Communications in Computational Chemistry
Commun. Comput. Chem., Vol. 7, No. 2, 120–126
doi: 10.4208/cicc.2025.92.01
Regular Article

Evaluation of Phase Networks in Transformer-Based Neural Network Quantum States

Lizhong Fu[1], Honghui Shang[1,*] and Jinlong Yang[1,2,*]

[1]*State Key Laboratory of Precision and Intelligent Chemistry, University of Science and Technology of China, Hefei, Anhui 230026, China;*

[2]*Hefei National Laboratory, University of Science and Technology of China, Hefei 230088, China.*

[*] Corresponding authors: shanghui.ustc@gmail.com; jlyang@ustc.edu.cn

Received on 02 April 2025; Accepted on 25 April 2025

Abstract: Neural network quantum states represent a powerful approach for solving electronic structures in strongly correlated molecular and material systems. For a neural network ansatz to be accurate, it must effectively learn the phase of a complex wave function. In this work, we demonstrate several different network structures as the phase network for a Transformer-based neural network quantum state implementation. We compare the accuracy of ground state energies, the number of parameters, and computational time across several small molecules. Furthermore, we propose a phase network setup that combines cross attention and multilayer perceptron structures, with the number of parameters remaining constant across different systems. Such an architecture may help reduce computational costs and enable transfer learning to larger quantum chemical systems.

Key words: neural network quantum states, phase network, electronic structure calculation.

1. Introduction

Solving the electronic structure of correlated molecular and material systems has long been an essential task in computational chemistry. In these systems, mean-field theories such as density functional theory (DFT) and the Hartree-Fock (HF) self-consistent field method fail to accurately describe the correlated electronic wave function, while the theoretically accurate full configuration interaction (FCI) method requires computational resources that scale exponentially with system size, making its application to large systems impractical.

The past decade has faced an explosive growth of applications of neural networks in different fields, where it has demonstrated remarkable ability in representing complicated functions encountered in various situations. In the context of computational many-body problems, this expressive power has been leveraged to represent the highly correlated electronic wave function. This approach, introduced in 2017 by Carleo and Troyer, is known as neural network quantum states (NNQS) [1]. NNQS methods have been applied to both spin and Fermionic models, such as the J1-J2 Heisenberg model and the Hubbard model [2–3]. In the context of computational chemistry, there have also been demonstrations in small molecular and material systems. Some of these works feature a real-space representation of the electronic wave function, in which a neural network takes the coordinates of electrons $(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \dots)$ as input, and outputs the wave function $\Psi(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \dots)$. Notable implementations of this method include FermiNet [4], PauliNet [5], LapNet [6], and DeepErwin [7]. Other works adopt a second-quantization formulation of the electronic structure problem, expressing both the wave function and the Hamiltonian in the occupation number representation over a given set of single-particle orbitals [8–9]. These methods introduce a basis set that enables direct comparison with standard quantum chemistry approaches and allows for systematic accuracy improvements by selecting increasingly comprehensive basis sets. However, the basis set approximation introduces an error that is absent in the real-space representation of the wave function.

One essential difference between NNQS and other applications of neural networks is that the electronic wave function is complex-valued, requiring a complex-valued network, whereas most deep learning applications focus on real-valued network outputs. This results in a lack of complex-valued network designs and limits the available choices for NNQS implementations. In many NNQS methods, this issue is circumvented by parameterizing the complex wave function using two separate real-valued networks: one for the amplitude and one for the phase. It has been shown in spin systems that learning the phase of the wave function is a challenging task for the network, and providing a reasonable initial guess based on the sign rule dramatically improves convergence behavior [2]. However, in quantum chemistry systems, no such simple rule exists for the phase, leaving it entirely up to the phase network to find the ground state. Therefore, choosing an appropriate structure for the phase network can improve both the energy landscape and the expressive ability of the NNQS wave function, leading to a faster convergence to the true wave function.

In this work, we compare several different implementations of phase network in QiankunNet, a transformer based NNQS method [10]. We compare ground state energy results on 17 typical small molecules. We also compare the number of parameters and time costs for each of these structures, in searching for a phase network structure that is both numerically accurate and computationally efficient.

2. Theoretical method

2.1 Transformer-based neural network quantum states

In computational chemistry, the wave function of a correlated system $|\Psi\rangle$ can be expressed as a state vector consisting of 2^n complex coefficients, where n is the number of spin orbitals used to define the system. NNQS method employs a neural network to represent such a state vector. The network takes the electron configuration on spin orbitals as input and returns the corresponding coefficients for each configuration. These coefficients are then used to calculate expectation values of operators, including the Hamiltonian. During a training process, one first sample a batch of configurations, then calculate the energy expectation values. The energy estimator serves as a loss function, whose gradient is calculated and used to update parameters in the network. Specifically, in our implementation, the wave function ansatz is of the following form:

$$\Psi_\theta = A_\theta e^{i\phi_\theta}$$

Where A_θ is the network representing amplitude of the wave function. We use GPT-style decoder-like layers for the amplitude network. Such a structure not only shows ability to capture long-range dependencies, but also enables the adoption of autoregressive sampling. Compared to classical Malkov Chain Monte Carlo (MCMC) sampling, the BAS algorithm has been shown to accelerate the sampling process significantly, thereby facilitating application of NNQS methods to larger systems [11-12]. The ϕ_θ part of the wave function ansatz is a network specifically designed to represent the phase of the wave function, which is the primary focus in this work.

It is worth noting that for systems with open boundary conditions, such as molecules, the electronic wave function can be treated as a real-valued function. In this case, the phase term $e^{i\phi_\theta}$ reduces to a simple ±1 sign before each amplitude coefficient. This raises the question of whether it is possible to directly represent the sign instead of the full phase, or even absorb the sign into the amplitude using a single real-valued network. However, directly optimizing the sign is challenging due to its inherently combinatorial nature. Moreover, absorbing the sign into the amplitude network leads to a rugged energy landscape, as the network must pass through zero to switch signs—potentially creating energy barriers that hinder optimization. Maintaining a separate phase network also enables extension to systems with periodic boundary conditions, where the wave function must remain genuinely complex-valued.

2.2 Phase network structures

In previous work, we used a simple multilayer perceptron (MLP) as the phase network. Such a structure consists only of fully-connected layers (FC) and can be expressed as

$$\log \phi = A_n(\cdots (A_2(\mathrm{A}_1 x + \mathrm{b}_1) + b_2) \cdots) + b_n$$

in which A_i, b_i are learnable parameters, and x is the input configuration. 4 hidden layers and a hidden layer dimension of 512 is used, thus the total number of trainable parameters is approximately $789\mathrm{k} + 512 \times n$, where n is the number of spin orbitals. This network structure is depicted in **Figure 1(a)**.

Since the decoder-like amplitude network has only about 50,000 parameters, it raises the question of whether such a large number of parameters in the phase network is truly necessary. To address this, we test three alternative approaches that replace the MLP with networks containing fewer parameters. The first approach is directly inspired by the amplitude network. We use several encoder layers to represent the phase, as illustrated in **Figure 1(b)**. In an encoder layer, the learnable parameters are located in the position-wise fully connected layers of dimension $n_{\mathrm{embedding}} \times n_{\mathrm{embedding}}$, as well as the position-wise feed forward network (FFN) [13]. By choosing embedding dimensions and FFN hidden dimensions smaller than the MLP hidden dimension (512), this structure can have significantly fewer parameters than an MLP. In our amplitude network implementation, the attention embedding dimensions are taken to be 32 or 48, while the FFN hidden dimension is 128. This suggests that similar hyperparameters may be also applicable in the case of phase dimensions.

Another approach is to use decoder layers as the phase ansatz. A typical decoder layer in the Transformer architecture consists of a self-attention layer, a cross attention layer, and a position-wise fully connected network. Since our task does not involve "transforming" one sequence into another, there are no encoder outputs available to serve as queries for the cross attention layer. Therefore, we use an array of zeros, with the same length as the input sequence, as the query for the cross attention layer. This structure is implemented directly using torch.nn.TransformerDecoderLayer, and a depiction can be found in **Figure 1(c)**. It is important to note that this phase network—referred to as "decoder" layers below and in the figures—is not the same as the decoder-like amplitude network. The amplitude network consists of GPT-style decoder-like Transformer layers, which, although commonly referred to as "decoder layers" in many contexts, are essentially encoder layers with masked self-attention modules.

The third method is inspired by applications in multimodal learning. In a cross attention mechanism, the length of the output sequence matches that of the query sequence, which does not necessarily have to be the same as the key and value sequences. In the