

# Solving the Many-Electron Schrödinger Equation With a Transformer Architecture

Jorge Munoz Laredo

November 23, 2025

- 1 The Schrödinger Wave Function and the physical laws that rule
  - Schrödinger Equation
  - Physical laws and conditions
  - Optimizing an Ansatz
- 2 Deep Learning Tool Box
  - Natural Gradient Descent
  - Attention Mechanism
- 3 Fermi Net and Psiformer
  - Fermi Net
  - Psi Former

# The Schrödinger equation

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

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

- $\Psi \in \mathcal{H}$  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  $\vec{r}$  of a particle and the temporal evolution ( $t$ ).
- $\Psi$  encodes all information about the system;  $|\Psi|^2$  gives a probability density that integrates to 1.

## Hamiltonian

In the position basis:

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

# Time Dependent Form

When the wave function could be written as:

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

TDSE returns you that:

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

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

Find a  $\Psi$ , such that:

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

Find the potential  $V$  of the system.

# 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:

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

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:  $V_{ee} = \sum_{i < j} \frac{1}{r_{ij}}$ .

$$\hat{H} = - \sum_{i=1}^N \frac{1}{2} \nabla_i^2 - \sum_{I=1}^M \frac{1}{2M_I} \nabla_I^2 - \sum_{i=1}^N \sum_{I=1}^M \frac{Z_I}{|\mathbf{r}_i - \mathbf{R}_I|} \quad (1)$$

$$+ \sum_{1 \leq i < j \leq N} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} + \sum_{1 \leq I < J \leq M} \frac{Z_I Z_J}{|\mathbf{R}_I - \mathbf{R}_J|} \quad (2)$$

**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  $\Psi$ .

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

Where  $\phi$  are called spin orbitals

# Kato cusp conditions, Jastrow Factor

When two electrons

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

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

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

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



# Loss: Variational Principle

Try an ansatz and optimize it.

Variational principle states that the energy of any ansatz is greater than the true energy.

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

Minimizing  $E[\Psi]$  drives the ansatz toward the ground state.  $E_0$  is the true ground energy (minimum possible).

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

Define:

$$p_\theta(x) = \frac{\Psi_\theta^2(x)}{\int dx' \Psi_\theta^2(x')} \quad \text{and} \quad E_L(x) = \Psi_\theta^{-1}(x) \hat{H} \Psi_\theta(x)$$

Then:

$$\mathcal{L}_\theta = \mathbb{E}_{x \sim p_\theta}[E_L(x)]$$

# Variational Monte Carlo

To evaluate that expectation we use Monte Carlo estimator.

## Quantum Monte Carlo

With the samples  $\mathbf{R}_1, \dots, \mathbf{R}_M \sim p_\theta(\mathbf{R})$  we can make:

$$\mathcal{L}_\theta = \mathbb{E}_{x \sim p_\theta}[E_L(x)] \approx \frac{1}{M} \sum_{i=1}^m E_L(\mathbf{R}_k)$$

With:

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

To obtain those samples we use the **Metropolis-Hastings Algorithm** which handles well high dimensions.

# Metropolis-Hastings Algorithm

1. Take a initial configuration  $\mathbf{X}_0 \in E$  arbitrary:
2. Propose  $\mathbf{X}' = \mathbf{X}_0 + \eta$ , where  $\eta \sim q(\eta)$ ,  $q$  is a probability density on  $E$  called proposal kernel. symmetric Gaussian
3. Compute the quantity:

$$A(\mathbf{X}_0, \mathbf{X}') = \min \left( 1, \frac{\rho(\mathbf{X}')}{\rho(\mathbf{X}_0)} \frac{q(\mathbf{X}' - \mathbf{X}_0)}{q(\mathbf{X}_0 - \mathbf{X}')} \right)$$

Where  $\rho$  is the target distribution where we want sample, In the case where  $q$  is symmetric, this simplifies to:

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

4. Generate a uniform number  $U \in [0, 1]$
5. If:  $U < A(\mathbf{X}_0 \rightarrow \mathbf{X}'_l)$  then  $\mathbf{X}_1 = \mathbf{X}'$ , otherwise try another  $\mathbf{X}'$ .  
Accept or decline.

6. Repeat until obtain  $N_{eq}$  accepted, the change stabilizes (stationary distribution) this phase is called **burn in**.
7. From  $\mathbf{X}_{N_{eq}}$  generate  $M$  samples until reach the sample  $\mathbf{X}_{N_{eq}+M+1}$ . 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:

$$\nabla_{\theta} \mathcal{L} = 2\mathbb{E}_{x \sim \Psi^2} [(E_L(x) - \mathbb{E}_{x' \sim \Psi^2} [E_L(x')]) \nabla \log |\Psi(x)|]$$

\* This expectation is calculated in the same way.

- ① The Schrödinger Wave Function and the physical laws that rule
  - Schrödinger Equation
  - Physical laws and conditions
  - Optimizing an Ansatz
- ② Deep Learning Tool Box
  - Natural Gradient Descent
  - Attention Mechanism
- ③ Fermi Net and Psiformer
  - Fermi Net
  - Psi Former

# Fisher Information Matrix

Gradient descent assumes that parameters lives on the Euclidian Space  $\mathbb{R}^n$ . ( $\mathcal{L}(\theta_1, \dots, \mathbf{x})$ )

But in the case where parameters lives on a probability distribution  $p(x, \theta)$  that assumption don't work to well. A natural **metric** when looking for compare the distributions  $p(x, \theta)$  and  $p(x, \theta + \delta\theta)$  is the **Fisher information Matrix**.

## Fisher Matrix $F(\theta)$

Let  $x \sim p(x|\theta)$ . Define the score:

$$s_{\theta}(x) = \nabla_{\theta} \log p(x|\theta)$$

$$F(\theta) = \mathbb{E}_{x \sim p(\cdot|\theta)} [s_{\theta}(x) s_{\theta}(x)^{\top}]$$

$s \in \mathbb{R}^d$  a column vector.  $d$  number of parameters.

## Natural Gradient

Using the Fisher Metric, the steepest descent direction of the loss is:

$$-F(\theta)^{-1}\nabla_{\theta}\mathcal{L}$$

The updates becomes:

$$\theta_{k+1} = \theta_k - \eta F(\theta_k)^{-1} \delta_{\theta} \mathcal{L}$$

- Compute  $F$ , expensive.
- KFAC ameliorate this with two approximations.

# Optimizer: KFAC (natural gradient)

The first approximation is:  $F_{ij}$  is assumed zero when  $\theta_i$  and  $\theta_j$  are on different layers (neural network). So we just care about the same layer  $\ell$ . Calculus say.

$$\mathbb{E}_{p(\mathbf{X})} \left[ \frac{\partial \log p(X)}{\partial \text{vec}(\mathbf{W}_\ell)} \frac{\partial \log p(X)}{\partial \text{vec}(\mathbf{W}_\ell)}^\top \right] = \mathbb{E}_{p(\mathbf{X})} [(\mathbf{a}_\ell \otimes \mathbf{e}_\ell)(\mathbf{a}_\ell \otimes \mathbf{e}_\ell)^\top]$$

Where  $\mathbf{a}_\ell$  are the forward activation and  $\mathbf{e}_\ell$  are the backward sensitivities for that layer.

$$\mathbb{E}_{p(\mathbf{X})} [(\mathbf{a}_\ell \otimes \mathbf{e}_\ell)(\mathbf{a}_\ell \otimes \mathbf{e}_\ell)^\top]^{-1} \approx \mathbb{E}_{p(\mathbf{X})} [\mathbf{a}_\ell \mathbf{a}_\ell^\top] \otimes \mathbb{E}_{p(\mathbf{X})} [\mathbf{e}_\ell \mathbf{e}_\ell^\top]^{-1}$$



# Attention on the room

**Multi Head Attention** Let  $d$  be the embedding dimension,  $n_h$  be the number of attention heads,  $d_h$  be the dimension per head, and  $\mathbf{h}_t \in \mathbb{R}^d$  be hidden dimension. The learn matrices are:

$$W^Q, W^K, W^V \in \mathbb{R}^{d_h n_h \times d}$$

We obtain the key, queries, and values vectors with:

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

We slice this vectors:

$$[\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_{n_h}] = \mathbf{q}$$

$$[\mathbf{k}_1, \mathbf{k}_2, \dots, \mathbf{k}_{n_h}] = \mathbf{k}$$

$$[\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{n_h}] = \mathbf{v}$$

In the  $i - th$  head:

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

# Attention!

Let  $W^O \in \mathbb{R}^{d \times d_h n_h}$  be the output projection matrix.

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

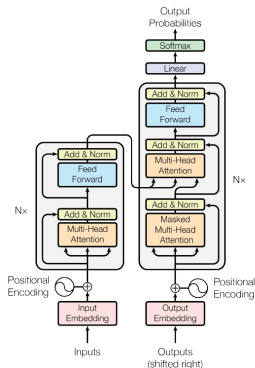


Figure 1: Original Transformer

- ① The Schrödinger Wave Function and the physical laws that rule
  - Schrödinger Equation
  - Physical laws and conditions
  - Optimizing an Ansatz
- ② Deep Learning Tool Box
  - Natural Gradient Descent
  - Attention Mechanism
- ③ Fermi Net and Psiformer
  - Fermi Net
  - Psi Former

## Orbital

To compute the orbitals we need to compute.

$$\phi_i^{k\alpha}(\mathbf{r}_j^\alpha; \{\mathbf{r}_{/j}^\alpha\}; \{\mathbf{r}^{\bar{\alpha}}\}) = \left( \mathbf{w}_i^{k\alpha} \cdot \mathbf{h}_j^{L\alpha} + g_i^{k\alpha} \right) \sum_m \pi_{im}^{k\alpha} \exp \left( -|\boldsymbol{\Sigma}_{im}^{k\alpha}(\mathbf{r}_j^\alpha - \mathbf{R}_m)| \right)$$

Where the hidden states are:

$$\begin{aligned} \mathbf{h}_i^{\ell+1\alpha} &= \tanh \left( \mathbf{V}^\ell \mathbf{f}_i^{\ell\alpha} + \mathbf{b}^\ell \right) + \mathbf{h}_i^{\ell\alpha} \\ \mathbf{h}_{ij}^{\ell+1\alpha\beta} &= \tanh \left( \mathbf{W}^\ell \mathbf{h}_{ij}^{\ell\alpha\beta} + \mathbf{c}^\ell \right) + \mathbf{h}_{ij}^{\ell\alpha\beta} \end{aligned} \quad (3)$$

Finally:

$$\psi(\mathbf{r}_1^\uparrow, \dots, \mathbf{r}_{n\downarrow}^\downarrow) = \sum_k \omega_k \left( \det \left[ \phi_i^{k\uparrow}(\mathbf{r}_j^\uparrow; \{\mathbf{r}_{/j}^\uparrow\}; \{\mathbf{r}^\downarrow\}) \right] \right. \quad (4)$$

$$\left. \det \left[ \phi_i^{k\downarrow}(\mathbf{r}_j^\downarrow; \{\mathbf{r}_{/j}^\downarrow\}); \{\mathbf{r}^\uparrow\}; \right] \right) . \quad (5)$$

# Fermi Net Architecture

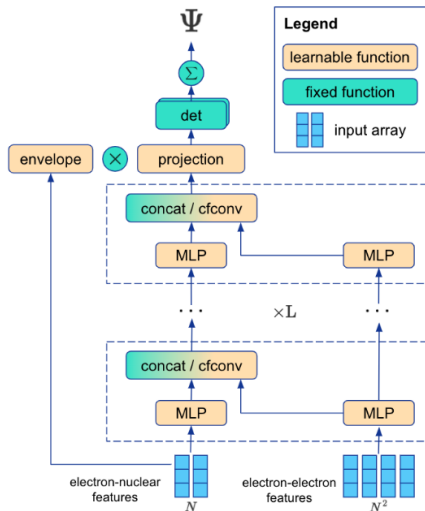


Figure 2: Fermi Net Architecture

# The Psiformer Ansatz (overview)

The output of **Psiformer** is:

$$\Psi_{\theta}(\mathbf{x}) = \exp(\mathcal{J}_{\theta}(\mathbf{x})) \sum_{k=1}^{N_{\text{det}}} \det \left[ \boldsymbol{\Phi}_{\theta}^k(x) \right]$$

We obtain the hidden states:

$$A_h^l = [\text{SelfAttn}(\mathbf{h}_1^l, \dots, \mathbf{h}_N^l; \mathbf{W}_q^{\ell h}, \mathbf{W}_k^{\ell h}, \mathbf{W}_v^{\ell h})]$$

$$A^{\ell} = \text{concat}_h[A_h^{\ell}]$$

And then used it apply:

$$\mathbf{f}_i^{\ell+1} = \mathbf{h}_i^{\ell} + W_o^{\ell} A^{\ell}$$

And generate a new hidden state:

$$\mathbf{h}_i^{\ell+1} = \mathbf{f}_i^{\ell+1} + \tanh\left(\mathbf{W}^{\ell+1} \mathbf{f}_i^{\ell+1} + \mathbf{b}^{\ell+1}\right)$$

# Psi Former Architecture

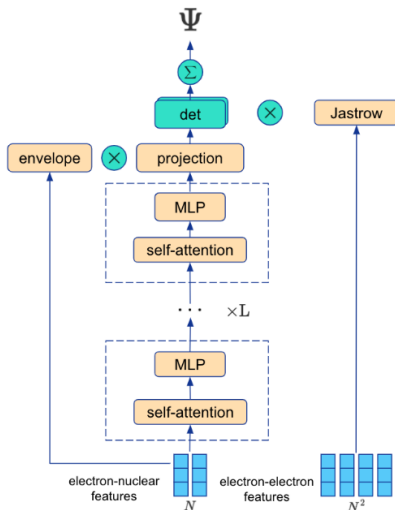


Figure 3: Psi Former Architecture

# Thanks

Thanks.