Solving the Many-Electron Schrödinger Equation

With a Transformer Architecture

Jorge Munoz Laredo

January 10, 2026

Outline

The Schrödinger equation

On 1926 Schrodinger derived the Time Dependent Form.(TDSE)

$$i\,\hbar\,\partial_t \Psi = \hat{H}\,\Psi, \tag{1}$$

- $\Psi \in \mathcal{H}$ is a complex value function called **wave function**.
- $\hat{H}$ is called the **Hamiltonian Operator** , encodes all the information of the energy of the system.
- Depends on the position $\vec{\mathbf{r}}$ of a particle and the temporal evolution (t).
- Ψ encodes all information about the system; $|\Psi|^2$ gives a probability density that integrates to 1.

Hamiltonian

In the position basis:

$$\hat{H} = \frac{\hat{\vec{P}}^2}{2m} + \hat{V} = -\frac{\hbar^2}{2m}\nabla^2 + \hat{V} \tag{2}$$

Time Dependent Form

When the wave function could be written as:

$$\psi(\vec{\mathbf{r}}, t) = R(\vec{\mathbf{r}})T(t) \tag{3}$$

TDSE returns you that:

$$T(t) = e^{-iEt/\hbar} \wedge \hat{H}R(\vec{\mathbf{r}}) = ER(\vec{\mathbf{r}}) \tag{4}$$

Where E is the total energy of the system. The eigen-problem becomes obtain R solving:

Find a Ψ, such that:

$$\left[-\frac{\hbar^2}{2m}\nabla^2 + V(\mathbf{r}, t)\right]\psi(\vec{\mathbf{r}}) = E\psi(\vec{\mathbf{r}}) \tag{5}$$

Find the potential V of the system.

Wave Function as Probability Density

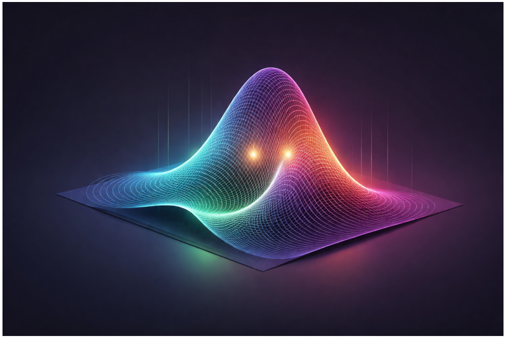


Figure 1: $|\Psi(\mathbf{R})|^2$ represent the probability to find a particle near the position $\mathbf{R}$.

Many-Body System

When considering a many-body system, we need to consider the position of each electron like also the spin of it. When considering n bodies, we have:

$$\hat{H}\psi(\mathbf{x}_1, \ldots, \mathbf{x}_n) = E\psi(\mathbf{x}_1, \ldots, \mathbf{x}_n) \quad (6)$$

With $\mathbf{x}_i = \{\mathbf{r}_i, \sigma\}$, where $\mathbf{r}_i \in \mathbb{R}^3$ is the position of each particle and $\sigma \in \{\uparrow . \downarrow\}$ is the so called spin.

Considerations

- Each particle interact with all the another particles in specific ways.
- For atoms, consider all the protons, electrons and neutrons.
- Solution obey physical laws.

Setting up the Hamiltonian

The first step is obtain a practical form of the **Hamiltonian**.

- Kinetic energy: $T = -\frac{1}{2}\sum_{i=1}^{N} \nabla_i^2$.
- Electron-nuclear attraction: $V_{en} = -\sum_{i,I} \frac{Z_I}{r_{iI}}$.
- Electron-electron repulsion: $V_{ee} = \sum_{i<j} \frac{1}{r_{ij}}$.

$$\hat{H} = -\sum_{i=1}^{N} \frac{1}{2}\nabla_i^2 - \sum_{I=1}^{M} \frac{1}{2M_I}\nabla_I^2 - \sum_{i=1}^{N}\sum_{I=1}^{M} \frac{Z_I}{|\mathbf{r}_i - \mathbf{R}_I|} + \sum_{1\leq i<j\leq N} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} + \sum_{1\leq I<J\leq M} \frac{Z_I Z_J}{|\mathbf{R}_I - \mathbf{R}_J|} \tag{7}$$

Borh Oppenheimer approximation helps with.

Fermi-Dirac statistics

All the fermions follow the Fermi-Dirac Statistics, this is.

- Electrons are indistinguishable fermions.
- Exchanging two electrons flips the wavefunction's sign: $\Psi(\dots i, j \dots) = -\Psi(\dots j, i \dots)$.
- Pauli exclusion: no two electrons can occupy the same

Slater Determinant

We can enforce this using a determinant to enforce an antisymmetric Ψ.

$$\psi = \begin{vmatrix} \phi_1^k(\mathbf{x}_1) & \dots & \phi_1^k(\mathbf{x}_n) \\ \vdots & & \vdots \\ \phi_n^k(\mathbf{x}_1) & \dots & \phi_n^k(\mathbf{x}_n) \end{vmatrix} \tag{8}$$

Where ϕ are called spin orbitals

Kato cusp conditions, Jastrow Factor

When two electrons

- Coulomb potentials cause a sharp cusp in Ψ when particles overlaps.
- Electron–nucleus cusp: $\left.\dfrac{\partial \Psi}{\partial r_{iI}}\right|_{r_{iI}=0} = -\, Z_I \, \Psi(0).$
- Electron–electron cusp: $\left.\dfrac{\partial \Psi}{\partial r_{ij}}\right|_{r_{ij}=0} = \dfrac{1}{2}\, \Psi(0).$

Jastrow Factor $\exp(\mathcal{J})$

In this work we are going to use this specific form:

$$\mathcal{J}_\theta(x) = \sum_{i<j;\sigma_i=\sigma_j} -\frac{1}{4}\frac{\alpha_{par}^2}{\alpha_{par} + |\mathbf{r}_i - \mathbf{r}_j|} + \sum_{i,j;\sigma_i\neq\sigma_j} -\frac{1}{2}\frac{\alpha_{anti}^2}{\alpha_{anti} + |\mathbf{r}_i - \mathbf{r}_j|} \tag{9}$$

Loss: Variational Principle

Variational principle states:

$$E[\Psi] = \frac{\langle \Psi \mid H \mid \Psi \rangle}{\langle \Psi \mid \Psi \rangle} \geq E_0$$

Minimizing $E[\Psi]$ drives the ansatz toward the ground state.

$$E[\Psi] = \mathcal{L}(\Psi_\theta) = \frac{\langle \Psi_\theta | \hat{H} | \Psi_\theta \rangle}{\langle \Psi_\theta | \Psi_\theta \rangle} = \frac{\int d\mathbf{R} \Psi^*(\mathbf{R}) \hat{H} \Psi(\mathbf{R})}{\int d\mathbf{R} \Psi^*(\mathbf{R}) \Psi(\mathbf{R})}$$

Define:

$$p_\theta(\mathbf{R}) = |\Psi_\theta(\mathbf{R})|^2 \frac{1}{\int d\mathbf{R}' \Psi_\theta^2(\mathbf{R}')} \wedge E_L(\mathbf{R}) = \frac{\hat{H}\Psi_\theta(\mathbf{R})}{\Psi_\theta(\mathbf{R})}$$

Then:

$$\mathcal{L}_\theta = \mathbb{E}_{\mathbf{R} \sim |\Psi_\theta|^2} [E_L(\mathbf{R})] \tag{10}$$

Variational Monte Carlo

Quantum Monte Carlo

With the samples $\mathbf{R}_1, \dots, \mathbf{R}_M \sim |\Psi|^2_\theta(\mathbf{R})$ we can make:

$$\mathcal{L}_\theta = \mathbb{E}_{\mathbf{R}\sim\Psi^2_\theta}[E_L(\mathbf{R})] \approx \frac{1}{M}\sum_{i=1}^{M} E_L(\mathbf{R}_k) \tag{11}$$

With:

$$E_L(\mathbf{R}_k) = \frac{\hat{H}\psi(\mathbf{R}_k)}{\psi(\mathbf{R}_k)} = -\frac{1}{2}\frac{\nabla^2\psi(\mathbf{R_k})}{\psi(\mathbf{R}_k)} + V(\mathbf{R}_k)$$

Obtain $\mathbf{R}_k \rightarrow$ Metropolis-Hastings Algorithm

Metropolis-Hastings Algorithm

Goal: Generate many samples $\mathbf{R} \sim \rho$, Requirement: $C\rho$

1. $\mathbf{R}_0 \in E$ random.
2. Propose $\mathbf{R}' = \mathbf{R}_0 + \eta$,where $\eta \sim q(\eta)$, (Normal Gaussian)
3. Compute the quantity:

$$A(\mathbf{R_0}, \mathbf{R}') = \min\left(1, \frac{\rho(\mathbf{R}')}{\rho(\mathbf{R}_0)}\right)$$

4. Generate a uniform number $U \in [0,1]$. If: $U < A(\mathbf{R}_0, \mathbf{R}')$ then $\mathbf{R_1} = \mathbf{R}'$, otherwise try another $\mathbf{R}'$. Accept or decline.

Metropolis-Hastings

In each sample generates $E_L(\mathbf{R}_k)$ then average to obtain $\mathbb{E}(E_L)$ and begin the back propagation step.

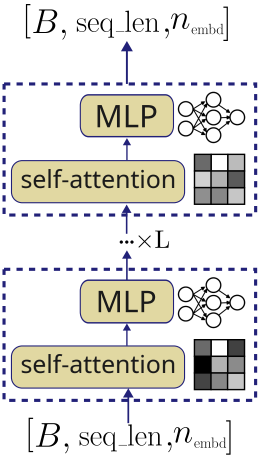
Gradients of the Loss

Using calculus you obtain:

$$\nabla_\theta \mathcal{L} = 2\mathbb{E}_{x\sim\Psi^2}[(E_L(x) - \mathbf{E}_p(E_L)) \log \psi] \tag{12}$$

This expectation is calculated in the same way.

Transformer Architecture



$[B, \text{seq\_len}, n_{\text{embd}}]$
MLP
self-attention
$\cdots \times L$
MLP
self-attention
$[B, \text{seq\_len}, n_{\text{embd}}]$

Figure 2: Tranformer backbone

Multi Head Attention → Self Attention

- n_{embd} the embedding dimension
- n_h the number of attention heads
- d_h the dimension per head
- $\mathbf{h}_t \in \mathbb{R}^{n_{\text{embd}}}$ the hidden dimension.

Attention on the room

The learnable matrices are:

$$W^Q, W^K, W^V \in \mathbb{R}^{n_{\text{embd}} \times n_{\text{embd}}}$$

$$\mathbf{k}_i = \mathbf{W}^k \mathbf{h}_i, \mathbf{q}_i = \mathbf{W}^q \mathbf{h}_i, \mathbf{v}_i = \mathbf{W}^v \mathbf{h}_i$$

$$[\mathbf{q_1}, \mathbf{q_2}, \ldots, \mathbf{q_{n_h}}] = \mathbf{q}$$

$$[\mathbf{k_1}, \mathbf{k_2}, \ldots, \mathbf{k_{n_h}}] = \mathbf{k}$$

$$[\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{n_h}}] = \mathbf{v}$$

In the $i-th$ head:

$$\mathbf{o}_{t,i} = \sum_{j=1}^{t} \text{Softmax}\left(\frac{\mathbf{q}_{t,i}^T \mathbf{k}_{j,i}}{\sqrt{d_h}}\right) \mathbf{v}_{j,i} \tag{13}$$

W^O the output projection matrix.

$$\mathbf{u}_t = W^O [\mathbf{o}_{t,1}; \mathbf{o}_{t,2}; \ldots; \mathbf{o}_{t,n_h}]$$

Outline

Psiformer Ansatz

Ansatz: Proposal model that you propose guided by intuition and that you optimize.

- be *antisymmetric* under particle exchange,
- capture strong *electron–electron correlations.*

Proposed ansatz

Motivated by these constraints, we propose a Slater–Jastrow form:

$$\Psi_\theta(\mathbf{R}) = \underbrace{\exp\left(\mathcal{J}_\theta(\mathbf{R})\right)}_{\text{Coulomb correlations}} \times \underbrace{\sum \omega_k \det\left[\boldsymbol{\phi}_\theta^k(\mathbf{R})\right]}_{\text{antisymmetry}}$$

Psi Former Architecture

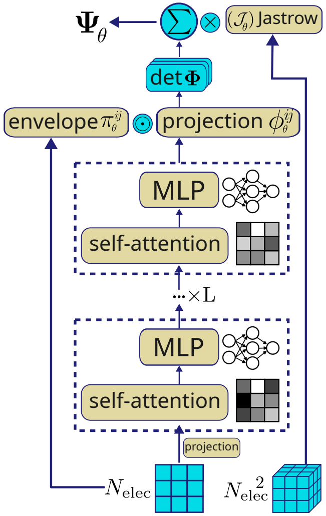
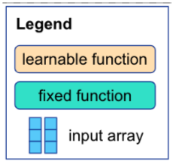


Figure 3: Psi Former Architecture



- The envelope is a matrix with learned exponential decay as elements.
- Learn the Jastrow factor only have two parameters.
- In practice we use logarithm scale.

Psiformer Shapes

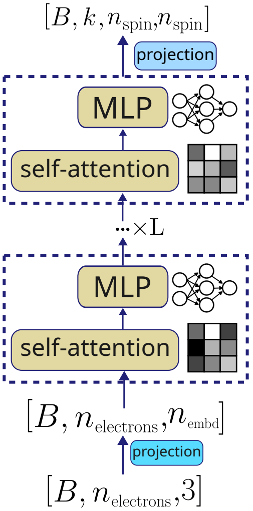

$[B, k, n_{\text{spin}}, n_{\text{spin}}]$
projection
MLP
self-attention
$\cdots \times \text{L}$
MLP
self-attention
$[B, n_{\text{electrons}}, n_{\text{embd}}]$
projection
$[B, n_{\text{electrons}}, 3]$

Figure 4: Psiformer shapes handling

Methodology

Hyperparameter	Small	Large
Layers L	2	4
Heads H	4	8
Model Dim d	256	512
MLP Dim d_{ff}	1024	2048
Determinants K	1	2
MCMC walkers N_w	1024	2048
MCMC steps / iter	10	10
Learning rate	2×10^{-4}	1×10^{-4}
Total parameters	50441	$3M$

Figure 5: Psiformer Torch Small and Large

Goal: Obtain accurate ground state-energies using an Ansatz created with **Torch** library

Training Loop:

Set the spins and electron number

1.Sample configurations $\mathbf{R}_k$

2.Estimate local energy E_L

3.Backpropagation using **A.D**

4.Update parameters θ using **AdamGrad**.

5. Track metrics with **Wandb.**

Implementation : Laplacian Computation

The kinetic energy requires the Laplacian: $\nabla^2\Psi(\mathbf{R}) = \sum_i \frac{\partial^2\Psi}{\partial R_i^2}$

```
R = R_o.requires_grad_(True)      # particle coordinates
psi = model(R)                   # neural wavefunction
# first derivative: gradient
grad_psi = torch.autograd.grad(
    psi, R,
    create_graph=True,
    # retain_graph=True
)[0]
# second derivative: Laplacian
laplacian = 0.0
for i in range(R.shape[-1]):
    laplacian += torch.autograd.grad(
        grad_psi[..., i], R,
        # create_graph=True,
        retain_graph=True
    )[0][..., i]
```

Determinant Stability in Psiformer

$$\Psi_\theta(\mathbf{R}) \propto \det[\mathbf{\Phi}_\theta(\mathbf{R})]$$

Derivative of a determinant.

$$\frac{\partial \det(\mathbf{A})}{\partial \mathbf{A}} = \det(\mathbf{A})\mathbf{A}^{-T}$$

Key instability. If $\mathbf{\Phi}$ becomes singular or nearly singular:

$$\det \mathbf{A} \to 0 \quad \Rightarrow \quad [a_{ii}^{-1}] \to \infty$$

so the gradient can explode even when the wavefunction itself is small.

Fix: Custom Operation

Idea: Use SVD ($A = U\Sigma V^T$) to compute $\nabla \log \det A = A^{-T}$ without explicit inversion (Appendix D).

```
import torch

class StableLogDet(torch.autograd.Function):
    @staticmethod
    def forward(ctx, A):
        # Decompose A: S are singular values
        U, S, Vh = torch.linalg.svd(A)
        ctx.save_for_backward(U, S, Vh)
        return S.log().sum()

    @staticmethod
    def backward(ctx, g):
        U, S, Vh = ctx.saved_tensors
        # Reconstruct A^{-T} = U * diag(1/S) * Vh
        inv_S = torch.diag_embed(1.0 / S)
        grad_A = U @ inv_S @ Vh
        return g * grad_A
```

Optimizer: Adam vs. AdamW

Core Difference: AdamW *decouples* weight decay from the gradient update to fix regularization on adaptive optimizers.

1. Adam (Entangled L2 Regularization)

- Decay is added to the gradient, so it gets scaled by the adaptive variance.

$$
\begin{aligned}
g_t &= \nabla \mathcal{L} + \lambda \theta_t \\
\theta_{t+1} &= \theta_t - \text{AdamStep}(g_t)
\end{aligned}
$$

2. AdamW (Decoupled Weight Decay)

- Decay is applied directly, bypassing the adaptive scaling mechanism.

$$
\begin{aligned}
g_t &= \nabla \mathcal{L} \\
\theta_{t+1} &= \theta_t - \text{AdamStep}(g_t) - \eta \lambda \theta_t
\end{aligned}
$$

Keeping the GPU Busy: Batched Energy Evaluation

Naive training issue. Initial training evaluated energies step-by-step over MCMC samples.

$$\text{GPU utilization} \approx 30\%$$

Solution: batched evaluation. MCMC samples are reshaped and flattened:

$$(\text{mc\_steps}, B, n_e, 3) \;\rightarrow\; (\text{mc\_steps} \times B, n_e, 3)$$

Result.

$$\text{GPU utilization} \approx 99\%$$

Results: Convergence Curve

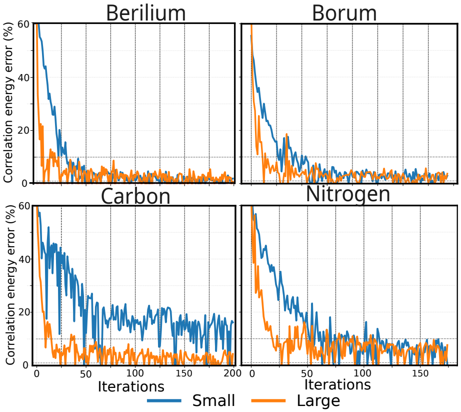


Figure 6: Convergence Curve

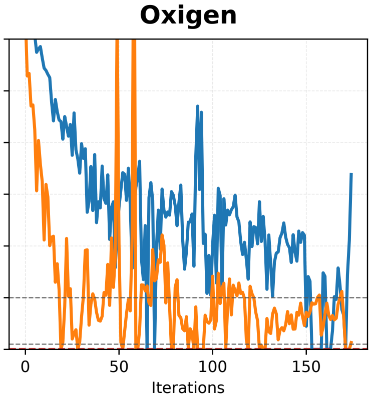


Figure 7: Oxigen Convergence

Results: Energy Estimates

Atom	E_b	E_s	E_l	Δ_s	Δ_l	Δ_{l-s}
H	-0.500	-0.492	**-0.498**	0.008	0.002	**-0.006**
He	-2.903	-2.801	**-2.893**	0.102	0.010	**-0.092**
Li	-7.478	-7.097	**-7.243**	0.381	0.235	**-0.146**
Be	-14.667	-13.901	**-14.237**	0.766	0.430	**-0.336**
B	-24.653	-24.042	**-24.567**	0.611	0.086	**-0.525**
C	-37.845	-35.492	**-36.457**	2.353	1.388	**-0.965**
N	-54.589	-50.492	**-51.700**	4.097	2.889	**-1.208**
O	-75.067	-63.492	**-72.139**	11.575	2.928	**-8.647**

Figure 8: Ground state energies (Ha) for H-O, baseline vs Psiformer

Comments

- Pretraining using external data.
- Laplacian Bottleneck
- KFCA Optimizer (Natural Gradient Descent)
- Flash Attention
- Learning transferability
- Scaling Laws

References

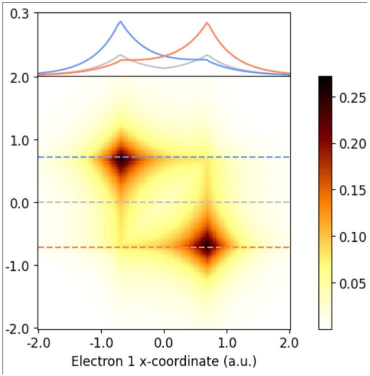
F. Hermann, Z. Schätzle, and F. Noé,
A Self Attention Ansatz for Ab Initio Quantum Chemistry,
arXiv:2309.12345, 2023.

J. Pfau, J. S. Spencer, A. G. D. G. Matthews, and W. M. C. Foulkes,
Ab Initio Solution of the Many-Electron Schrödinger Equation with Deep Neural Networks,
Physical Review Research, 2, 033429, 2020.

A. Vaswani et al.,
Attention Is All You Need,
Advances in Neural Information Processing Systems (NeurIPS), 2017.

Thanks!

I thank the Computer Science Faculty for providing access to GPU resources, in particular NVIDIA GeForce RTX 4080 Super which enabled the training and evaluation of Psiformer models reported in this work.


Electron 1 x-coordinate (a.u.)