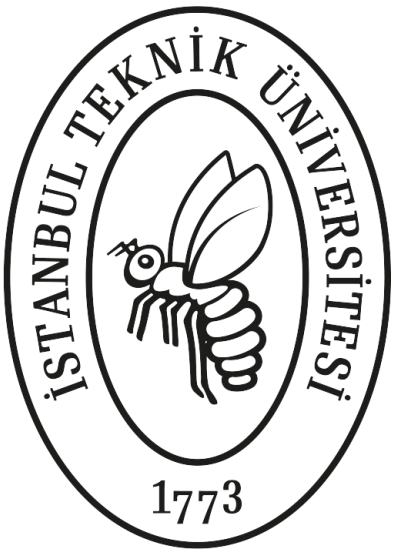
ISTANBUL TECHNICAL UNIVERSITY

FACULTY OF SCIENCE AND LETTERS

Graduation Project



Machine Learning and Non-linear Schrödinger Equation

Hüseyin Talha Şenyaşa

Department : Physics Engineering
Student ID : 090120132
Advisor : Assoc. Prof. A. Levent Subaşı

FALL 2017

Summary

We train an artificial neural network to estimate the ground state energy of a one-dimensional Bose-Einstein condensate in harmonic trapping potential. Such a system can be described by the solution of a non-linear Schrödinger equation also called a Gross-Pitaevskii equation. We also use the method for the inverse problem of predicting the non-linearity parameter using the ground state density profile for a given harmonic trapping potential.

Contents

1 Introduction and Motivation

Machine learning.
General usage area.
ML in physics and Phyiscs in ML.
ML&SE article.
Ours difference.

2 Gross Pitaevskii Equation

2.1 General information about GPE

2.2 Why and how nonlinearty is introduced.

2.3 Physical and mathematical interpretation of interaction parameter. (phy: attractive, repulsive math:dominance of the terms)

2.4 Stationary form.

2.5 Potential, kinetic and interaction energy expressions.

2.6 Reduction of dimension.

2.7 Analytic solution and approximation.

2.8 Numeric Solution and Dataset Generation

2.8.1 Scaling

The scaling of GPE is generally done according to potential type and there are more than one scaling conventions **REFS**. In our study, we use a more general scaling to investigate how different scalings affect the precision. To do that, we compare numerical solutions by representing same physical system with different scaling coefficients in both framework. Then we compare these results first internally and then we do a cross check.
We also compare numerical solutions of two different solvers implemented in different frameworks to show that the numerical solutions are **consistent**.

GPE is given as,

$$\frac{-\hbar^2}{2m}\frac{d^2\psi}{dz^2} + V(z)\psi + g|\psi|^2\psi = \mu\psi \tag{1}$$

First we define dimensionless potential, and then we make the length dimensionless,

$$\overline{V}(z) \equiv \frac{V(z)}{\gamma E_0}, \qquad \widetilde{z} \equiv \frac{z}{\beta L}$$

Here γ and β positive real numbers. E_0 is in energy unit and L is in length and they are defined respectively as;

$$E_0 = \frac{\hbar^2}{2m}$$

$$\widetilde{V}(\widetilde{z}) \equiv \overline{V}(\beta L z)$$

If these transformations are pluged into the Eq. (1) it becomes,

$$\frac{-\hbar^2}{2m\gamma E_0}\frac{1}{\beta^2 L^2}\frac{d^2\psi}{d\widetilde{z}^2} + \widetilde{V}(\widetilde{z})\psi + \frac{g}{\gamma E_0}|\psi|^2\psi = \frac{\mu}{\gamma E_0}\psi \tag{2}$$

To obtain final form, we define dimensionless energy, wave function and interaction parameter respectively.

$$\widetilde{\mu} \equiv \frac{\mu}{\gamma E_0}, \qquad \widetilde{\psi} \equiv \psi\sqrt{\frac{\beta L}{N}}, \qquad \widetilde{g} \equiv \frac{gN}{\gamma E_0 \beta L}$$

To control scaling coefficients we set the coefficient of the kinetic term to an arbitrary positive real number α

$$\alpha = \frac{\hbar^2}{2m\gamma E_0}\frac{1}{\beta^2 L^2},$$

and set $E_0 = \hbar^2/2m$. Now the scaling of GPE can be controlled by α and β only.

$$-\alpha\frac{d^2\widetilde{\psi}}{d\widetilde{z}^2} + \widetilde{V}(\widetilde{z})\widetilde{\psi} + \widetilde{g}|\widetilde{\psi}|^2\widetilde{\psi} = \widetilde{\mu}\widetilde{\psi} \tag{3}$$

We are going to change these scaling coefficients and see their effects by comparing results. An example of setting the scale coefficients is shown in appendix A.

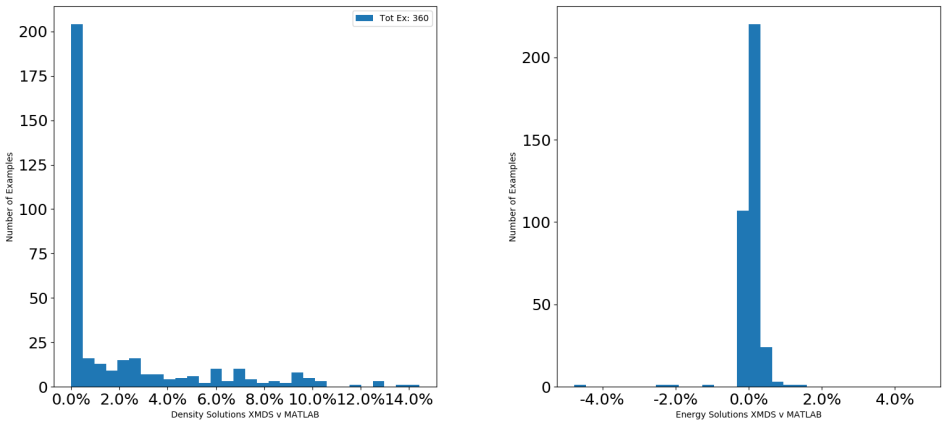


Figure 1: Error in density and energy

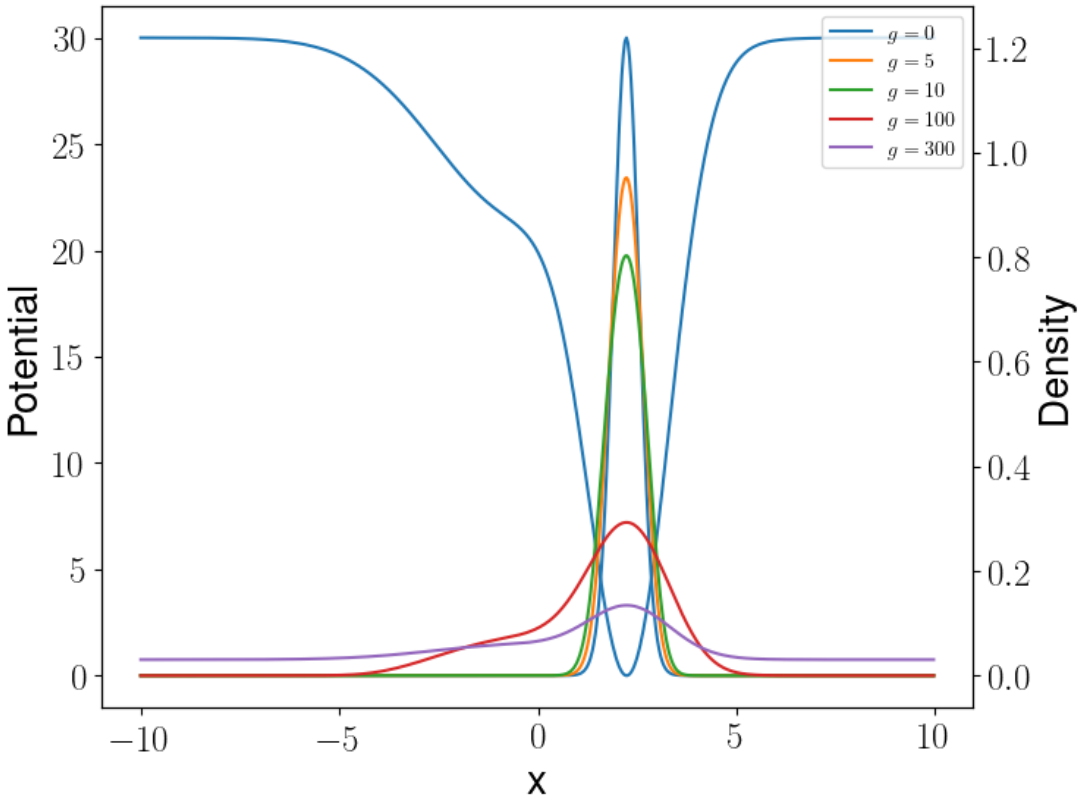


Figure 2: Density under Double Inverted Gaussian Potential with different interaction parameter.

2.8.2 Brief info about imaginary time evolution. (detailed in APPENDIX)

2.8.3 XMDS framerwork and other programs.

2.8.4 Potential generation

In potential generation, we built a modular structure such that the algorithms to generate potentials are independent from the restrictions forced by numerical techniques such as boundary conditions or scaling. The generated potentials are sent to another method supplied by module, and this method rescales, and applies a envelope functions to ensure that potential goes to numerical limit at boundaries.
The envelope functions is defined as,

$$\text{env}(x) = \tanh\left(\beta(x + x_l)\right) + \tanh\left(\beta(x + x_r)\right) \tag{4}$$

where x_l and x_r bounds are given in the Table 1 and the plot is given in Fig. 3. As it is shown, envelope function goes to zero at values greater than x_r or lower than x_l and the rapidness is proportional with β.

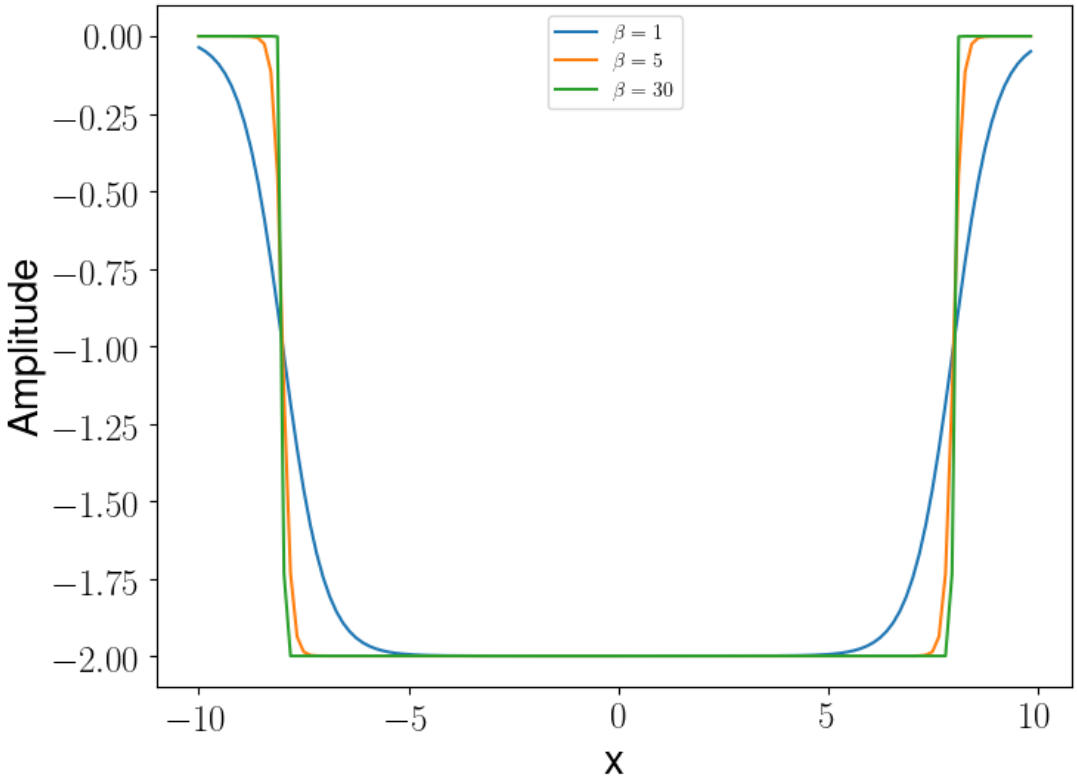
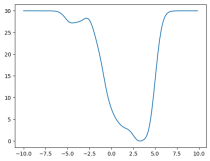
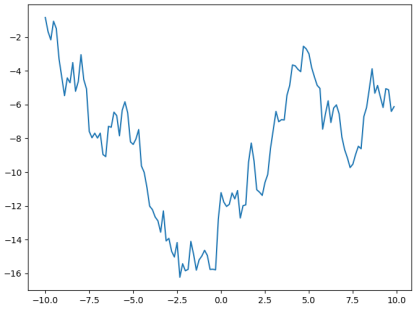


Figure 3: Envelope Function

The two potential in Figure 3 are combined and a gaussian filter applied.





2.8.5 Potential types (with analytic forms etc.)

We use **six** different types of potential in our study. The first three potential which have analytic form given in Table 1. are 1D version of potentials studied in [1]. We implement two more random potential generator to study effect of generation process. The generation processes of the random potentials are described in Section 2.8.6.

2.8.6 Random potential generations with different method. (Reason)

To be able to observe the effect of random potential generation to the results we use three different random potential generation algorithms. The first one is random walk with random step size such that the first value of the potential array is initialized with a random number. After that, another random number is added to this value to obtain the next element of the array and so on. The distribution of the random numbers is gaussian in this process. The resultant array is not guaranteed to be smooth. Gaussian filter is applied with a random sigma value to the potential array to make it smooth.

Algorithm 1 RandomPotential1

procedure RandomPotential1
    $Points = GaussianDistributedRandomPoints()$
    $Len = Length(Points)$
    $Potential[0] = Points[0]$
    for i $= 0$ **to** $Len - 1$ **do**
        $Potential[i+1] = Potential[i] + Points[i]$
    $Potential = GaussianFilter(Potential, \sigma)$

Table 1: Potentials

Potential	Analytic Form / Explanation	Parameters	Min	Max
Infinite Well	$V(x) = \begin{cases} 0 & \text{if } x_l < x < x_r \\ \infty & \text{if } \textit{otherwise} \end{cases}$	x_l x_r $x_r - x_l$	-9 -4.5 1	4.5 9 8
Harmonic	$V(x) = \frac{1}{2}m\omega^2(x - x_0)^2$	ω x_0	0.01 -5	3 5
DI Gaussian	$V(x) = -A_1 \exp(\frac{(x-\mu_1)^2}{\sigma_1^2}) - A_2 \exp(\frac{(x-\mu_2)^2}{\sigma_2^2})$	A_1, A_2 μ_1, μ_2 σ_1, σ_2	1 -5 0.5	10 5 4
Random1	Random walk with random step size	Step Size (Y axis)	-4	4
Random2	Summation of sines and cosines with random coefficients	Number of Terms A_1, A_2 n_1, n_2 σ	1 -4 -6.30 0.1	100 4 6.30 10
Random3	Substraction of two binary grid	Scale Factor	8	8
Random4				

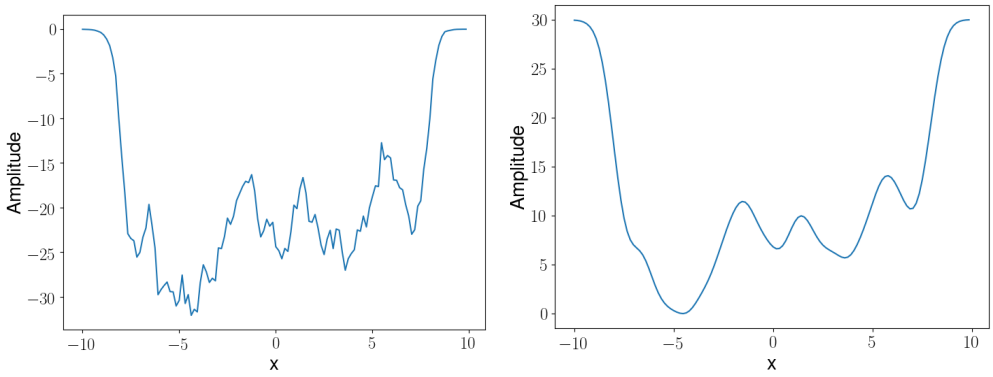


Figure 5: RandomPotential1 Before and After Process

The second one is summation of sines and cosines with random coefficients. Firstly, the number of terms are determined, after that each coefficient of the terms are assigned and summed iteratively.

Algorithm 2 RandomPotential2

procedure RANDOMPOTENTIAL2
  $Nterms = RandomInteger(1, 100)$
  for i $= 0$ **to** $Nterms$ **do**
    $A = GaussianDistributedRandomNumber()$
    $B = GaussianDistributedRandomNumber()$
    $n_1 = GaussianDistributedRandomNumber() * \sigma * \pi/width$
    $n_2 = GaussianDistributedRandomNumber() * \sigma * \pi/width$
    $Potential \mathrel{+}= A\sin(n_1 x) + B\cos(n_2 x)$
  $Potential = GaussianFilter(Potential, \sigma)$

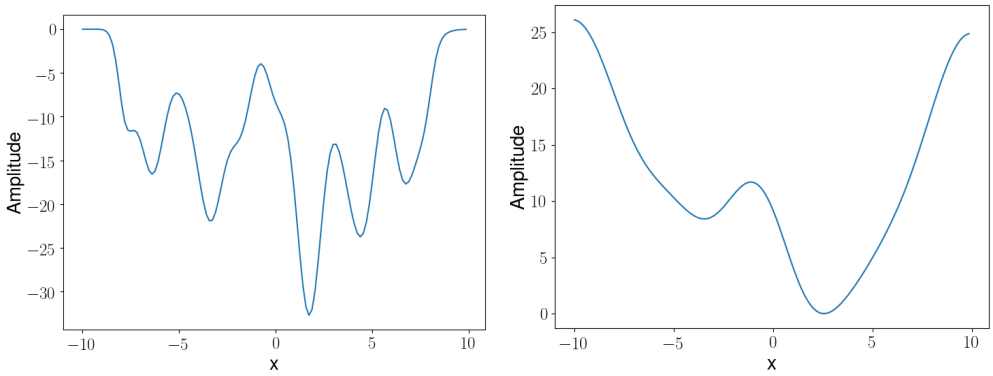


Figure 6: RandomPotential1 Before and After Process

The third one is one dimensional version of the algorithm described in [1] with slight modification. First a binary array of length 16 is generated by assigning random ones or zeros. Then the array is upscaled to 128 by repeating each element of the array 8 times. After that, another binary array of length 8 is generated with the same procedure and upscaled to 64 by repeating elements. Left and

right paddings of length 32 is added to the second binary array to be able to do element wise substraction between two binary array. Then, the second array is substracted from the first one to obtain the potential in binary form. Finally, a gaussian filter is applied to make the potential smooth.

Algorithm 3 RandomPotentia3

procedure RANDOMPOTENTIAL3
    $ScaleFactor = 8$
    for i $= 0$ **to** $NumberOfPoints/ScaleFactor$ **do**
        $BinaryGrid[i] = RandomInteger(0, 1)$
    $BinaryGrid = RepeatElements(BinaryGrid, ScaleFactor)$
    for i $= 0$ **to** $NumberOfPoints/(ScaleFactor * 2)$ **do**
        $BinaryGrid2[i] = RandomInteger(0, 1)$
    $BinaryGrid2 = RepeatElements(BinaryGrid, ScaleFactor)$
    $Padding = Zeros((Length(BinaryGrid) - Length(BinaryGrid2))/2)$
    $BinaryGrid2 = Concatanate(Padding, BinaryGrid2, Padding)$
    $Potential = BinaryGrid - BinaryGrid2$
    $Potential = GaussianFilter(Potential, \sigma)$

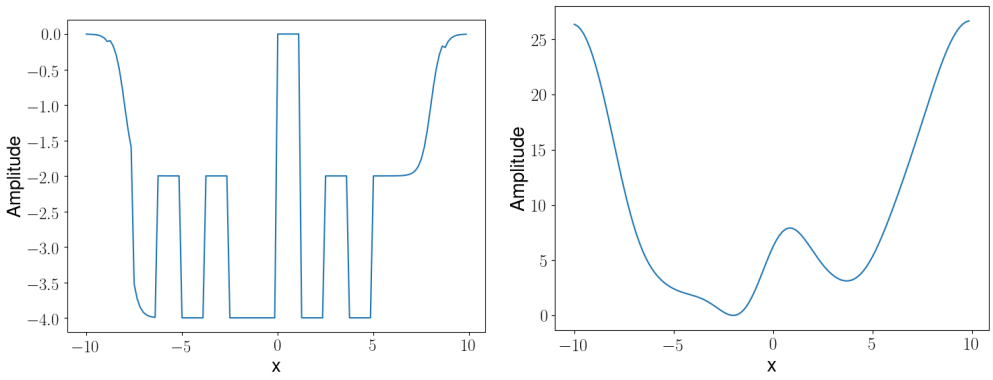


Figure 7: RandomPotential1 Before and After Process

2.8.7 Density and Ground State Energy

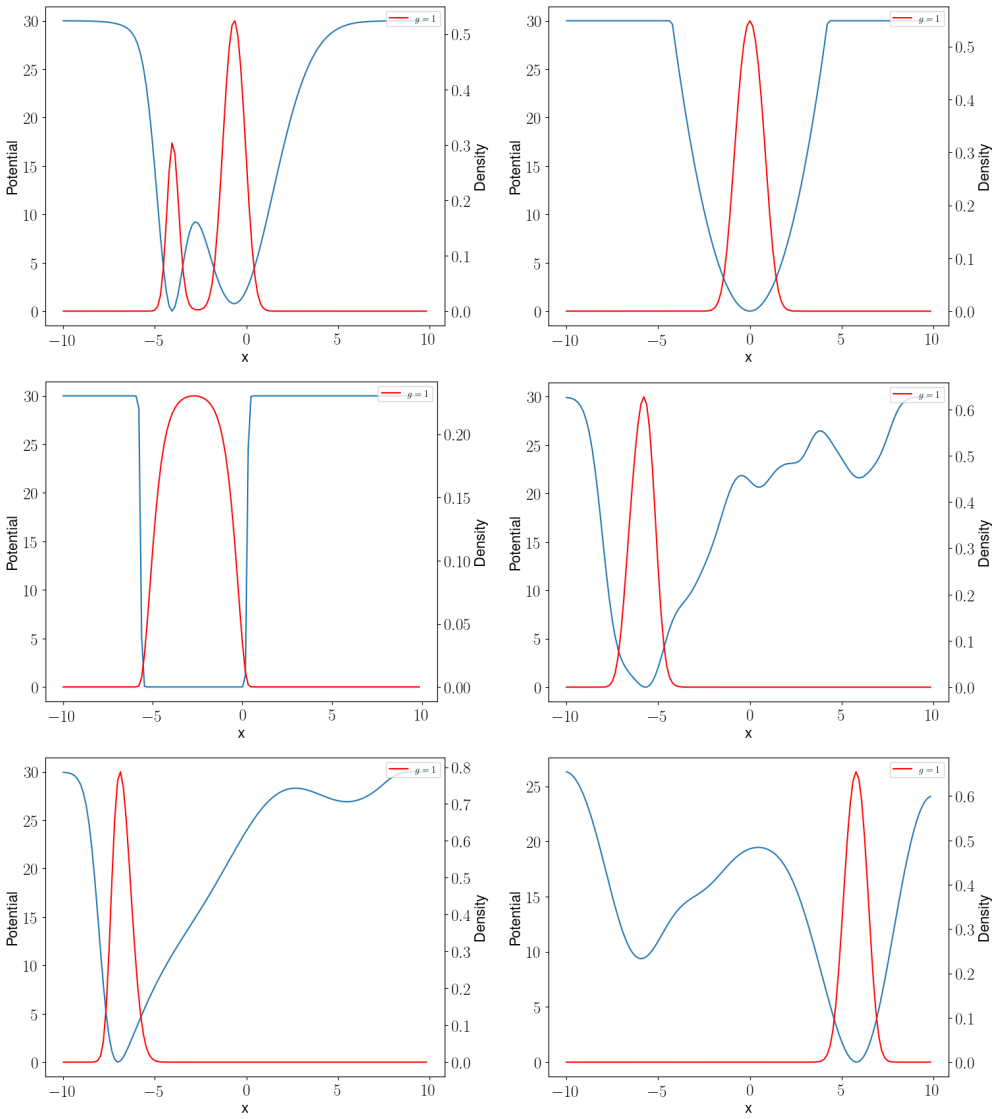


Figure 8: RandomPotential1 Before and After Process

2.8.8 Boundaries. (Table)

2.8.9 Convergence (detailed in APPENDIX)

2.8.10 Dataset generation. (Total number of examples etc)

2.9 Dataset Features

2.9.1 Energy distribution

3 Machine Learning

3.1 Network architecture

Architecture of the network.
A general figure like in the ML&SE article that describes the work done.
Another figure about internals of the network such as number of layers, how interaction parameter is introduced to the network etc.
Hyperparameters.

3.2 Training

Detailed info about dataset (energy distribution etc).
Indicate that if there is any method to increase the number of examples in low and high energy values.

3.3 Results

4 Inverse Problem

5 Conclusion and Discussion

5.1 Conclusion

5.2 Discussion

5.3 Effects of random potential generation method

5.4 Are there problems in low and high energies compared to the mean

5.5 Inverse problem

References

[1] K. Mills, M. Spanner, and I. Tamblyn, "Deep learning and the schrödinger equation," *Physical Review A*, vol. 96, no. 4, p. 042113, 2017.

A APPENDIX A

Obtaining harmonic trap potential scaling from Eq. (3)

$$V(z) \rightarrow V(\widetilde{z}) \rightarrow \widetilde{V}(\widetilde{z}) \tag{5}$$

$$\widetilde{V}(z) \equiv \frac{V(z)}{\gamma E_0} \tag{6}$$

$$\widetilde{z} \equiv \frac{z}{\beta L} \tag{7}$$

$$V(z) = \frac{1}{2} m\omega^2 (z - z_0)^2 \tag{8}$$

$$V(\widetilde{z}) = \frac{1}{2} m\omega^2 \beta^2 L^2 (\widetilde{z} - \widetilde{z_0})^2 \tag{9}$$

$$\widetilde{V}(\widetilde{z}) = \frac{1}{2} m\omega^2 \frac{\beta^2 L^2}{\gamma E_0} (\widetilde{z} - \widetilde{z_0})^2 \tag{10}$$

The coefficient of this equation must be dimensionless, therefore;

$$\frac{1}{2} m\omega^2 \frac{\beta^2 L^2}{\gamma E_0} = C \tag{11}$$

Where C is a positive constant. We know that $E_0 = \frac{\hbar^2}{2m}$ and the definition of α is given as,

$$\frac{\hbar^2}{2m\gamma E_0} \frac{1}{\beta^2 L^2} = \alpha \tag{12}$$

thus;

$$\frac{1}{\gamma} = \alpha \beta^2 L^2 \tag{13}$$

if we plug Eq. (13) in to Eq. (11), then equation becomes,

$$\frac{m^2 \omega^2}{\hbar^2} \alpha \beta^4 L^4 = C \tag{14}$$

in the case of expressing same physical system, m and ω must be constant. In

that case $\alpha = C(\beta L)^{-4}$.

In this case, α becomes

$$\alpha = \frac{1}{2}\left(\frac{\hbar\omega}{\gamma E_0}\right)^2$$

Conventionally, α is set to $1/2$, therefore;

$$\hbar\omega = \gamma E_0$$

$$\beta L = \sqrt{\frac{\hbar}{m\omega}}$$

βL is generally defined as harmonic oscillator length ℓ

$$\ell = \sqrt{\frac{\hbar}{m\omega}}$$

$$\widetilde{\mu} = \frac{\mu}{\hbar\omega}$$

$$\widetilde{g} = \frac{g}{\hbar\omega\ell}$$

Finally, dimensionless GPE scaled for harmonic trapping potential can be written as,

$$\widetilde{\mu}\widetilde{\psi} = -\frac{1}{2}\frac{d^2\widetilde{\psi}}{d\widetilde{z}^2} + \frac{1}{2}\widetilde{z}^2\widetilde{\psi} + \widetilde{g}|\widetilde{\psi}|^2\widetilde{\psi}$$