First principles physics-informed neural network for quantum wavefunctions and eigenvalue surfaces

Marios Mattheakis[1]*, **Gabriel R. Schleder**[1], **Daniel T. Larson**[2], **Efthimios Kaxiras**[1,2]
[1]John A. Paulson School of Engineering and Applied Sciences, Harvard University
[2]Department of Physics, Harvard University
{mariosmat, gschleder, dtlarson, kaxiras}@g.harvard.edu,

Abstract

Physics-informed neural networks have been widely applied to learn general parametric solutions of differential equations. Here, we propose a neural network to discover parametric eigenvalue and eigenfunction surfaces of quantum systems. We apply our method to solve the hydrogen molecular ion. This is an *ab initio* deep learning method that solves the Schrödinger equation with the Coulomb potential yielding realistic wavefunctions that include a cusp at the ion positions. The neural solutions are continuous and differentiable functions of the interatomic distance and their derivatives are analytically calculated by applying automatic differentiation. Such a parametric and analytical form of the solutions is useful for further calculations such as the determination of force fields.

1 Introduction

Physics-informed neural networks (PINNs) have been widely applied recently to study various kinds of differential equations [9]. Supervised PINNs can be trained on data to learn nonlinear differential operators [14], discover differential equations [21, 20], and solve inverse problems [1, 4, 18]. Unsupervised PINNs can be trained without using any labeled data to discover analytical and differentiable solutions of ordinary [12, 16] or partial differential equations (PDEs) [22, 11], and eigenvalue problems [7, 19, 13, 8].

In this work we introduce a novel PINN architecture to learn the quantum mechanical wavefunctions for electrons in molecules. This approach can obtain continuous potential energy surfaces and the associated parametric wavefunctions. In the physics of solids and molecules, the Coulomb potential plays a crucial role since it represents the basic interaction between the system's components, electrons and ions. This potential is characterized by singularities which yield cusps in the wavefunction solutions of the Schrödinger equation. Standard numerical approaches consider pseudo-potentials to avoid this issue by employing an effective potential of the ion screened by core electrons. Although this is an efficient approach for numerical calculations, this approximation requires the careful choice and generation of effective potentials to provide consistent wavefunctions and energies; it also omits the treatment of core electrons, which in some applications are important. Here we show that PINNs can solve the real Coulomb potential in a simple model, the hydrogen molecular ion H_2^+, thus enabling *ab initio* calculations for simple molecules. We expect that the method will be generalizable to more complicated cases, which is the subject of ongoing work. By exploiting the property of NN's to learn general representations, we obtain continuous and analytically differentiable eigenvalues and wavefunctions that depend on the interatomic distance, which make it feasible to obtain derivatives of the energy landscape; these derivatives are related to physical observables like forces and vibrational frequencies.

*https://scholar.harvard.edu/marios_matthaiakis

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

Main contributions: i) We propose a novel PINN architecture for solving the Schrödinger equation for the H_2^+ ion, including the singular Coulomb potential. Functions including cusps are explicitly included by embedding the approximate Linear Combination of Atomic Orbitals (LCAO) solution. ii) The network computes continuous and analytically differentiable wavefunction and eigenvalue surfaces for a range of interatomic distances. Being analytical, these solutions can be used to compute additional physical properties like forces and vibrational frequencies. iii) We obtain accurate solutions with fast training by embedding physics-inspired features in the solution like inversion symmetry, and by incorporating the asymptotic behavior through the LCAO approximation.

2 Related work

The optimization of PINNs is achieved by minimizing a loss function constructed by differential equations that encode the physical principles of a system. Specifically, the loss function may consist of an equation-driven and a data-driven component [20, 9, 16]. The former depends only on the neural solutions and their derivatives with respect to the inputs; the derivatives are calculated using automatic differentiation (autograd). The latter can compare neural predictions to ground truth data [20] or impose physical laws [16].

Recently, PINNs have been employed to solve quantum eigenvalue problems formulated by the stationary Schrödinger equation $\hat{\mathcal{H}}\psi = E\psi$, where $\hat{\mathcal{H}}$ is a known Hamiltonian operator and ψ, E are, respectively, the unknown wavefunction and energy we seek to obtain. Typically there are two main deep learning approaches that have been used to solve the Schrödinger equation [3, 15, 6]. The method introduced in Refs. [7, 8] considers a PINN that predicts both ψ and E. By minimizing a loss function to satisfy the Schrödinger PDE, the NN approximately calculates eigen-solutions. The second approach [19, 13] is based on the variational principle. The NN returns only ψ, from which the eigen-energy is computed as the expectation value $\langle\hat{\mathcal{H}}\rangle = \langle\psi|\hat{\mathcal{H}}|\psi\rangle/\langle\psi|\psi\rangle$ (in Dirac notation), which is minimized to optimize the NN. In this study we adopt the method used in Ref. [7, 8]. We generalize this deep learning approach to obtain accurate generalized parametric eigen-solutions, namely ψ and E as smooth and differentiable functions of the interatomic distance.

3 *Ab initio* deep learning models

Hydrogen molecular ion: We design a deep NN to obtain the ground state wavefunctions and energies for the single electron in H_2^+ as a function of the interatomic distance. For this we employ a PINN to solve for the eigenvalues and eigenfunctions of the Hamiltonian operator:

$$\hat{\mathcal{H}} = -\frac{1}{2}\nabla^2 - \frac{1}{|\mathbf{r}-\mathbf{R}_1|} - \frac{1}{|\mathbf{r}-\mathbf{R}_2|}, \tag{1}$$

where we use atomic units, $\mathbf{r} = (x, y, z)$, and, without loss of generality, the molecule is oriented along the x-axis, so $\mathbf{R}_1 = -\mathbf{R}_2 = (R, 0, 0)$. We seek NN solutions that are continuous functions of R, and thus we call them generalized parametric neural solutions.

Network architecture: The NN architecture we employ is shown in Fig. 1. The network inputs are $\mathbf{r}$, the coordinates of the electron, and the parameter R, which determines the geometry of the quantum system. As Fig. 1 demonstrates, there are several units in the architecture and the forward pass consists of several branches. The two inputs go through the atomic unit (AU) that returns the hydrogen atomic s-orbitals $\phi_{1,2} = s(|\mathbf{r} \pm \mathbf{R}|) = e^{-|\mathbf{r}\pm\mathbf{R}|}$ for the left ($x = -R$) and right ($x = R$) ions, respectively. This operation is used for feature-engineering and does not contain trainable parameters. Next, the approximate LCAO solution, $\psi_{\text{LCAO}} = \phi_1 \pm \phi_2$, is constructed by passing ϕ_1, ϕ_2 through the LCAO unit; we do not provide the normalized ψ_{LCAO} because the neural ψ will be normalized after the network training. Here we focus on the ground state and use the symmetric ψ_{LCAO} (the "+" sign); the extension to antisymmetric solutions (the "−" sign) is straightforward [8]. This approximate solution, which is accurate as $R \to \infty$, will be used to build the neural ψ. Though any initial guess is possible, starting from a physics-based asymptotic solution has obvious advantages. The features $\phi_{1,2}$ also pass through the Basis Unit (BU), which is a multi-layer fully connected feed forward NN (FFNN) that returns a nonlinear combination of ϕ_1 and ϕ_2, called $N(\mathbf{r}, R)$. This unit respects inversion symmetry by construction [2]: $N(x, y, z, R) = \sum_j w_j \left[B_j(x, y, z, R) + B_j(-x, y, z, R)\right] + b$, where B_j are the outputs of the neurons in the last hidden layer of the BU, while w_j and b indicates a linear output

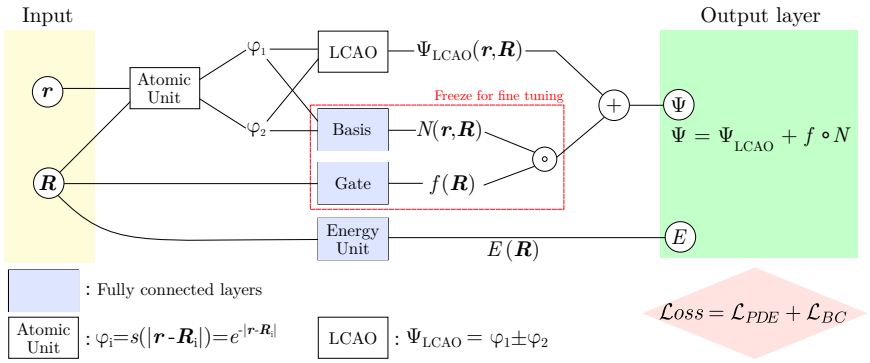

Figure 1: *Ab initio* neural network architecture. In the H_2^+ configuration, $\boldsymbol{R}$ is a vector of two triples, $\boldsymbol{R}_i$ is a single vector describing the position of one nucleus.

layer in the BU. The purpose of this unit is to correct ψ_{LCAO} while respecting the inversion symmetry. Because ψ_{LCAO} is increasingly more accurate as R increases, we introduce a gate $f(R)$ that is a small FFNN, only depending on R, which learns the range in R where $N(x, y, z)$ is important.

The full neural wavefunction is thus:

$$\psi(\mathbf{r}, R) = \psi_{\mathrm{LCAO}}(\mathbf{r}, R) + f(R) \circ N(\mathbf{r}, R), \tag{2}$$

where $\circ$ indicates the element-wise product.

The neural energy $E(R)$ is the output of a separate FFNN that takes only R as input, and hence is independent of $\mathbf{r}$ as required from physical considerations (the total energy of the molecule does not depend on the position of the electrons). This unit allows the network to learn a smooth and differentiable $E(R)$ that can be evaluated at any R. To improve the accuracy of $E(R)$, we perform a fine-tuning optimization where we freeze the BU and gate (red box in Fig. 1) and train the Energy unit (EU) alone. Optionally, after the training phase we can explicitly calculate $\langle \hat{\mathcal{H}} \rangle$ using the neural ψ to get a consistent value of the eigen-energy for any desired value of R.

Optimization: The network is optimized by minimizing a physics-informed loss function:

$$\mathcal{L} = \left\langle \left(\hat{\mathcal{H}}\psi(\mathbf{r}_i, R_i) - E(R_i)\psi(\mathbf{r}_i, R_i) \right)^2 \right\rangle_i + \left\langle \psi(\mathbf{r}_i, R_i)^2 \right\rangle_{|\mathbf{r}_i| > r_{\mathrm{cut}}}, \tag{3}$$

where the brackets denote averaging over the training points. This loss directs the NN to find $\psi(\mathbf{r}, R)$ and $E(R)$ that satisfy the Schrödinger equation with the Hamiltonian in Eq. 1 subject to the boundary conditions $\psi(\mathbf{r}, R) \to 0$ as $|\mathbf{r}| \to \infty$. In practice, the second term in Eq. 3, $\mathcal{L}_{\mathrm{BC}}$, requires that $\psi(\mathbf{r}, R)$ vanish for any $|\mathbf{r}|$ larger than a manually selected cutoff value, r_{cut}.

The architecture used to solve the H_2^+ system consists of 2 hidden layers of 16 neurons each for the BU, one layer of 10 neurons for the gate, and two layers of 32 neurons each for the EU, with a sigmoid activation for all the hidden neurons. We train the NN using the Adam [10] optimizer with a learning rate of 8×10^{-3}. The network is optimized for 5×10^3 epochs but we save the model with the lowest $\mathcal{L}$. For the fine-tuning phase we load the best model and train only the EU using the Adam optimizer with a learning rate of 10^{-4}.

4 Results

We employ the PINN desscribed above to solve the Schrödinger equation for H_2^+. The training set is constructed by sampling 10^6 points in the ranges $(x, y, z) \in [-18, 18]$ with a cutoff at $r_{\mathrm{cut}} = 17.5$ and $R \in [0.2, 3]$. The points are randomly sampled at every epoch yielding more robust training [16, 22]. In Fig. 2(a) we show the loss function during the training and fine-tuning phases, where the vertical dashed red line separates the two phases. Since $\mathcal{L}_{\mathrm{BC}}$ does not depend on E it does not change during the fine tuning. The code is written in pytorch [17] and can be found on github[2]. The training takes less than 5 minutes on an NVIDIA Tesla V100 GPU with 256 GB memory.

[2] `https://github.com/mariosmat/PINN_for_quantum_wavefunction_surfaces`

Neural energy potential surfaces and wavefunction: Figures 2 and 3 show the results after training the NN. Figure 2(b) shows the neural ψ (solid blue) and ψ_{LCAO} (dashed red) along the x-axis for two different values of R indicated by the dashed black lines. The two plots on the right show perspective 3D views of $\psi(x, y, 0)$ for the same two values of R.

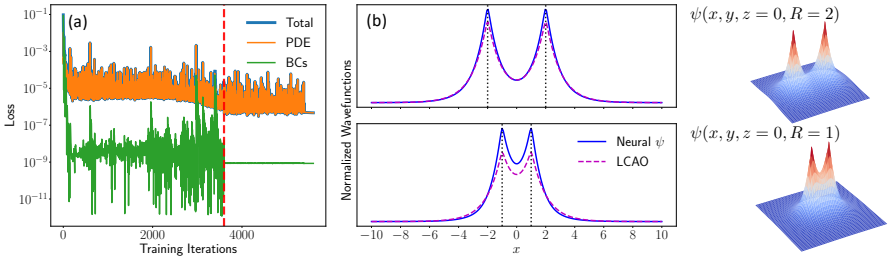

Figure 2: (a) The total loss function $\mathcal{L}$, from Eq.(3), and its components during the training and fine tuning phases, where the dashed red line separates the two phases. (b) Left: Neural ψ (solid blue) and ψ_{LCAO} (dashed red) along the x-axis for two different R values, with the ionic positions $\pm R$ indicated by the vertical dashed black lines. Right: $\psi(x, y, 0)$ for the same two values of R.

In Fig. 3 the upper left panel shows the total energy of the H_2^+ system, namely the sum of the electronic energy and the classical electrostatic energy of the nuclei $(1/2R)$. For the electronic energy we compare $E(R)$ coming directly from the PINN, the expectation value $\langle\hat{\mathcal{H}}\rangle$ calculated with both the neural $\psi(R)$ and ψ_{LCAO}, and a ground truth reference energy calculated in Ref. [5]. The lower left panel shows the differences from the reference energy, where we observe that the error in $E(R)$ is sometimes negative, in violation of the variational principle, while $\langle\hat{\mathcal{H}}\rangle$ is never smaller than the ground truth. The lower right panel shows the gate function, which gives increased weight to the BU for smaller R as expected. The upper right panel of Fig. 3 shows the force between the ions, obtained by differentiating the energy with respect to R. Since $E(R)$ is a continuous and differentiable function of R, we use autograd to calculate the force (dashed blue line). The other energies are computed using finite differences. The force from $E(R)$ using finite differences (solid blue line) deviates from the autograd calculation when the slope of the force is large and thus numerical error is accumulated. This shows the advantage of using the proposed $E(R)$ that eliminates errors from the numerical derivatives.

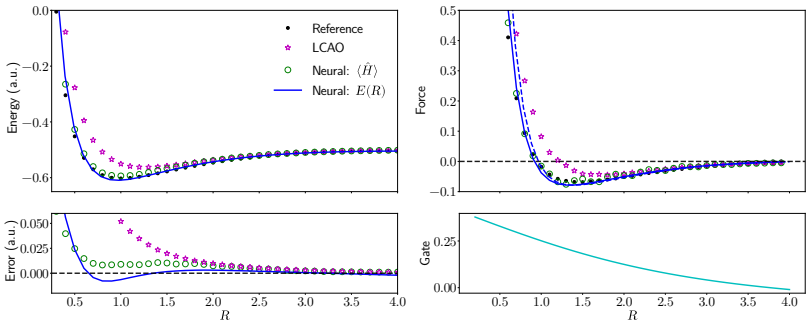

Figure 3: **Left panels**: Energy and relative error in atomic units (a.u.), as functions of R. **Right panels**: force, dE/dR, and gate function $f(R)$; finite differences are used to calculate the forces in all the cases except for the dashed blue line where autograd is used.

5 Conclusions

In this study we present a novel deep learning approach to obtain continuous and differentiable parametric wavefunctions and potential energy surfaces for molecular systems, using a PINN architecture for solving the first-principles quantum mechanical equations. We validate this approach by studying the electronic ground state of H_2^+. Although we presented results only for a simple molecule, the

proposed architecture is generalizable to other similar systems. The most important advantage of this approach is the ability to obtain neural-net wavefunctions and energies that are continuous and differentiable in the parameter space of the nuclear coordinates. Having such a form of the solutions is useful for further computations like the calculation of forces and vibrational frequencies.

Broader Impact

Solving eigenvalue partial differential equations is an important goal in many scientific fields including engineering, applied physics, and quantum chemistry. Solving these equations can be extremely demanding and frequently prohibitive due to the limitations of existing numerical methods. New technologies and more efficient methods for solving differential equations are crucial to accelerate progress in scientific research. In this work, we introduced a deep learning framework for solving eigenvalue partial differential equations for parametric potential functions. The proposed neural network is able to learn parametric eigenvalue and eigenfunction surfaces, namely neural solutions in terms of the independent variables and of the modeling parameters. We demonstrated the method's efficacy by solving the stationary Schrödinger equation for the hydrogen molecular ion with one electron. Generalization to more complicated problems is the subject of ongoing research.

Societal and Environmental Impact: We are not aware of any negative social impact from solving differential equations with neural networks, although, as for any scientific discovery, the results of using this mathematical tool depend on the intentions of the user. As far as the environment is concerned, the proposed method can be generalized to provide a wide range of solutions with a single training. Also, an approximated solution that is embedded in the neural network structure drastically reduces the training time and thus, the consumption of energy to arrive at the solution. For the specific problem presented in this work, the training of the NN took less than five minutes on an NVIDIA Tesla GPU.

Acknowledgements

Research was sponsored by the Army Research Office and was accomplished under Cooperative Agreement Number W911NF-21-2-0147 and Grant Number W911NF-21-1-0184; the STC Center for Integrated Quantum Materials, NSF Grant No. DMR-1231319; and NSF DMREF Award No. 1922172.

References

[1] Angeli, M., Neofotistos, G., Mattheakis, M., & Kaxiras, E. (2022). Modeling the effect of the vaccination campaign on the covid-19 pandemic. *Chaos, Solitons and Fractals*, 154, 111621.

[2] Bhattacharya, A., Mattheakis, M., & Protopapas, P. (2022). Encoding involutory invariance in neural networks. In *In IJCNN at IEEE World Congress on Computational Intelligence*: IJCNN.

[3] Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., Vogt-Maranto, L., & Zdeborová, L. (2019). Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4).

[4] Chen, Y., Lu, L., Karniadakis, G. E., & Negro, L. D. (2020). Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Opt. Express*, 28(8), 11618–11633.

[5] H., W. (1965). Electron energy for h2+ in the ground state. *The Journal of Chemical Physics*, 115 34, 2371–273.

[6] Hermann, J., Spencer, J., Choo, K., Mezzacapo, A., Foulkes, W. M. C., Pfau, D., Carleo, G., & Noé, F. (2022). Ab-initio quantum chemistry with neural-network wavefunctions.

[7] Jin, H., Mattheakis, M., & Protopapas, P. (2020). Unsupervised neural networks for quantum eigenvalue problems. In *2020 NeurIPS Workshop on Machine Learning and the Physical Sciences*: NeurIPS.

[8] Jin, H., Mattheakis, M., & Protopapas, P. (2022). Physics-informed neural networks for quantum eigenvalue problems. In *In IJCNN at IEEE World Congress on Computational Intelligence*: IJCNN.

[9] Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. *Nature Review Physics*.

[10] Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

[11] Krishnapriyan, A. S., Gholami, A., Zhe, S., Kirby, R. M., & Mahoney, M. W. (2021). Characterizing possible failure modes in physics-informed neural networks. In *2022 NeurIPS*: NeurIPS.

[12] Lagaris, I. E., Likas, A., & Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9, 987–1000.

[13] Li, H., Zhai, Q., & Chen, J. Z. Y. (2021). Neural-network-based multistate solver for a static schrödinger equation. *Phys. Rev. A*, 103, 032405.

[14] Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E. (2021). Learning nonlinear operators via deeponet based on the universal approximation theorem of operators involving nonlinear partial differential equations. *Nature machine intelligence*, 3, 218–229.

[15] Manzhos, S. (2020). Machine learning for the solution of the schrödinger equation. *Machine Learning: Science and Technology*, 1(1), 013002.

[16] Mattheakis, M., Sondak, D., Dogra, A. S., & Protopapas, P. (2022). Hamiltonian neural networks for solving equations of motion. *Phys. Rev. E*, 105, 065305.

[17] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., Devito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in pytorch. In *NeurIPS*.

[18] Paticchio, A., Scarlatti, T., Mattheakis, M., Protopapas, P., & Brambilla, M. (2020). Semi-supervised neural networks solve an inverse problem for modeling covid-19 spread. In *2020 NeurIPS Workshop on Machine Learning and the Physical Sciences*: NeurIPS.

[19] Pfau, D., Spencer, J. S., Matthews, A. G. D. G., & Foulkes, W. M. C. (2020). Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Phys. Rev. Research*, 2, 033429.

[20] Raissi, M., Perdikaris, P., & Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378, 686–707.

[21] Rudy, S. H., Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2017). Data-driven discovery of partial differential equations. *Science Advances*, 3(4).

[22] Sirignano, J. A. & Spiliopoulos, K. (2018). Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375, 1339–1364.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]