Artificial neural network (ANN) simulations and control of quantum many-body dynamics in time-dependent control fields

Candidate Number: 1025496
Nov 15, 2019

Vacation Placement Information

I conducted my research project from June 24, 2019 to August 10, 2019 onsite at Princeton University, US and it is still ongoing now remotely. My supervisor is Professor Herschel Rabitz, and I worked very closely with Dr. Tak-San Ho, the senior scientist in Professor Rabitz's group.

1 Abstract

In this experiment, Artificial Neural Network (ANN) models are examined for its capability to simulate quantum many-body physics systems. In particular, ANN models such as Restricted Boltzmann Machine (RBM) and Feed Forward Neural Networks are tested and optimized for Ising model. The supremacy of ANN models compared to conventional numerical models are demonstrated up to 10 spin 1/2 particles.

2 Introduction

Simulation of molecular dynamics is simulating quantum many-body physics systems. It is therefore concerned with the efficient computation of the evolving wave packet whose dimensionality grows exponentially with the degrees of freedom. A challenge for molecular quantum dynamics calculations is the curse of dimensionality with respect to the degrees of freedom.

Following a recent paper by Carleo [1] in 2017, a method to solve quantum many-body problems using ANN with variational Monte Carlo calculations becomes a rising research topic in the field. It was shown that the quantum many-body ground state can be efficiently stored in the artificial neural networks, where the number of parameters used to represent the neural network is much smaller than the size of the Hilbert space.

In this paper, instead of ground state, all excited states are simulated, and the applied magnetic field in the Ising model is time-dependent, which is more of interest to the quantum optimal control research community.

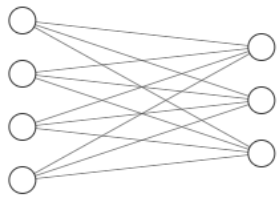
2.1 Ising model

The Ising model adopted is one of the most popular models for quantum many-body physics. In the model, a chain of spin 1/2 particles are coupled and evolve under magnetic fields. The specific Hamiltonian used is as follows.

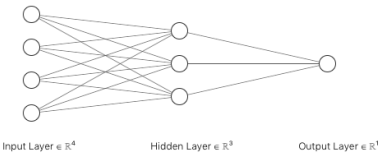
$$H(t) = B0 \sum_i Z_i + J \left(\sum_i Z_i \otimes Z_{i+1} + B1(t) \sum_i X_i \right) \tag{1}$$

$$where, X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix};\ Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix};\ Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix};\ I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

In Eq(1), i is summed over all spins. X and Z represent Pauli matrices. Assuming the chain of spins is along Z direction, the first term represents time-independent longitudinal magnetic field, the second term represents the coupling term, and the last term represents time-dependent horizontal magnetic field which serves as control field in optimal control theory.



(a) RBM with $N_{in} = 4, N_{hid} = 3$



(b) FFNN with $N_{in} = 4, N_{hid} = 3$

Figure 1: Models

3 Methodology

The central idea is that since the Schrodinger equations give the rate of change of the wave functions, which are the outputs of the neural networks, then the rate of change can also be obtained for the parameters of the neural networks by back propagation. Therefore, ideally the neural network evolves along with time and the goal is to have less degrees of freedom than the number of available quantum states.

After finding the ANN model to use, the simulation of the system consists of two major parts, one is the evolution of the neural network and the other is the choice of the initial states and conditions.

3.1 Benchmarking method: Direct integration

The Ising model evolves under time-dependent Schrodinger equation.

$$i\hbar\frac{\partial}{\partial t}|\Psi(t)\rangle = H(t)|\Psi(t)\rangle, \tag{2}$$

If assuming the initial wave function and the Hamiltonian are known, then it is simple integrating the Schrodinger equation for all possible states, which has a number of $2^{N_{in}}$, here N_{in} denotes the number of spins. To benchmark the ANN model, the integration method used for the direct integration method, also written as exact method, is 4th order Runge-Kutta method. This integration method calculates the derivative for 4 times at every time step, giving a result of 4th order precision.

3.2 ANN models

Artificial Neural Networks are in essence a mapping from the input neurons to the output neuron. Assuming the parameters of the neural network such as the weights, bias, and activation functions are Ω, and the input neurons are represents as ν.

$$|\Psi(t)\rangle = f(\boldsymbol{\nu}; \Omega(t)) \tag{3}$$

Here the problem is dealing with a system of discrete quantum states $|\nu_1, \cdots, \nu_n\rangle$, the wave function can be represented as follows.

$$|\Psi(t)\rangle = \sum_{\nu_1} \cdots \sum_{\nu_N} |\nu_1, \cdots, \nu_n\rangle\langle\nu_1, \cdots, \nu_N|\Psi(t)\rangle, \tag{4}$$

where the input neurons $\boldsymbol{\nu}$ take values of ± 1. In the following, two ANN models, RBM and FFNN, are introduced.

3.2.1 Restricted Boltzmann Machine

For the RBM model, an example for 4 spins and 3 hidden neurons is given in Fig(1a). The output wavefunction is given by the following summation over all possible configurations where the value of each hidden neuron h_i take on ± 1 in convention.

$$\Psi_{RBM} = \sum_{h_i} exp(\sum_j a_j\nu_j + \sum_i b_i h_i + \sum_{ij} W_{ij}h_i\nu_j) \tag{5}$$

3.2.2 Feed Forward Neural Network

For the FFNN model, by far only one hidden layers has been implemented, and the resulting network is similar to Fig (1b). Since we are using complex network, we only use one output to represent the complex value wavefunction, so there is only one output neuron in the output layer. The wave function is given as the following equation where f_{hid} and f_{out} are the two activation functions at the hidden and output layers.

$$\Psi_{FFNN}(t) = f_{out}(\sum_i W_i^{(2)} f_{hid}(\sum_j W_{ij}^{(1)} v_j + b_i^{(1)})) \tag{6}$$

Noting the values of the neural network are complex, all activation function should be analytic in order to have derivatives. Commonly used activation functions and their corresponding derivatives are as follows.

$$f_1(z) = e^z, \frac{df_1(z)}{dz} = e^z \tag{7}$$

$$f_2(z) = z, \frac{df_2(z)}{dz} = 1 \tag{8}$$

$$f_3(z) = \frac{e^z}{1+e^z}, \frac{df_3(z)}{dz} = \frac{e^z}{(1+e^z)^2} \tag{9}$$

$$f_4(z) = \frac{z}{1+|z|}, \frac{df_4(z)}{dz} = \frac{1+|z|/2}{(1+|z|)^2} \tag{10}$$

$$f_5(z) = log(cosh(z)), \frac{df_5(z)}{dz} = tanh(z) \tag{11}$$

3.3 Evolution of neural network parameters

As a result of the Dirac-Frenkel-McLachlan time-dependent varia-tional principle and least action principle, the evolution of the neural network parameters Ω can be expressed as follows. Detailed derivations can be seen in J. Haegema [2].

$$\sum_{k=1}^{M} \left\langle \frac{\partial \Psi_{\text{ANN}}}{\partial \Omega_j} \middle| \frac{\partial \Psi_{\text{ANN}}}{\partial \Omega_k} \right\rangle \times \frac{\partial}{\partial t}\Omega_k(t) = -\frac{i}{\hbar} \left\langle \frac{\partial \Psi_{\text{ANN}}}{\partial \Omega_j} \middle| H \middle| \Psi_{\text{ANN}} \right\rangle, \ j = 1, \cdots, M, \tag{12}$$

with

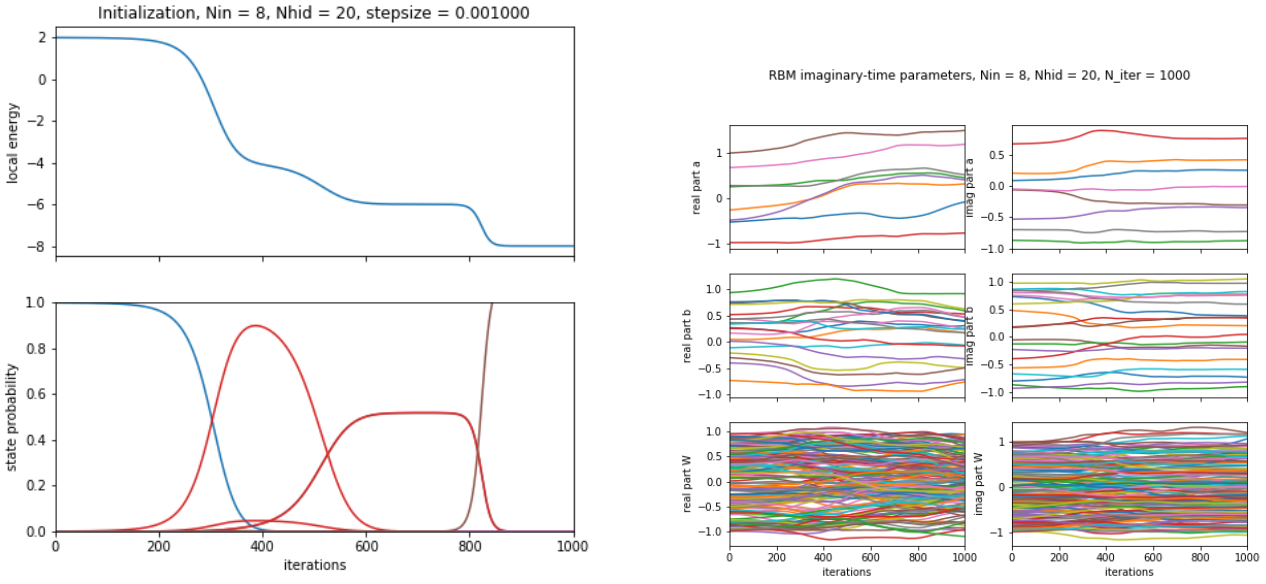
$$\frac{\partial |\Psi_{\text{ANN}}(\mathbf{\Omega}(t))\rangle}{\partial \Omega_k} = \begin{pmatrix} \frac{\partial \Psi_{\text{ANN}}\left(\nu_1^1, \cdots, \nu_N^1, \mathbf{\Omega}(t)\right)}{\partial \Omega_k} \\ \frac{\partial \Psi_{\text{ANN}}\left(\nu_1^2, \cdots, \nu_N^2, \mathbf{\Omega}(t)\right)}{\partial \Omega_k} \\ \vdots \\ \frac{\partial \Psi_{\text{ANN}}\left(\nu_1^K, \cdots, \nu_N^K, \mathbf{\Omega}(t)\right)}{\partial \Omega_k} \end{pmatrix}_{K \times 1},$$

Here, to simplify the expression, the rate of change for the parameters can also be written as,

$$\frac{\partial}{\partial t}\Omega_k(t) = -\frac{i}{\hbar} S^{-1} F \tag{13}$$

Since in most cases, S is ill-conditioned, a few methods are available for regularization such as singular value truncation. In the current version, Tikhonov regularization is applied by,

$$\sigma = SVD(S) \quad and \quad \sigma_{ii} = \frac{\sigma_{ii}}{\sigma_{ii}^2 + \lambda^2}, \text{here } \lambda \text{ is a hyper parameter subject to tuning} \tag{14}$$



(a) Local energy and state probabilities evolution

(b) Neural network parameters evolution

Figure 2: Imaginary time evolution for initialization, $N_{in} = 8, N_{hid} = 20, RBM$

3.4 Initialization

Initialization refers to the methods of assigning initial values to the neural network parameters, which also determines the initial wave function Ψ_0. Since the dimensionality of the model is expected to be less than the degrees of freedom, it is reasonable that the output space is not the full Hilbert space. Therefore, if the initial condition is not chosen cautiously, it is very likely that the model will blow up or become inconsistent with the exact method.

By far, the state-of-the-art is the ground-state initialization method. In this approach, the neural network is initialized such that the wave function is the ground state of the Hamiltonian $H(0)$. Since it can be time consuming to find the eigenvalues and eigenstates of the Hamiltonian when the number of spins is large, imaginary time evolution is used here.

When under constant Hamiltonian, the time dependent wave functions can usually be expressed as,

$$|\Psi(t)\rangle = |\Psi(0)\rangle e^{-i\frac{E}{\hbar}t} \tag{15}$$

Then, if the time evolves as an imaginary value, the obtained wave function will be that of the ground state, defined as the least energy state. Thus, the initialization is done by evolving the neural networks according to Eq(13) until the energy stabilizes to the ground state energy.

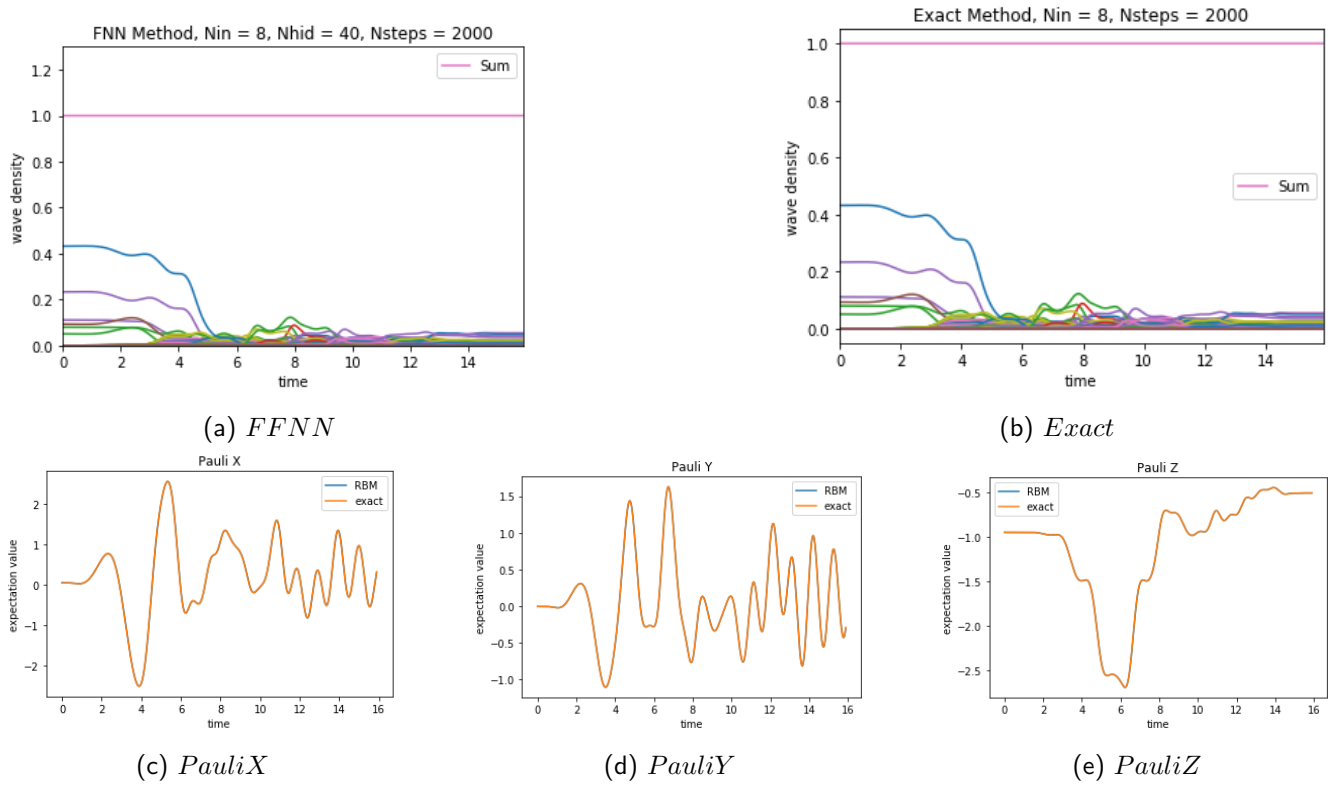
4 Results

For an operator $\hat{O}$, the expected value is calculated as,

$$\left\langle \hat{O} \right\rangle = \frac{\left\langle \Psi_{\text{ANN}} \left| \hat{O} \right| \Psi_{\text{ANN}} \right\rangle}{\left\langle \Psi_{\text{ANN}} \right| \left. \Psi_{\text{ANN}} \right\rangle} \tag{16}$$

For the initialization process using imaginary time evolution, the expected value of the Hamiltonian operator is calculated and used as the local energy. Some plots for this process are shown in Fig(2).

In the upper plot of Fig(2a), the local energy clearly decreases and eventually reaches the minimum or ground state energy. The lower plot of Fig(2a) gives a plot of the state probabilities for all $2^8 = 256$ possible quantum states. From the graph it is shown that most of the states decay to zero with only some of the degenerate ground states having non-zero probabilities. The left and right hand side of Fig(2b) shows the evolution of the real and imaginary parts of the neural network parameters Ω respectively.



(a) $FFNN$ (b) $Exact$

(c) $PauliX$ (d) $PauliY$ (e) $PauliZ$

Figure 3: FFNN evolution process, $N_{in} = 8, N_{hid} = 40, FFNN$

For the ANN evolution process, not only the plot of the ANN process is compared to the benchmarking method of direct integration, the expected value for the spins along x, y, and z directions are calculated as indicators of the consistency. The expected values of the spins are obtained from the expected value of the Pauli matrices.

For FFNN, some example plots are in Fig(3) when the activation functions of f_1 in Eq(7) is used for for hidden layer and f_2 in Eq(8) is used for output layer. It can be seen from Fig(3a) and Fig(3b) that the visualized state probabilities fits well. The plot in Fig(3c), Fig(3d), and Fig(3e) confirms that as well. In the latter three graphs, the blue line for FFNN and yellow line for exact method is perfectly aligned with each other, indicating a great fit between the two.

For RBM model, the results are plotted in Fig(4) below. From the graphs it can be said that the RBM model also approximates the direct integration method well. The difference of the two methods can be seen from the expectation values in Fig(4c), Fig(4d), and Fig(4e). In this specific case, the wave function of the RBM model had some fluctuation at around t = 1, which can be seen from Fig(4a) that the sum of all quantum state probabilities doesn't equal one anymore. The reason for these cases are usually the need to inverse ill-conditioned matrices. However, since the expected value is still consistent, this issue doesn't harm the representative power of the model since it can be solved easily by performing normalization at every time step.

5 Analysis

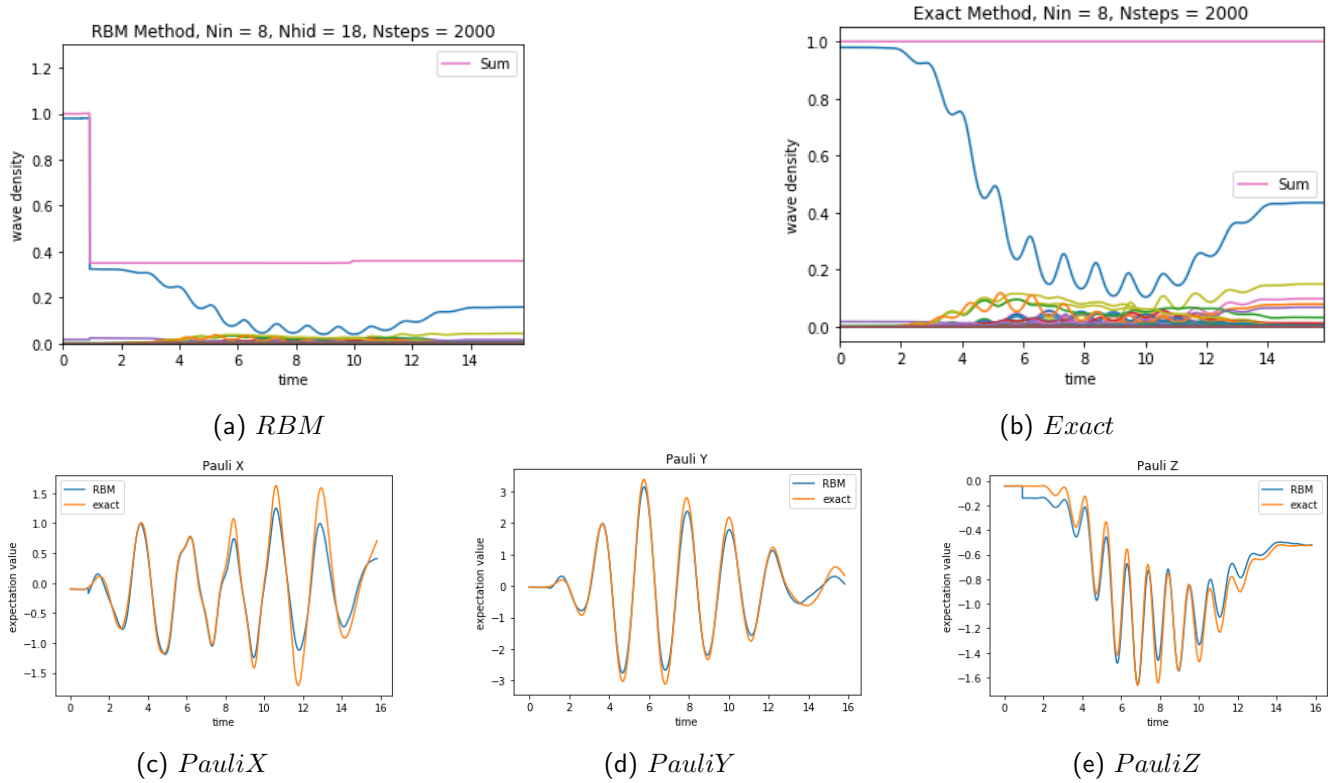
Overall, the ANN models approximate the time evolution of the Ising model to a satisfactory extent. For the tests performed on up to 10 spins by far, the degrees of freedom in the ANN networks is usually much smaller than the dimension of the Hilbert space. For example, in the case of RBM plotted in Fig(4), the degree of freedom is $N_{in} + N_{hid} + N_{in} \times N_{hid} = 170$, while the simulated system has $2^8 = 256$ dimensions.

1. Fit

The fit between our model and exact method usually matches very well in the starting parts, but gradually deviate from each other as time goes on.

2. Models

The FFNN method requires more hidden neurons to have stable solution. However, as long as FFNN has stable solution the fit between FFNN and exact solution is apparently much better than RBM method.



(a) RBM (b) $Exact$

(c) $PauliX$ (d) $PauliY$ (e) $PauliZ$

Figure 4: RBM evolution process, $N_{in} = 8, N_{hid} = 18, RBM$

One thing to note is that we are only using one effective activation function for FFNN, because other combinations doesn't work by far.

The main deficit of the simulations is that only less than 10 spins have been simulated yet, which is not enough to find the complexity or representative powers of the model yet. One of the obstacles is that the computing device used has very limited computing power. Another major facing is the blow up issue, which comes from the fact that at one point the tangent space of the neural network isn't orthogonal enough to reach the desired direction for the change of network parameters. Since the ideal neural network has less degrees of freedom than the number of possible states, it is understandable that the process can't be exact. Beisdes, no matter for all the regularization methods or pseudo-inverse method, the idea is only to minize the error of the inversion. Therefore, it is expected that as the evolution proceeds, the method would deviate from the actual evolution to some extent.

5.1 Future improvements

As far as I can see, the main obstacles to be overcome are:

1. Computational resource. Finding a bigger cluster or implement the code using parallelization and GPU should help increase the speed of testing.

2. Model sensitivity to initial conditions. While ground state is achieved, with different starting points in the imaginary time evolution process, the neural network can still have different initial parameters, which has non-trivial impacts.

3. Effective measure of comparison. Currently the comparison is mainly done by visualizing the wave functions and expected values for spins. However, to do effecient and extensive testing and benchmarking, a new measure should be invented for the expectation of Pauli matrices such that it can be determined quantitatively if the model is a decent fit.

4. Choosing good tangent space vectors throughout. Here, good means as orthogonal as possible.

5. The literature in complex activation functions and neural networks is lacking, and there is a possibility that it works differently.

5.2 Potential directions

1. Time-dependent input neurons. Now the input neurons only take on values of ± 1, which is constant throughout. According to the adiabatic approximation in M. V. Berry [3], the states driven by $H_j(t)$ would take on the expression below. Therefore, it might be possible to add time dependence in the input neuron amplitudes to smooth out the time dependence of the neural network parameters.

$$|\psi_j(t)\rangle = \exp\left(-\frac{i}{\hbar}\int_0^t \nu_j(t')dt' - \int_0^t \langle \nu_j(t')|\partial_{t'}\nu_j(t')\rangle dt'\right)|\nu_j(t)\rangle. \tag{17}$$

2. Developing multi-layer FFNN with different combinations of activation functions. Historically, the FFNN or even the machine learning field is of interest to practical use only after deep learning is found, which introduces very deep neural nets.

3. Using time t as one of the input neuron, which basically turns the problem into a big data optimization problem.

4. Pseudo-spectral method as suggested by Dr Tak-San Ho, using orthogonal polynomials such as the time-dependent Chebyshev polynomials and constant weights to represent the neural network parameters. As a result, optimize the weights based on the cost function defined by the Schrodinger equation.

5. According to K. Ido [3], an implementation of Markov Chain Monte Carlo sampling can be used for larger number of spins to sample from the state space and approximate the parameters' gradients with reasonable amount of time.

6 Conclusion

We have simulated the Ising model up to 10 spins using Artifical Neural Network models, and specifically demonstrated the simulation with Restricted Boltzmann Machine and Feed Forward Neural Network. The supermacy is demonstrated such that the dimension of the problem is reduced from exponential to ideally polynomial. With the implementation of Monte Carlo Sampling and other techniques, we hope to demonstrate for extensive number of spins and also other physical models. Besides our results being directly applicable to the optimal control experiments, the efficiency for simulating and controling many-body systems can be increased by ANN mdoels, which is going to be critical for the related physics and chemistry community.

References

[1] Carleo G, Troyer M. Solving the quantum many-body problem with ar- tificial neural networks. Science 2017;355(6325):6026. http://dx.doi.org/ 10.1126/science.aag2302, URL http://science.sciencemag.org/content/355/6325/602.

[2] J. Haegeman, J. I. Cirac, T. J. Osborne, I. Piorn, H. Verschelde, F. Verstraete, Phys. Rev. Lett., 2011, 107, 070601.

[3] K.Ido, T.Ohgoe, M.Imada, Time-dependentmany-variable variational Monte Carlo method for non equilibrium strongly correlated electron systems. Phys. Rev. B 92, 245106 (2015).