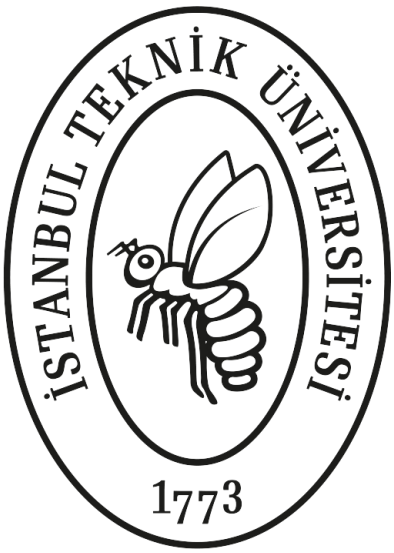
ISTANBUL TECHNICAL UNIVERSITY

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 different type of potentials including random. 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

In quantum mechanical systems one must obtain the wave function to determine physical properties of the system. For single particle systems, obtaining the wave function is easy compared to many body systems. The solution can be even harder if the described system involves interaction between particles because such differential equations may have nonlinear terms. In nonlinear case, there are no general rule to solve and they are treated individually in most cases. If there are no analytic solution exists and approximation is not on the table then the only chance is numerical solution of the equation to obtain the wave function, and then one can determine the physical properties of the system. However, machine learning applications in recent years showed that physical properties of the system can be predicted without solving these equations.

The reason why machine learning has such a capability is that artificial neural networks used in machine learning can approximate any continuous function within desired accuracy. This means that if we take $\epsilon > 0$ as desired accuracy, $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ (bold means vector) output of the network and $\boldsymbol{g}(\boldsymbol{x})$ real value of the function, then it is guaranteed that there exists a network that satisfies the relation $|\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x})| < \epsilon$. From this point of view, a process or calculation that can be thought of as a function can be represented by a corresponding neural network that mimics this function in desired accuracy [1]. With this in mind, neural networks have the potential to learn general functions and be exploited for their advantages. Approximate value of the quantity can be determined for different scenarios.

Many different kind of applications of machine learning have already been implemented in physics[1]. For example, In [3, 4] machine learning is applied to quantum many body problems. A machine learning method called Unsupervised Learning to detect patterns in big datasets is used for discovering phase transitions [5]. There are also developed techniques in machine learning inspired from physics such as quantum tensor networks [6]. Relation between physics and machine learning also caused foundation of a new branch called Quantum Machine Learning which aims to implement a quantum software to make machine learning faster than its classical version [7].

In [8], machine learning approach is applied to a 2D Schrödinger equation also known as Gross-Pitaevskii equation with random potentials. The authors built a

[1] For a detailed list, see [2]

convolutional deep neural network, and trained it to predict ground state energy of the system under four different confining potentials including random potential. Their study showed that machine learning is a promising alternative in electronic structure calculations of quantum systems. In our study inspired from the article mentioned above, we apply machine learning method to the nonlinear Schrödinger equation to predict ground state energy of a Bose-Einstein condensate with random interaction parameter at absolute zero temperature under six different one dimensional trapping potentials including random.

2 Gross Pitaevskii Equation

A Bose-Einstein Condensate (BEC) at zero temperature is described by Gross Pitaevskii equation (GPE) also known as non-linear Schrodinger Equation (NLSE). It is mean field approximation of a quantum many body system which the hamiltonian of the system is given by;

$$\hat{H} = \sum_{i=1}^{N} \left(\frac{\boldsymbol{p}_i^2}{2m} + V(\boldsymbol{r}_i) \right) + \frac{1}{2} \sum_{i=1}^{N} \sum_{j \neq i}^{N} U(|\boldsymbol{r}_i - \boldsymbol{r}_j|) \tag{1}$$

where $\mathbf{p}_i$ i^{th} atom's momentum, $\boldsymbol{r}$ is position vector, m is mass, V is external potential and U is interaction between i^{th} and j^{th} atoms. Since the ground state energy is the only possible minimum energy, one can minimize this hamiltonian in order to obtain ground state energy. To do that, the mean field approximation is utilized to represent all bosons with the same wave function since the condensate is at zero temperature which allows to assume that all bosons are at the ground state [9]. The constraint in the minimization step involves satisfying the normalization condition that is given as,

$$\int |\psi(\boldsymbol{r})|^2 \, \mathrm{d}^3\boldsymbol{r} = N \tag{2}$$

where N is the number of particles in the system. This condition equals to minimizing the thermodynamic potential, free energy $F = E - \mu N$ such that $\delta F = 0 = \delta E - \mu \delta N = 0$ where μ is chemical potential [9]. If we plug in the corresponding equations to this expression it becomes,

$$F(\psi) = \int \psi^* \hat{H} \psi \, \mathrm{d}^3\boldsymbol{r} - \mu \int |\psi|^2 \, \mathrm{d}^3\boldsymbol{r} \tag{3}$$

Therefore, the problem can be stated as minimization of the above equation [10]. Applying variation method to this equation while treating ψ^* and ψ as independent objects and using $U(|\mathbf{r}_i - \mathbf{r}_j|) = g\delta(\mathbf{r}_i - \mathbf{r}_j)$ one can obtain Gross

Pitaevskii Equation in stationary form as,

$$\frac{-\hbar^2}{2m}\nabla^2\psi + V(\boldsymbol{r})\psi + g|\psi|^2\psi = \mu\psi \tag{4}$$

where $\hbar$ is Planck constant, $\psi(\boldsymbol{r})$ is the wave function, ∇^2 is the Laplacian operator, g is interaction parameter and it is defined as

$$g = \frac{4\pi\hbar^2 a_s}{m} \tag{5}$$

where, a_s is the s wave scattering length. Nonlinearity of the equation is caused by the cubic term $g|\psi|^2$ which represents the interactions between bosons. The $|\psi|^2$ term is interpreted as density, thus; there occurs an energy contribution caused by the mean field interactions. If there is no interaction GPE reduces to the Schrödinger Equation (SE) and becomes a linear equation. By definition, g can be positive or negative. If $g > 0$, it represents repulsive interaction, and if $g < 0$, it represents attractive interactions [11]. The time evolution of the system must occur according to the Schrödinger Equation, therefore; the time dependent GPE can be written as,

$$i\hbar\frac{\partial\Psi(\boldsymbol{r},t)}{\partial t} = \hat{H}\Psi(\boldsymbol{r},t) \tag{6}$$

where t is time and $\Psi(\boldsymbol{r},t) = \psi(\boldsymbol{r})e^{-i\mu t/\hbar}$. If it is rewritten in open form,

$$i\hbar\frac{\partial\Psi(\boldsymbol{r},t)}{\partial t} = [\frac{-\hbar^2}{2m}\nabla^2 + V(\boldsymbol{r}) + g|\Psi(\boldsymbol{r},t)|^2]\Psi(\boldsymbol{r},t) \tag{7}$$

The ground state energy of the system is given in Eq. (3) which is,

$$\langle E\rangle = \int \Psi^*\hat{H}\Psi\,\mathrm{d}^3\boldsymbol{r} \tag{8}$$

$$E = \int\left(\frac{\hbar^2}{2m}|\nabla\Psi|^2 + V|\Psi|^2 + \frac{g}{2}|\Psi|^4\right)d^3\boldsymbol{r} \tag{9}$$

Here, the terms represent kinetic, potential and interaction energy respectively.[2] The potential does not depend on time so the hamiltonian does not too which means the total energy of the system is conversed.

[2]Contribution from the interaction energy is divided by two to eliminate double counting while pairing bosons.

2.1 Reduction of dimension.

2.2 Analytic solution and approximation.

There is no general solution of GPE. Known analytic solutions exist only for few cases and lack of analytic solution is also one of the main motivation of the application of machine learning technique. In our study, infinite well and harmonic potential with zero interaction parameter has analytic solutions and the ground states energies are given by, **WELL**

$$\mu = \frac{1}{2}\hbar\omega \tag{10}$$

respectively.
GPE generally solved by numerically or by approximation such as variational calculation or Thomas-Fermi approximation. **dilute** In the following section we briefly introduce Thomas-Fermi approximation and then we add comparison with numerical solutions.

2.2.1 Thomas Fermi Approximation

It is said that the second derivative term with respect to position represents the kinetic energy. When the potential and the interaction energy are dominant compared to the kinetic energy, the kinetic term can be neglected. This situation occurs when the condensate is large adequately and the interaction between bosons is repulsive [9]. In another perspective, neglecting the kinetic term equivalent to making an assumption that there is no difference between the energy requirements to add a particle at arbitrary points [10]. When the kinetic term is dropped, the new equation can be written as;

$$V(x)\psi(x) + g|\psi(x)|^2\psi(x) = \mu\psi(x) \tag{11}$$

this equation is analytically solvable and the solution is given by,

$$n(x) = |\psi(x)|^2 = \begin{cases} (\mu_{TF} - V)/g & \text{if } |x| \leq x_{TF} \\ 0 & \textit{otherwise} \end{cases} \tag{12}$$

where x_{TF} is called Thomas-Fermi Length and $n(x)$ is density. The density defined in this range and cannot be negative, therefore; if the normalization condition Eq. (2) is applied then the equation reads,

$$\int_{-\infty}^{\infty} |\psi|^2 \, \mathrm{d}x = \int_{-x_{TF}}^{x_{TF}} \frac{(\mu - V)}{g} \, \mathrm{d}x = N \tag{13}$$

With this equation, density of the system can be obtain via invoking boundary conditions. For an exact analytic expression example, one dimensional harmonic potential can be used,

$$\frac{1}{g}\left[\int_{-x_{TF}}^{x_{TF}} \mu \,\mathrm{d}x - \int_{-x_{TF}}^{x_{TF}} \frac{1}{2} m\omega^2 x^2 \,\mathrm{d}x\right] = N \tag{14}$$

$$\frac{2\mu x_{TF}}{g} - \frac{m\omega^2 x_{TF}^3}{3g} = N \tag{15}$$

From boundary conditions,

$$\mu = V(x_{TF}) = \frac{1}{2} m\omega^2 x_{TF}^2 \tag{16}$$

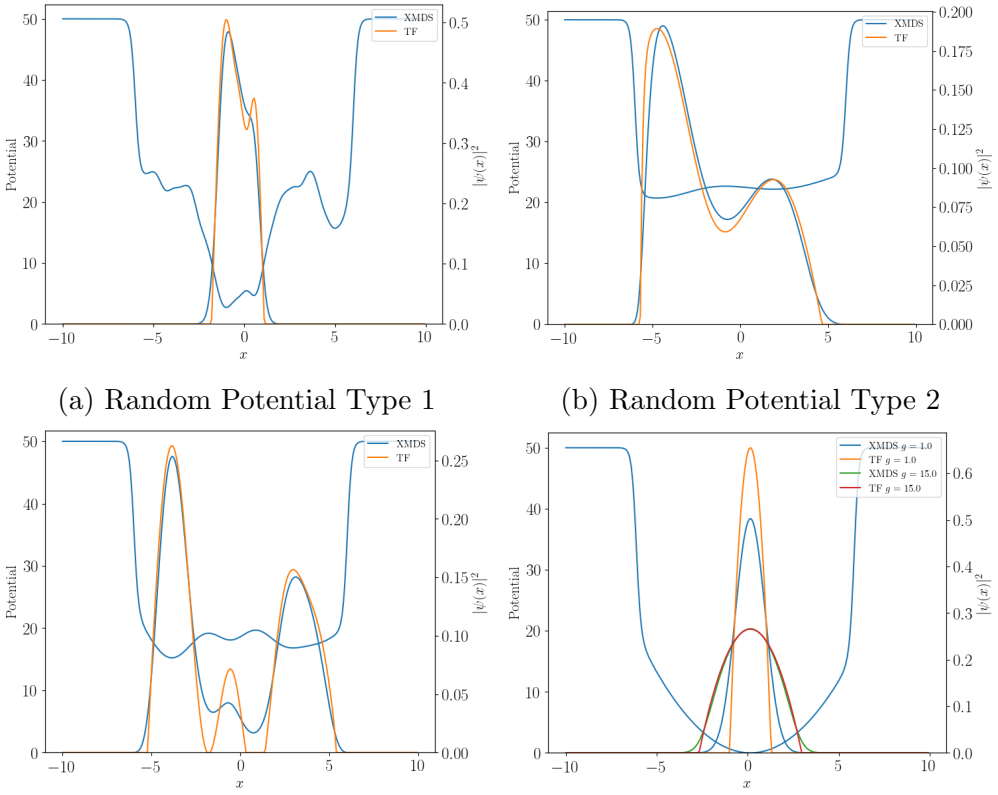
If we combine Eq. (15) and Eq. (16) the equations reads,

$$\frac{m\omega^2 x_{TF}^3}{g} - \frac{m\omega^2 x_{TF}^3}{3g} = N \tag{17}$$

$$\frac{4}{3}\left(\frac{2\mu}{m\omega^2}\right)^{1/2} \frac{\mu}{g} = N \tag{18}$$

$$\mu = \left(\frac{9}{32}(N\omega g)^2 m\right)^{1/3} \tag{19}$$

Here we give comparison of densities obtained by numerical solution and Thomas-Fermi approximation for different potential types.



(a) Random Potential Type 1

(b) Random Potential Type 2

(c) Random Potential Type 3

(d) Harmonic Potential

Figure 1: Here the potential types given in labels. The interaction parameter g of the random potentials is 15. The error between the numerical solution and approximation shrinks as the interaction parameter increases which corresponds to increment in interaction energy.

As shown in Fig 1d, the interaction parameter is great enough to use Thomas-Fermi approximation but since the density is determined by the Eq (13) the characteristic must be same with the potential used. This situation can be seen vividly in Fig 1c.

2.3 Numerical Solution and Dataset Generation

The dataset generation step is divided into two main parts and implemented independently. The first one is generating desired potential and the second one is numerical solution of the GPE under this potential by giving the generated potential to the numerical solution framework. The detailed description of the potential generation process is given in section 2.3.3.

In numerical solution part, we use a framework called XMDS [12], implemented specifically to solve differential equation systems with well optimized numerical methods. In this framework partial differential equation systems can be described by a markup language called XML. When equation system is declared properly,

XMDS produces a source code written in C++ that solves the equation with specified numerical method. Because of modularity in our implementation, the framework only solves the equation by supplied parameters, the framework does not generate anything internally such as potential, even the scaling factors are supplied externally. Such a modularity has an huge advantage such that changing numerical solution framework does not effect potential generation step, therefore; the work to change numerical solution framework is minimized. The only requirement is implementation of input output operations. By using this advantage, we also use another numerical solution framework called GPELab [13] implemented in Matlab to compare solutions' consistency and effect of scaling.

2.3.1 Scaling

The scaling of GPE is generally done according to potential type and there are more than one scaling conventions **REFS**. In this section we use a more general approach to scale GPE since the solutions will be numerical and we briefly would like to investigate how different scalings affect the solutions' consistency. To do that, the dimensionless quantities are introduced with variables so that scaling coefficient controlled by these variables.
First we define dimensionless potential, and then we make the length dimensionless,

$$\frac{-\hbar^2}{2m}\frac{d^2\psi}{dz^2} + V(x)\psi + g|\psi|^2\psi = \mu\psi \tag{20}$$

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

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

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

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

If these transformations are pluged into the Eq. (20) it becomes,

$$\frac{-\hbar^2}{2m\gamma E_0}\frac{1}{\beta^2 L^2}\frac{d^2\psi}{d\widetilde{x}^2} + \widetilde{V}(\widetilde{x})\psi + \frac{g}{\gamma E_0}|\psi|^2\psi = \frac{\mu}{\gamma E_0}\psi \tag{21}$$

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.

$$-\alpha\frac{d^2\widetilde{\psi}}{d\widetilde{x}^2} + \widetilde{V}(\widetilde{x})\widetilde{\psi} + \widetilde{g}|\widetilde{\psi}|^2\widetilde{\psi} = \widetilde{\mu}\widetilde{\psi} \tag{22}$$

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.

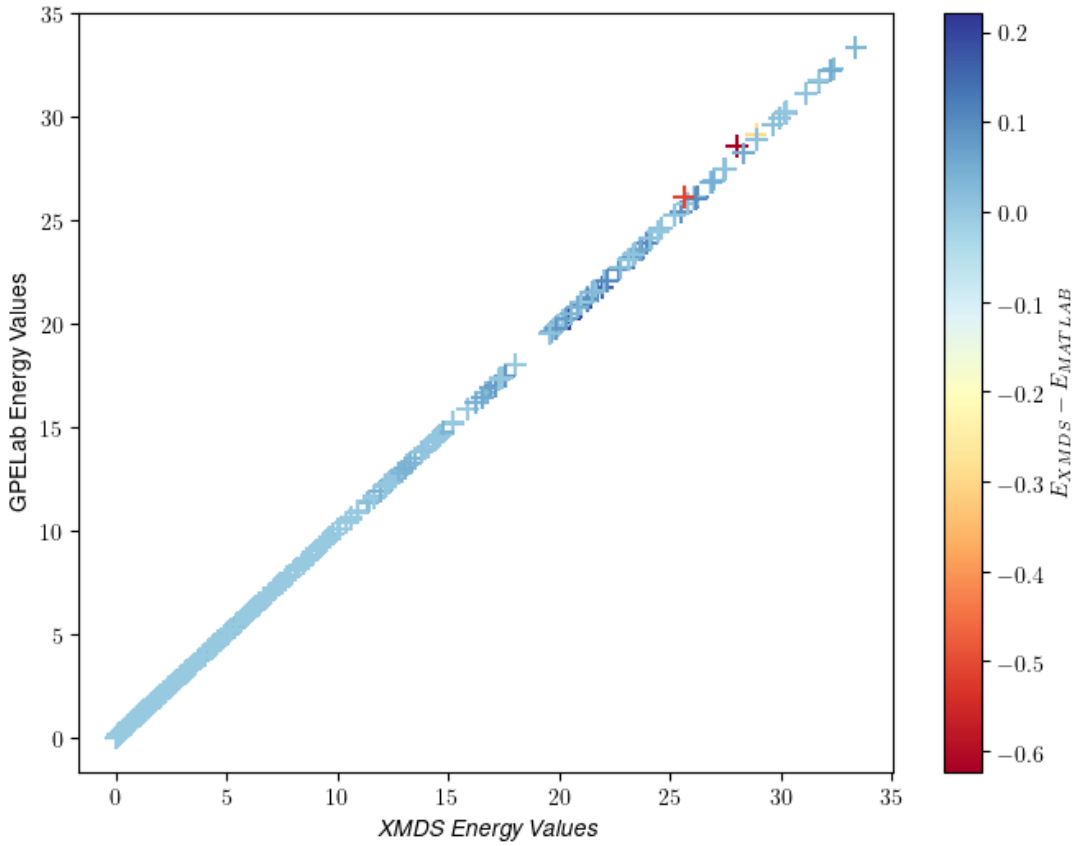


Figure 2: Error in density and energy

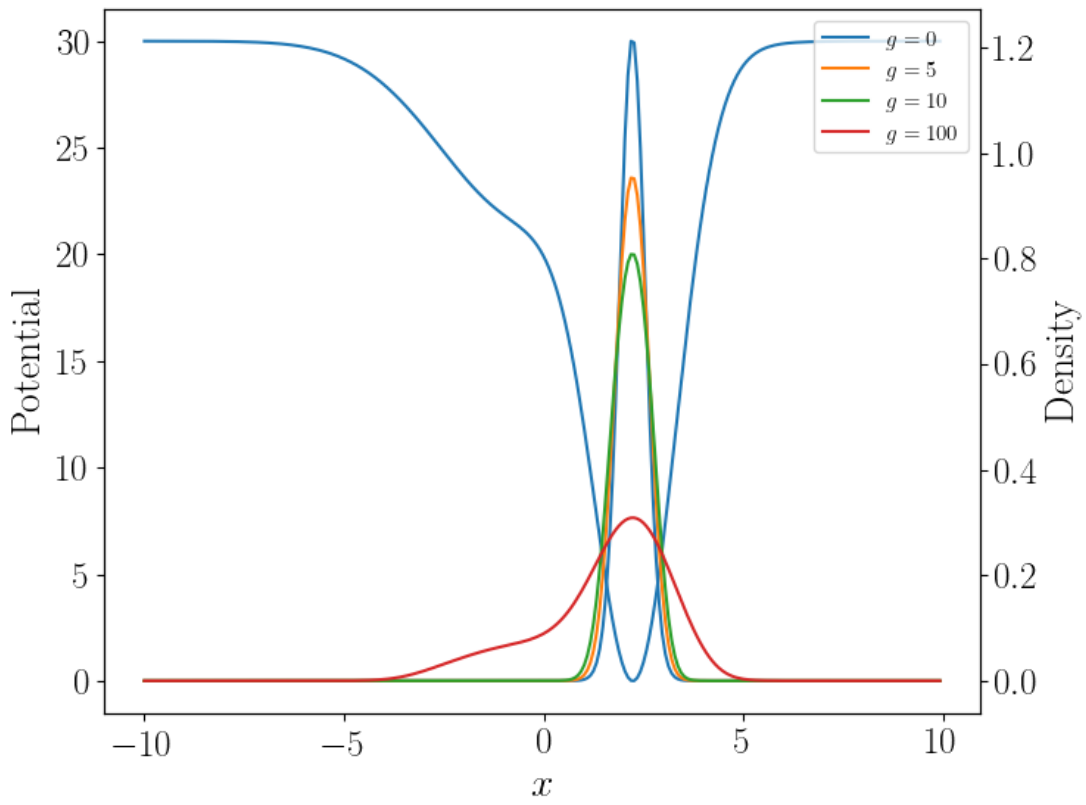


Figure 3: Density under Double Inverted Gaussian Potential with different interaction parameter.

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

2.3.3 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. A python class object responsible for handling this process. This class is constructed with suitable parameters such as "number of points", "width" etc. The generated potentials are sent to another method supplied by class. This method re-scales, and applies an envelope function to ensure that potential goes to numerical limit at boundaries which is given as,

$$\begin{aligned} V(x < x_l) &= V_0 \\ V(x > x_r) &= V_0 \end{aligned} \tag{23}$$

To handle these conditions, we define two envelope functions given by the following expressions,

$$\text{Env}_{LR}(x) = [(1 + \tanh(\beta(x + x_L))) + (1 - \tanh(\beta(x + x_R)))]/2 \tag{24}$$

$$\text{Env}_M(x) = 1 - \text{Env}_{LR}(x) \tag{25}$$

where x_L and x_R are bounds given in the Table (1) and the plot is given in Fig. 4.

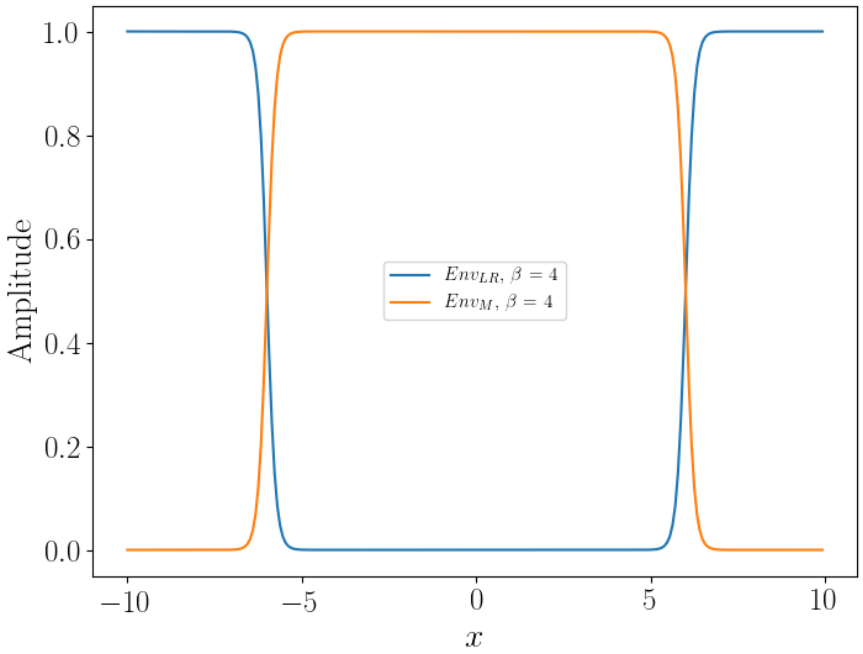


Figure 4: Envelope Function

As shown in Fig. 4 Env_M neutralizes the values while x goes to boundaries and only the region in the middle survives. For Env_{LR} the scenario is vice versa. Simultaneous application of these functions plus re-scaling brings the potential desired form shown in Fig. 5

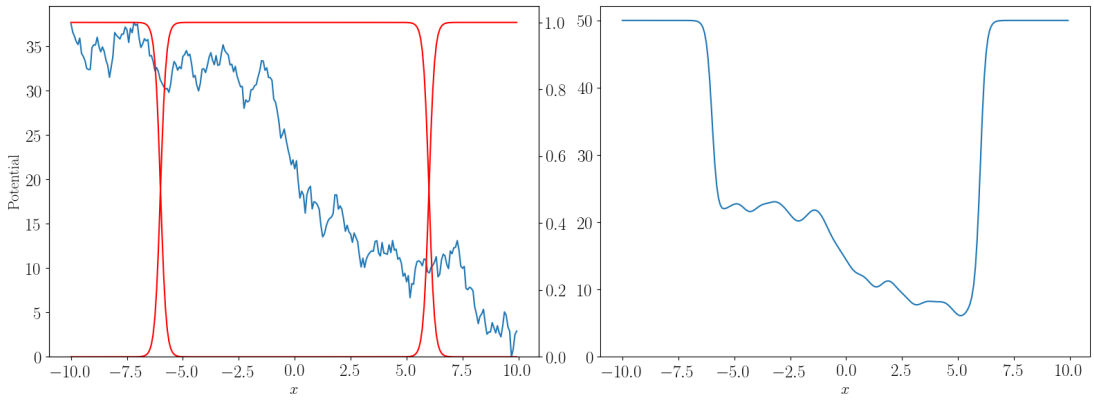


Figure 5: Envelope functions are applied to a random potential. After that, a gaussian filter is applied for smoothness.

2.3.4 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 [8]. We implement two more random potential generator to study effect of generation process. The generation processes of the random potentials are described in Section 2.3.5.

2.3.5 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**
        $Potential[i+1] = Potential[i] + Points[i]$
    $Potential = GaussianFilter(Potential, \sigma)$

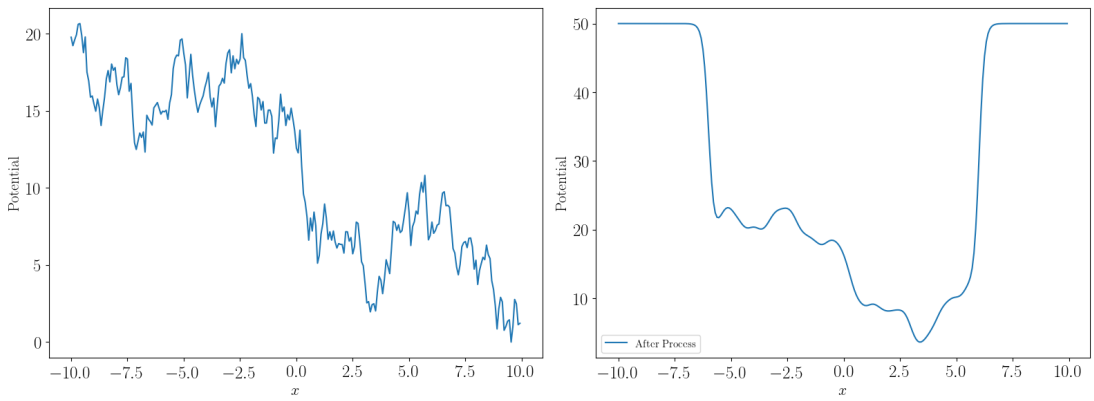


Figure 6: Random Potential 1 Before and After Process

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.

Table 1: Potentials

Potential	Analytic Form / Explanation	Parameters	Min	Max	Distribution
Envelope	$\text{Env}_{LR}(x) = [(1 + \tanh(\beta(x + x_L))) + (1 - \tanh(\beta(x + x_R)))]/2$ $\text{Env}_M(x) = 1 - \text{Env}_{LR}(x)$	x_L x_R β	-9 -4.5 -4.5	4.5 9 9	
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	
Random#1	$V(x_{i+1}) = V(x_i) + [X \sim \mathcal{N}(\mu, \sigma)]$	μ σ	-4 0.5	4 4	
Random#2	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	
Random#3	Substraction of two binary grid	Scale Factor	8	8	

Algorithm 2 RandomPotential2

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

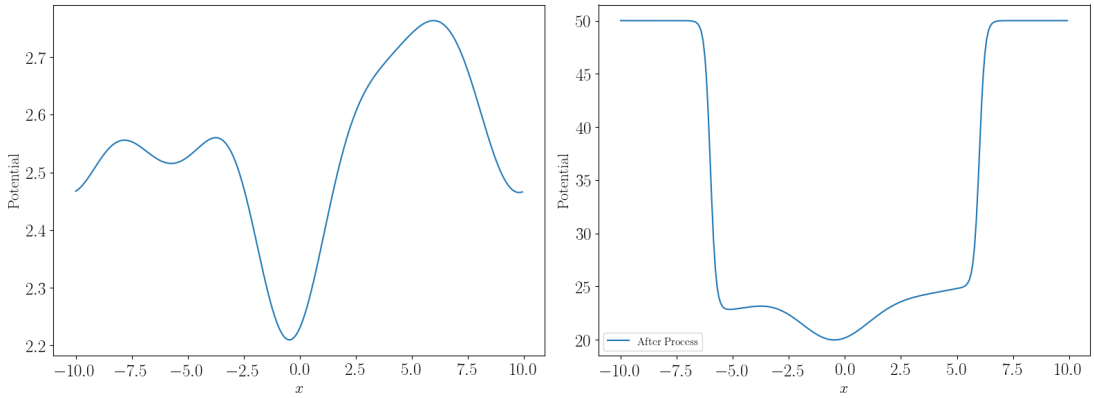


Figure 7: Random Potential 2 Before and After Process

The third one is one dimensional version of the algorithm described in [8] 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**
        $BinaryGrid2[i] = RandomInteger(0, 1)$
    $BinaryGrid2 = RepeatElements(BinaryGrid, ScaleFactor)$
    $Padding = Zeros((Length(BinaryGrid) - Length(BinaryGrid2))/2)$
    $BinaryGrid2 = Concatanate(Padding, BinaryGrid2, Padding)$
    $Potential = BinaryGrid - BinaryGrid2$
    $Potential = GaussianFilter(Potential, \sigma)$

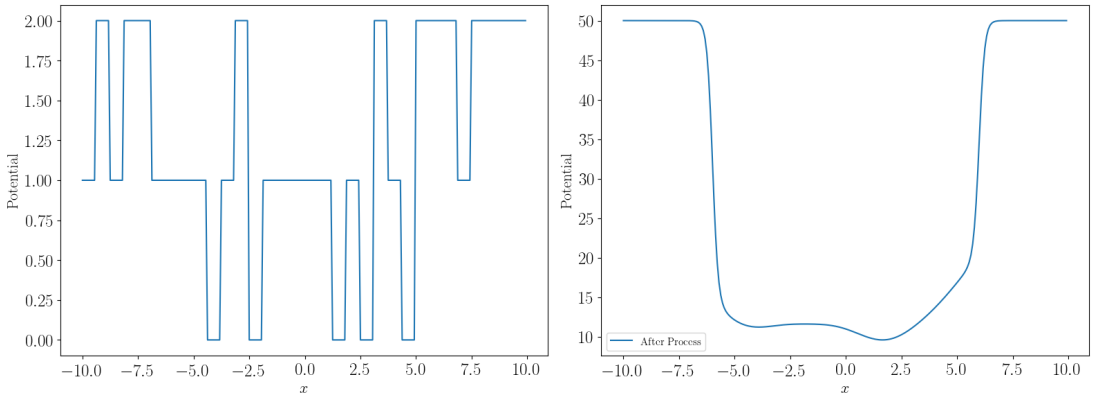


Figure 8: Random Potential 3 Before and After Process

2.3.6 Density and Ground State Energy

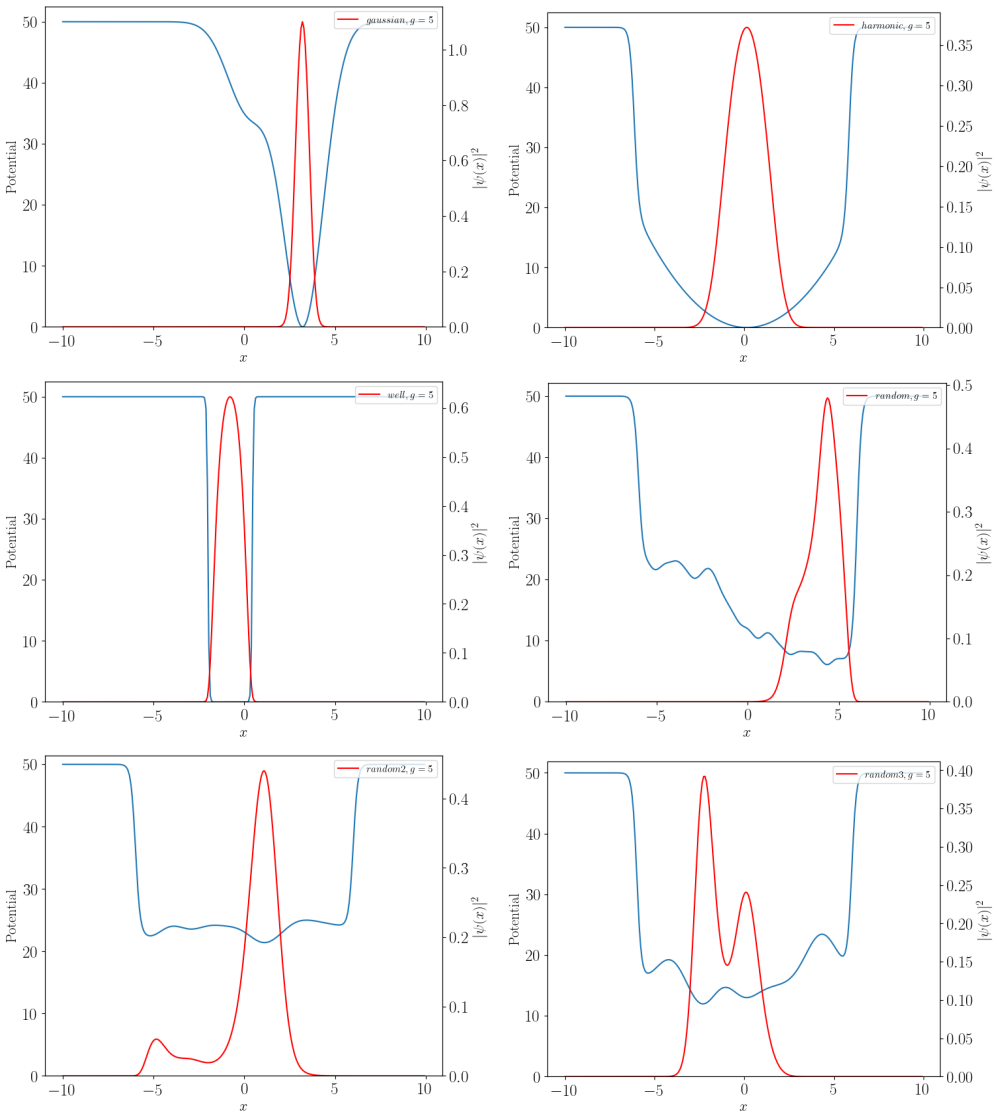


Figure 9: Potentials and Corresponding Ground State Densities

2.3.7 Boundaries. (Table)

2.3.8 Convergence (detailed in APPENDIX)

2.3.9 Dataset generation. (Total number of examples etc)

2.4 Dataset Features

2.4.1 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] M. A. Nielsen, "Neural networks and deep learning," 2015.

[2] PhysicsML, "Papers."

[3] G. Carleo and M. Troyer, "Solving the quantum many-body problem with artificial neural networks," *Science*, vol. 355, no. 6325, pp. 602–606, 2017.

[4] Z. Cai, "Approximating quantum many-body wave-functions using artificial neural networks," *arXiv preprint arXiv:1704.05148*, 2017.

[5] L. Wang, "Discovering phase transitions with unsupervised learning," *Physical Review B*, vol. 94, no. 19, p. 195105, 2016.

[6] E. M. Stoudenmire and D. J. Schwab, "Supervised learning with quantum-inspired tensor networks," *arXiv preprint arXiv:1605.05775*, 2016.

[7] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning, 2016," *arXiv preprint arXiv:1611.09347*.

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

[9] C. J. Pethick and H. Smith, *Bose-Einstein condensation in dilute gases*. Cambridge university press, 2002.

[10] J. Rogel-Salazar, "The gross–pitaevskii equation and bose–einstein condensates," *European Journal of Physics*, vol. 34, no. 2, p. 247, 2013.

[11] C. F. Barenghi and N. G. Parker, *A primer on quantum fluids*. No. arXiv: 1605.09580, Springer, 2016.

[12] G. R. Dennis, J. J. Hope, and M. T. Johnsson, "Xmds2: Fast, scalable simulation of coupled stochastic partial differential equations," *Computer Physics Communications*, vol. 184, no. 1, pp. 201–208, 2013.

[13] X. Antoine and R. Duboscq, "Gpelab, a matlab toolbox to solve gross–pitaevskii equations i: Computation of stationary solutions," *Computer Physics Communications*, vol. 185, no. 11, pp. 2969–2991, 2014.

A APPENDIX A

Obtaining harmonic trap potential scaling from Eq. (22)

$$V(x) \to V(\widetilde{x}) \to \widetilde{V}(\widetilde{x}) \tag{26}$$

$$\widetilde{V}(x) \equiv \frac{V(x)}{\gamma E_0} \tag{27}$$

$$\widetilde{x} \equiv \frac{x}{\beta L} \tag{28}$$

$$V(x) = \frac{1}{2} m\omega^2 (x - z_0)^2 \tag{29}$$

$$V(\widetilde{x}) = \frac{1}{2} m\omega^2 \beta^2 L^2 (\widetilde{x} - \widetilde{z_0})^2 \tag{30}$$

$$\widetilde{V}(\widetilde{x}) = \frac{1}{2} m\omega^2 \frac{\beta^2 L^2}{\gamma E_0} (\widetilde{x} - \widetilde{z_0})^2 \tag{31}$$

The coefficient of this equation must be dimensionless, therefore;

$$\frac{1}{2} m\omega^2 \frac{\beta^2 L^2}{\gamma E_0} = C \tag{32}$$

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

$$\frac{\hbar^2}{2m\gamma E_0} \frac{1}{\beta^2 L^2} = \alpha \tag{33}$$

thus;

$$\frac{1}{\gamma} = \alpha \beta^2 L^2 \tag{34}$$

if we plug Eq. (34) in to Eq. (32), then equation becomes,

$$\frac{m^2 \omega^2}{\hbar^2} \alpha \beta^4 L^4 = C \tag{35}$$

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,

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