqrt{\mathrm{r}_i^2 - \mathrm{r}_j^2}$.

Separable Hamiltonian with the relative motion part ($r_{ij} = r$)

$$\hat{H}_r = -\nabla_r^2 + \frac{1}{4}\omega^2 r^2 + \frac{1}{r},$$

Analytical solutions in two and three dimensions (M. Taut 1993 and 1994).

Generative models: Why Boltzmann machines?

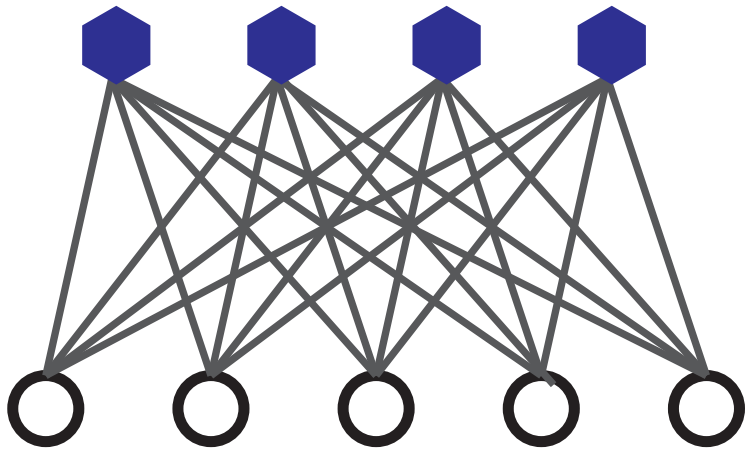
What is known as restricted Boltzmann Machines (RMB) have received a lot of attention lately. One of the major reasons is that they can be stacked layer-wise to build deep neural networks that capture complicated statistics.
The original RBMs had just one visible layer and a hidden layer, but recently so-called Gaussian-binary RBMs have gained quite some popularity in imaging since they are capable of modeling continuous data that are common to natural images.
Furthermore, they have been used to solve complicated quantum mechanical many-particle problems or classical statistical physics problems like the Ising and Potts classes of models.

The structure of the RBM network

Hidden Layer

Interactions

Visible Layer



$b_\mu(h_\mu)$

$W_{i\mu}v_i h_\mu$

$a_i(v_i)$

The network

The network layers:

1. A function $\boldsymbol{x}$ that represents the visible layer, a vector of M elements (nodes). This layer represents both what the RBM might be given as training input, and what we want it to be able to reconstruct. This might for example be the pixels of an image, the spin values of the Ising model, or coefficients representing speech.
2. The function $\boldsymbol{h}$ represents the hidden, or latent, layer. A vector of N elements (nodes). Also called "feature detectors".

Goals

The goal of the hidden layer is to increase the model's expressive power. We encode complex interactions between visible variables by introducing additional, hidden variables that interact with visible degrees of freedom in a simple manner, yet still reproduce the complex correlations between visible degrees in the data once marginalized over (integrated out).

The network parameters, to be optimized/learned:

1. $\boldsymbol{a}$ represents the visible bias, a vector of same length as $\boldsymbol{x}$.
2. $\boldsymbol{b}$ represents the hidden bias, a vector of same lenght as $\boldsymbol{h}$.
3. W represents the interaction weights, a matrix of size $M \times N$.

Joint distribution

The restricted Boltzmann machine is described by a Bolztmann distribution

$$P_{\mathrm{rbm}}(\boldsymbol{x}, \boldsymbol{h}) = \frac{1}{Z} \exp -E(\boldsymbol{x}, \boldsymbol{h}),$$

where Z is the normalization constant or partition function, defined as

$$Z = \int \int \exp -E(\boldsymbol{x}, \boldsymbol{h}) d\boldsymbol{x} d\boldsymbol{h}.$$

Note the absence of the inverse temperature in these equations.

Network Elements, the energy function

The function $E(\boldsymbol{x}, \boldsymbol{h})$ gives the **energy** of a configuration (pair of vectors) $(\boldsymbol{x}, \boldsymbol{h})$. The lower the energy of a configuration, the higher the probability of it. This function also depends on the parameters $\boldsymbol{a}$, $\boldsymbol{b}$ and W. Thus, when we adjust them during the learning procedure, we are adjusting the energy function to best fit our problem.

Defining different types of RBMs (Energy based models)

There are different variants of RBMs, and the differences lie in the types of visible and hidden units we choose as well as in the implementation of the energy function $E(\boldsymbol{x}, \boldsymbol{h})$. The connection between the nodes in the two layers is given by the weights w_{ij}.

Binary-Binary RBM:

RBMs were first developed using binary units in both the visible and hidden layer. The corresponding energy function is defined as follows:

$$E(\boldsymbol{x}, \boldsymbol{h}) = -\sum_{i}^{M} x_i a_i - \sum_{j}^{N} b_j h_j - \sum_{i,j}^{M,N} x_i w_{ij} h_j,$$

where the binary values taken on by the nodes are most commonly 0 and 1.

Gaussian binary

Gaussian-Binary RBM:

Another varient is the RBM where the visible units are Gaussian while the hidden units remain binary:

$$E(\boldsymbol{x}, \boldsymbol{h}) = \sum_{i}^{M} \frac{(x_i - a_i)^2}{2\sigma_i^2} - \sum_{j}^{N} b_j h_j - \sum_{i,j}^{M,N} \frac{x_i w_{ij} h_j}{\sigma_i^2}.$$

Representing the wave function

The wavefunction should be a probability amplitude depending on $\boldsymbol{x}$. The RBM model is given by the joint distribution of $\boldsymbol{x}$ and $\boldsymbol{h}$

$$P_{\mathrm{rbm}}(\boldsymbol{x}, \boldsymbol{h}) = \frac{1}{Z} \exp -E(\boldsymbol{x}, \boldsymbol{h}).$$

To find the marginal distribution of $\boldsymbol{x}$ we set:

$$P_{\mathrm{rbm}}(\boldsymbol{x}) = \frac{1}{Z} \sum_{\boldsymbol{h}} \exp -E(\boldsymbol{x}, \boldsymbol{h}).$$

Now this is what we use to represent the wave function, calling it a neural-network quantum state (NQS)

$$|\Psi(\boldsymbol{X})|^2 = P_{\mathrm{rbm}}(\boldsymbol{x}).$$

Define the cost function

Now we don't necessarily have training data (unless we generate it by using some other method). However, what we do have is the variational principle which allows us to obtain the ground state wave function by minimizing the expectation value of the energy of a trial wavefunction (corresponding to the untrained NQS). Similarly to the traditional variational Monte Carlo method then, it is the local energy we wish to minimize. The gradient to use for the stochastic gradient descent procedure is

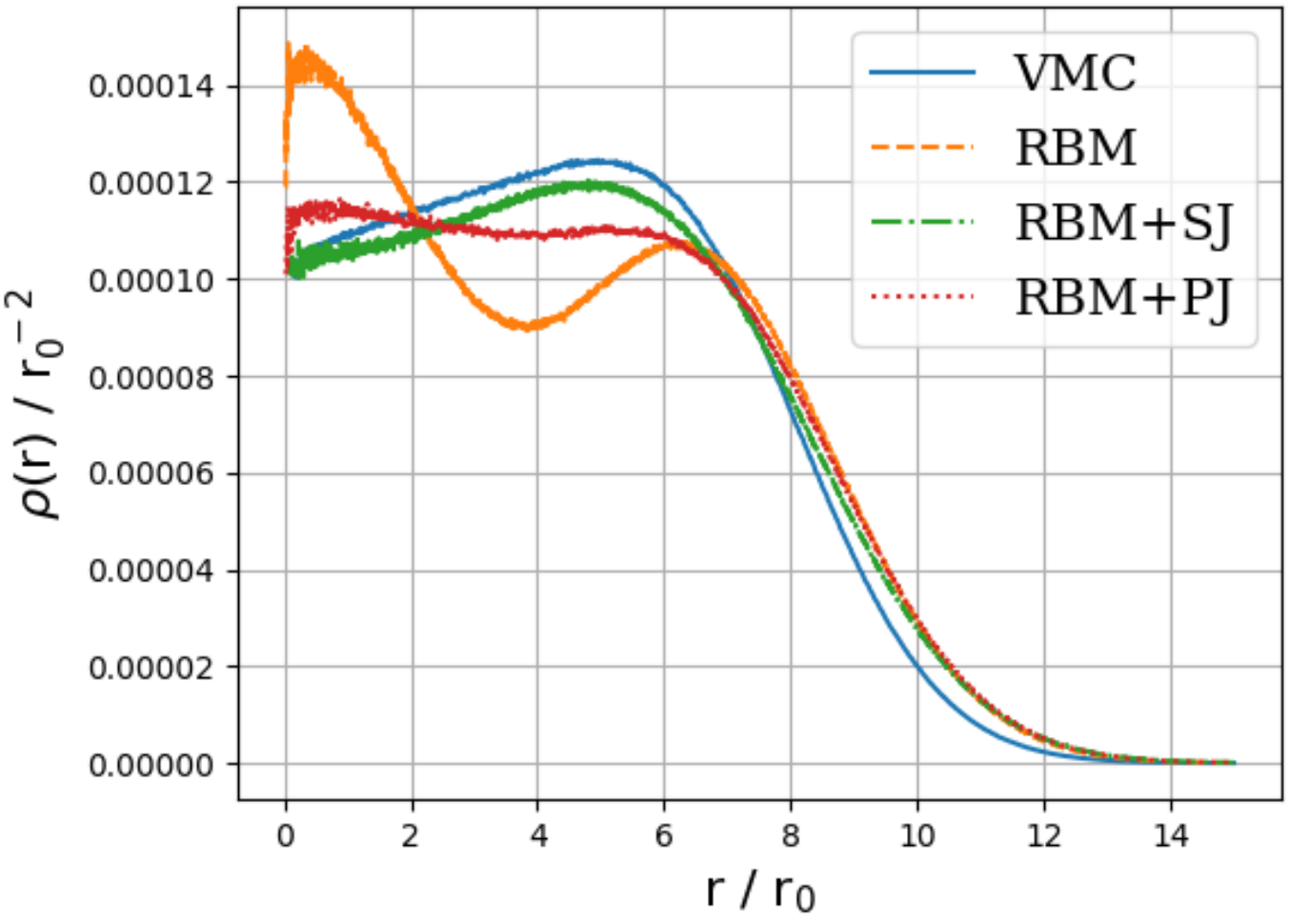
$$C_i = \frac{\partial \langle E_L \rangle}{\partial \theta_i} = 2(\langle E_L \frac{1}{\Psi} \frac{\partial \Psi}{\partial \theta_i} \rangle - \langle E_L \rangle \langle \frac{1}{\Psi} \frac{\partial \Psi}{\partial \theta_i} \rangle),$$

where the local energy is given by

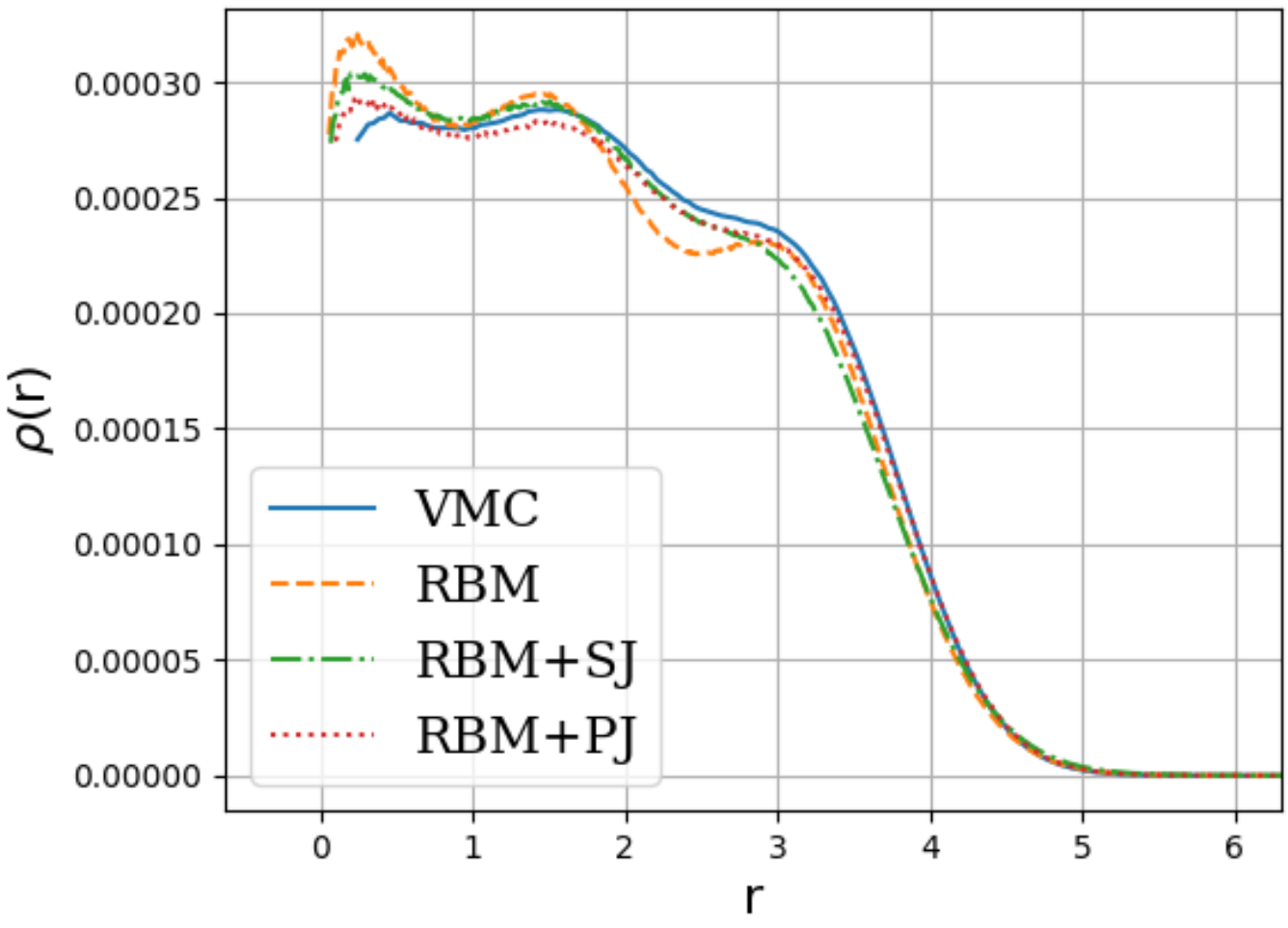
$$E_L = \frac{1}{\Psi} \hat{\boldsymbol{H}} \Psi.$$

Quantum dots and Boltzmann machines, onebody densities

$N = 6$, $\hbar\omega = 0.1$ a.u.



Onebody densities $N = 30$, $\hbar\omega = 1.0$ a.u.



Expectation values as functions of the oscillator frequency