

Article

Using Artificial Neural Networks to Solve the Gross–Pitaevskii Equation

Ioannis G. Tsoulos ^{1,*} , Vasileios N. Stavrou ²  and Dimitrios Tsalikakis ³

¹ Department of Informatics and Telecommunications, University of Ioannina, 451 10 Ioannina, Greece

² Division of Physical Sciences, Hellenic Naval Academy, Military Institutions of University Education, 185 39 Piraeus, Greece; vstavrou@hna.gr

³ Department of Engineering Informatics and Telecommunications, University of Western Macedonia, 501 00 Kozani, Greece; dtsalikakis@uowm.gr

* Correspondence: itsoulos@uoi.gr

Abstract: The current work proposes the incorporation of an artificial neural network to solve the Gross–Pitaevskii equation (GPE) efficiently, using a few realistic external potentials. With the assistance of neural networks, a model is formed that is capable of solving this equation. The adaptation of the parameters for the constructed model is performed using some evolutionary techniques, such as genetic algorithms and particle swarm optimization. The proposed model is used to solve the GPE for the linear case ($\gamma = 0$) and the nonlinear case ($\gamma \neq 0$), where γ is the nonlinearity parameter in GPE. The results are close to the reported results regarding the behavior and the amplitudes of the wavefunctions.

Keywords: Gross–Pitaevskii equation; simulations; machine learning; neural

MSC: 65K10; 68T07



Citation: Tsoulos, I.G.; Stavrou, V.N.; Tsalikakis, D. Using Artificial Neural Networks to Solve the Gross–Pitaevskii Equation. *Axioms* **2024**, *13*, 711.

<https://doi.org/10.3390/axioms13100711>

Academic Editor: Simeon Reich

Received: 21 August 2024

Revised: 1 October 2024

Accepted: 11 October 2024

Published: 15 October 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Bose–Einstein condensates (BECs) have become a subject of research in many areas of physics, quantum optics, condensed matter physics, and atomic physics, among others. The Gross–Pitaevskii equation has been successfully used to study a wide range of Bosonic systems. Various analytical and numerical solutions have been employed to estimate the energy eigenstates and the eigenvalues of the GPE for several applicable physical potentials. The BECs in a box, the gravitational surface trap, the harmonic trap, the double-Well Potential and the Jacobian elliptic potentials are a few potentials [1–5] that have been used in the above-mentioned research areas. Also, recently, artificial neural networks (ANNs) and physics-informed neural networks (PINNs) have been used to estimate the condensate wavefunctions described by the Gross–Pitaevskii equation in one-dimensional (1D) physical structures [6,7]. ANNs and PINNs give competitive advantages against other numerical techniques due to their fast solvability and the reliability of the results in the case of nonlinear differential equations (like the Gross–Pitaevskii equation). Furthermore, in recent decades, several experiments have been performed and compared to theoretical predictions. For instance, the experimental measurements on BECs of trapped alkali–metal vapors [8,9] are in good agreement with the estimated results created by the Gross–Pitaevskii theory.

This article considers hybrid models that utilize neural networks [10,11] to solve the previously mentioned Gross–Pitaevskii equation for various potentials. Neural networks have been employed in many real-world problems in a variety of scientific areas, such as problems from physics [12–14], solving differential Equations [15,16], agriculture problems [17,18], problems derived from chemistry [19–21], problems found in economics [22–24], problems in medicine [25,26], etc. A common way to express neural

networks is as functions $N(\vec{x}, \vec{w})$, where the vector $\vec{x}$ stands for the input pattern and the vector $\vec{w}$ defines the vector of parameters under estimation, which is usually called a weight vector. According to the authors of [27], a neural network can be defined as the following summation:

$$N(\vec{x}, \vec{w}) = \sum_{i=1}^K w_{(d+2)i-(d+1)} \sigma \left(\sum_{j=1}^d x_j w_{(d+2)i-(d+1)+j} + w_{(d+2)i} \right). \quad (1)$$

Neural networks have also been incorporated in the past to solve differential equations. For example, consider the work of Lagaris et al. [28], where different models based on neural networks were incorporated for a series of differential equation cases. Aarts and de Ver [29] proposed a method that utilizes neural networks to model and solve partial differential equations. Also, Parisi et al. have proposed [30] a method that trains neural networks with genetic algorithms in order to solve differential equations. Tsoulos et al. [31] suggested a method that constructs neural networks with the assistance of grammatical evolution [32] to solve differential equations. A review of methods that involve neural networks for the solutions of differential equations can be found in the work of Kumar et al. [33]. Recently, Zhang and Li [34] proposed a scheme that incorporates artificial neural networks to solve Caputo fractional-order differential equations.

Furthermore, artificial neural networks have also been used in a variety of problems derived from physics, such as quantum mechanics problems [35], problems that appear in particle physics [36], solutions for the quantum many-body problem [37], heat transfer problems [38], etc.

The parameter K represents the number of processing nodes and the parameter d stands for the dimension of vector $\vec{x}$. The function $\sigma(x)$ is commonly known as the sigmoid function, and it is defined as follows:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}. \quad (2)$$

The estimation of the optimal set of parameters for a neural network is also called training of the neural network, and a variety of methods have been proposed in the recent literature for training neural networks. These methods include the Back Propagation method [39,40], the RPROP method [41,42], the ADAM optimizer [43], etc. Also, evolutionary techniques that are considered a group of global optimization methods [44,45] have been used to train neural networks. A characteristic that distinguishes evolutionary techniques is that they try to imitate natural processes through vector operations. Furthermore, as optimization methods are global, they can avoid local minima of the objective function by aiming for the global minimum of the function. As far as artificial neural networks are concerned, this behavior results in finding lower values of the neural network error. In the relevant literature, there are several research publications that focus on the application of evolutionary techniques in the training of artificial neural networks, such as the application of genetic algorithms [46], the incorporation of the particle swarm optimization method [47,48], the use of the differential evolution method [49,50], ant colony optimization [51], the gray wolf optimizer [52], the whale optimization method [53], etc. In this article, modified versions of two evolutionary techniques were incorporated to solve the Gross–Pitaevskii equation with a model that utilizes an artificial neural network. The above two techniques were chosen for the optimization of the proposed model as they have proven their value in a number of difficult optimization problems. Also, since artificial neural networks have an extended set of local minima using their ability to produce candidate solutions that are modified with natural processes, these local minima can be avoided by aiming for an optimal configuration of the artificial neural network parameters.

The proposed technique utilizes two global optimization techniques for numerical solutions to the equation. Both techniques create populations of candidate solutions that are combined with each other in a series of steps, with the ultimate goal of solving the

equation numerically. These candidate solutions include the parameters of an artificial neural network-based model. Given the ability of artificial neural networks to adapt to each function, but also with the use of well-tested global optimization techniques, greater certainty is provided for the numerical solution of the equation.

The rest of this paper is organized in the following sections: in Section 2, the proposed models utilized here and the associated optimization methods are presented; in Section 3, the experimental results are shown and discussed; and finally, in Section 4, some conclusions are presented.

2. The Proposed Method

This section begins with the description of the used model and continues with the definition of the optimization methods utilized to train this model. The method of artificial neural networks can solve problems faster, whereas analytical techniques do not have significant success. It is worth mentioning that the linear and nonlinear solutions of GPE can describe different limits of physical systems, e.g., Bose–Einstein condensates or any other physical system. Furthermore, the eigenfunctions can have zero values at specific boundary points due to the physical properties of the system under investigation. Lastly, the external potential influences the GPE solution by introducing different terms in the GPE equation and, as a result, they describe different physical systems.

2.1. The GPE Equation

We solve the eigenvalue problem in the case of a nonlinear Schrödinger equation (GPE equation). For the case of a one-dimensional problem, the ground states and higher modes of a Bose–Einstein condensate (BEC) in a time-independent external potential are described by

$$\left(-\frac{\hbar^2}{2m}\Delta + V_{ext}(x) + g_s|\Psi(x)|^2\right)\Psi(x) = \mu\Psi(x) \quad (3)$$

where m , μ , g_s , and V_{ext} are the mass of the atom, the chemical potential, the non-linearity parameter and an external potential, respectively. The above-mentioned equation is time-independent GPE which describes the dynamics of a BEC with the nonlinear term g_s .

Considering that the wavefunction $\Psi(x)$ is normalized to one, the GPE equation can be written as:

$$\left(-\frac{d^2}{dx^2} + \tilde{V}_{ext}(x) + \gamma\Psi^2(x)\right)\Psi(x) = \epsilon\Psi(x) \quad (4)$$

where $x \in [a, b]$ and we use the rescaled values (L natural length scale, $x \rightarrow xL$ and $\Psi \rightarrow \sqrt{N}\Psi/L^{3/2}$). The length scale is arbitrary and may depend on various physical parameters. Furthermore, the dimensional quantities are described by the following relations:

$$\begin{aligned} \tilde{V}_{ext}(x) &= \frac{2mL^2V_{ext}}{\hbar}, \\ \gamma &= \frac{2Nmgs}{L\hbar^2}, \\ \epsilon &= \frac{2m\mu L^2}{\hbar^2} \end{aligned} \quad (5)$$

In the current investigation, we assume that the potential can be described by three distinct models: Bose–Einstein condensates, gravitational trap and harmonic trap.

2.1.1. Bose–Einstein Condensates (BEC) in a Box

Here, the potential is provided from the following equation:

$$\tilde{V}_{ext}(x) = \begin{cases} 0 & \text{if } x \in [0, 1] \\ \infty & \text{otherwise} \end{cases} \quad (6)$$

with standard boundary conditions ($\psi_n(0) = 0$ and $\psi_n(1) = 0$) and natural length scale $L = \hbar / \sqrt{2m\mu}$ and “n” is the mode of the solution of the wavefunction.

2.1.2. Gravitational Trap

The used external potential (gravitational) has the following form:

$$\tilde{V}_{ext}(x) = \begin{cases} mgx & \text{if } x > 0 \\ \infty & \text{otherwise} \end{cases} \quad (7)$$

where m is the mass of the atom and g is the gravitational acceleration. The boundary conditions are $\Psi_n(0) = 0$ and $\Psi_n(\infty) = 0$ and Equation (4) yields

$$\left(-\frac{d^2}{dx^2} + x + \gamma \Psi_n^2(x) \right) \Psi_n(x) = \varepsilon_n \Psi_n(x) \quad (8)$$

with natural length scale $L = (\hbar^2 / 2m^2g)^{1/3}$.

2.1.3. Harmonic Trap

In this case, the potential is defined as:

$$\tilde{V}_{ext}(x) = \frac{1}{2} \omega^2 x^2 \quad (9)$$

The wavefunctions vanish as x reaches ∞ and Equation (4) yields

$$\left(-\frac{d^2}{dx^2} + x^2 + \gamma \Psi_n^2(x) \right) \Psi_n(x) = \varepsilon_n \Psi_n(x) \quad (10)$$

with natural length scale $L = \sqrt{\hbar / m\omega}$.

2.2. The Used Model

The used model is defined as:

$$\Psi_n(x) = x(1-x)N(\vec{w}, x) \quad (11)$$

where $N(\vec{w}, x)$ stands for an artificial neural network. This model is used for the case of BEC potential since it vanishes at $x = 0$ and $x = 1$. Models with similar properties incorporating a neural network have also been utilized in the other two potentials. The differential equation presented in Equation (4) is solved in current work by dividing the space $[a, b]$ into m equidistant points. These points form the set $T = [x_0 = a, x_1, x_2, \dots, x_m = b]$. Hence, by applying the model of Equation (11) to every point of the set, the following function should be minimized in order to numerically solve the differential equation of (4):

$$\min_{\vec{w}} \sum_{i=1}^m \left(\left(-\frac{d^2}{dx^2} + V_{ext}(x_i) + \gamma \Psi_n^2(x_i) \right) \Psi_n(x_i) - \varepsilon_n \Psi_n(x_i) \right)^2 \quad (12)$$

In order to minimize the previously defined equation, two distinct evolutionary techniques will be used: a modified version of the genetic algorithm and a variant of the particle swarm optimization method.

2.3. The Used Genetic Algorithm

A modified version of a genetic algorithm is proposed to solve the differential Equation (11). Initially, genetic algorithms were proposed by John Holland [54] and they are inspired by biology. The main genetic algorithm starts by creating an initial population of potential solutions for the current optimization problem. The potential solutions are called chromosomes, and they change repeatedly through a series of iterations, also called

generations. In every generation, biologically inspired operations are applied on the chromosomes, such as selection, crossover, and mutation [55]. Genetic algorithms have been applied to a wide series of optimization and practical problems from the relevant literature, such as electromagnetics [56], placement of wind turbines [57], feature selection [58], power flow [59], transportation scheduling [60], etc. Furthermore, genetic algorithms have been incorporated to train neural networks in a series of recent works, such as the paper of Leung et al. [61], where they have been utilized to construct the structure of neural networks. Moreover, they have been applied to the problem of rainfall-runoff forecasting [62] by training neural networks, and they have also been utilized to evolve neural networks for the deformation modulus of rock masses [63], etc.

The main steps of the used modified genetic algorithm are listed below.

1. Initialization Step

- (a) **Define** N_C as the number of chromosomes and N_G as the number of allowed generations.
- (b) **Set** K as the number of processing nodes (weights) for the artificial neural network.
- (c) **Randomly** produce N_C chromosomes. Every chromosome is considered a potential set of parameters for the artificial neural network.
- (d) **Set** p_s as the selection rate and p_m as the mutation rate with the assumption that $p_s \leq 1$ and $p_m \leq 1$.
- (e) **Set** $\text{iter} = 0$ as the generation number.

2. Fitness calculation

- (a) **For** $i = 1, \dots, N_C$, **do**
 - i. **Create** the neural network $N(x, \vec{g}_i)$ for chromosome g_i .
 - ii. **Create** the model $\Psi_n(x, g_i)$ for chromosome g_i as

$$\Psi_n(x, g_i) = x(x-1)N(x, \vec{g}_i)$$

using Equation (11).

- iii. **Calculate** the fitness f_i as

$$f_i = \sum_{j=1}^m \left(\left(-\frac{d^2}{dx^2} + \text{Vext}(x_j) + \gamma \Psi_n^2(x_j, g_i) \right) \Psi_n(x_j, g_i) - \varepsilon_n \Psi_n(x_j, g_i) \right)^2 \quad (13)$$

- (b) **EndFor**

3. Genetic operations

- (a) **Selection procedure.** Initially, the chromosomes are sorted according to their fitness, the first $(1 - p_s) \times N_C$ are copied without changes to the next generation and the remaining are substituted by chromosomes that will be produced in the crossover procedure.
- (b) **Crossover procedure.** During this step, for every couple $\tilde{z}, \tilde{w}$ of produced offsprings, two chromosomes denoted as z and w are selected from the current population using tournament selection. The new chromosomes are constructed using the following equations:

$$\begin{aligned} \tilde{z}_i &= a_i z_i + (1 - a_i) w_i \\ \tilde{w}_i &= a_i w_i + (1 - a_i) z_i \end{aligned} \quad (14)$$

where a_i is a random number with the property $a_i \in [-0.5, 1.5]$ [64].

- (c) **Mutation procedure.** In this procedure, a random number r is selected for each element j of every chromosome g_i . If $r \leq p_m$, then the corresponding element $g_{i,j}$ is changed randomly using the following scheme [64]:

$$g_{i,j} = \begin{cases} g_{i,j} + \Delta(\text{iter}, r_j - g_{i,j}) & t = 0 \\ g_{i,j} - \Delta(\text{iter}, g_{i,j} - l_j) & t = 1 \end{cases} \quad (15)$$

where t is a random number that accepts the values 0 or 1. The function $\Delta(\text{iter}, y)$ is defined as:

$$\Delta(\text{iter}, y) = y \left(1 - r \left(1 - \frac{\text{iter}}{N_G} \right)^b \right) \quad (16)$$

The number r is a random number in $[0, 1]$ and the parameter b controls the magnitude of change in the chromosome.

4. Termination Check

- (a) **Set** $\text{iter} = \text{iter} + 1$
(b) A termination rule that was initially introduced in the work of Tsoulos [65] is used here.

At each iteration iter , the algorithm calculates the variance, denoted as $\sigma^{(\text{iter})}$, of the best fitness value up to that iteration. This variance will continually drop as the algorithm will discover lower values or these values will remain constant. In this case, the variance for the last iteration where a lower value was discovered is recorded, and if the variance falls below half of this record, then there is probably no new minimum for the algorithm to find and therefore it should terminate. The suggested termination rule is defined as:

$$\text{iter} \geq N_G \text{ OR } \sigma^{(\text{iter})} \leq \frac{\sigma^{(\text{klast})}}{2} \quad (17)$$

where klast is the last generation where a new lower value for the best fitness was located.

- (c) If the termination rule does not hold, then go to step 2.

5. Application of local optimization

- (a) **Obtain** the chromosome with the lowest fitness value in the population and denote it as g_{best} .
(b) **Apply** a local search procedure to chromosome g_{best} . In the current work, a Broyden–Fletcher–Goldfarb–Shannon (BFGS) variant of Powell [66] was used. The use of local minimization techniques is important in order to reliably find a true local minimum from the global optimization method. In the optimization theory, the region of attraction $A(x^*)$ of some local minimum x^* is defined as:

$$A(x^*) = \{x : x \in S, L(x) = x^*\} \quad (18)$$

where $L(x)$ stands for the used local optimization method. Global optimization methods, in most cases, identify points that are inside the region of attraction $A(x^*)$ of a local minimum x^* but not necessarily the actual local minimum. For this reason, it is considered necessary to have a combination of the global optimization method with local optimization techniques. Moreover, according to the relevant literature, the error function of artificial neural networks may contain a very large number of local minimums, making the aforementioned combination important in the process of finding an optimal set of parameters for the artificial neural network.

2.4. The Used PSO Variant

Particle swarm optimization (PSO) [67] is another evolutionary technique used to tackle global optimization problems. The PSO method forms a population of candidate solutions, called particles. These particles are evolved through a series of steps until some stopping criteria are met. Two vectors are used in the PSO method: the current position of the particle and the velocity of each particle, denoted as $\vec{p}$ and $\vec{u}$, respectively. The PSO method was tested on a wide series of problems from various areas, such as physics [68,69], chemistry [70,71], medicine [72,73], economics [74], etc. Additionally, the PSO technique was applied successfully on neural network training [75]. The proposed work utilizes an implementation of PSO, as suggested by Charilolis and Tsoulos [76] and this implementation is used to minimize the function of Equation (12). The method has the following steps:

1. Initialization

- (a) **Set** $\text{iter} = 0$ as the iteration counter.
- (b) **Define** N_C as the number of particles and N_G as the number of allowed generations.
- (c) **Set** $p_l \in [0, 1]$ as the local search rate.
- (d) **Set** the parameters c_1, c_2 in the range $[1, 2]$.
- (e) **Initialize** the m particles $p_1, p_2, \dots, p_{N_C}$. Each particle represents a set of parameters for the artificial neural network.
- (f) **Randomly** initialize the corresponding velocities $u_1, u_2, \dots, u_{N_C}$.
- (g) **For** $i = 1, \dots, N_C$, carry out $\text{pb}_i = p_i$. The vector pb_i denotes the best-located position for the particle p_i .
- (h) **Set** $p_{\text{best}} = \arg \min_{i \in 1 \dots N_C} f(p_i)$, where the function $f(p)$ denotes the fitness value of particle p .

2. Check for termination. The same termination scheme as in the genetic algorithm is used here.

3. For $i = 1, \dots, N_C$, Do

- (a) **Compute** the velocity u_i as:

$$u_i = \omega u_i + r_1 c_1 (\text{pb}_i - p_i) + r_2 c_2 (p_{\text{best}} - p_i) \quad (19)$$

with

- i. The parameters r_1, r_2 are randomly selected numbers in $[0, 1]$.
- ii. The variable ω stands for the inertia value and is computed as:

$$\omega_{\text{iter}} = 0.5 + \frac{r}{2} \quad (20)$$

where r denotes a random number with the property $r \in [0, 1]$ [77]. The above inertia calculation mechanism gives the particle optimization method the ability to more efficiently explore the search space of the objective function and consequently identify regions where the artificial neural network parameters will more effectively solve the equation.

- (b) **Compute** the new position of the particle as $p_i = p_i + u_i$
- (c) **Draw** a random number $r \in [0, 1]$. If $r \leq p_l$, then $p_i = \text{LS}(x_i)$, where $\text{LS}(x)$ is the same local search optimization procedure used in the genetic algorithm.
- (d) **Create** the neural network $N(x, \vec{p}_i)$ for particle p_i .
- (e) **Create** the associated model $\Psi_n(x, p_i)$ for particle p_i as

$$\Psi_n(x, p_i) = x(x - 1)N(x, \vec{p}_i)$$

using Equation (11).

- (f) **Calculate** the corresponding fitness value f_i of p_i using Equation (13).

- (g) If $f(p_i) \leq f(pb_i)$, then $pb_i = p_i$
4. **End For**
5. **Set** $p_{\text{best}} = \arg \min_{i \in 1 \dots N_C} f(x_i)$
6. **Set** $\text{iter} = \text{iter} + 1$.
7. **Go to Step 2** to check for termination.

3. Results

All the used algorithms as well as the provided machine learning models were coded in ANSI C++ with the assistance of the freely available Optimus optimization environment, which can be downloaded from <https://github.com/itsoulos/GlobalOptimus/> (accessed on 9 August 2024). The experiments were conducted on an AMD Ryzen 5950X with 128 GB of RAM, running Debian Linux. The values used for the simulation parameters are shown in Table 1.

Table 1. The experimental values used in the conducted experiments.

Parameter	Meaning	Value
m	Number of points used to divide the interval $[a, b]$	100
N_C	Number of chromosomes/particles	500
N_G	Maximum number of allowed generations	200
p_s	Selection rate	0.90
p_M	Mutation rate	0.05
p_l	Local search rate	0.01

Figures 1 and 2 depict the plots for the used model and for the BEC potential. The values $\gamma = 9.1865$ and $\varepsilon_n = 23$ were used in the first case and the values $\gamma = 9.8271$ and $\varepsilon_n = 54$ were used in the second case.

As it is induced from the figures, the wavefunctions vanish at the $x = 0$ and $x = 1$. Also, the genetic algorithm and the PSO variant achieved similar simulation results.

Figures 3 and 4 demonstrate the application of the proposed model to the gravitational trap potential. The first figure represents the results for both optimization methods using the values $\gamma = 45.4993$ and $\varepsilon_n = 10$ and the second stands for the simulation results for $\gamma = 193.6644$ and $\varepsilon_n = 20$. The above results have the proper, expected behavior, making the proposed numerical algorithms particularly significant for GPE.

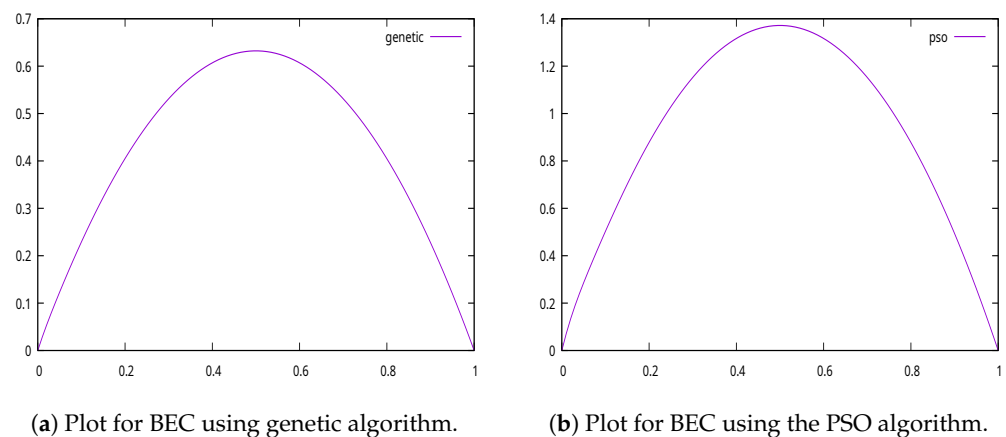
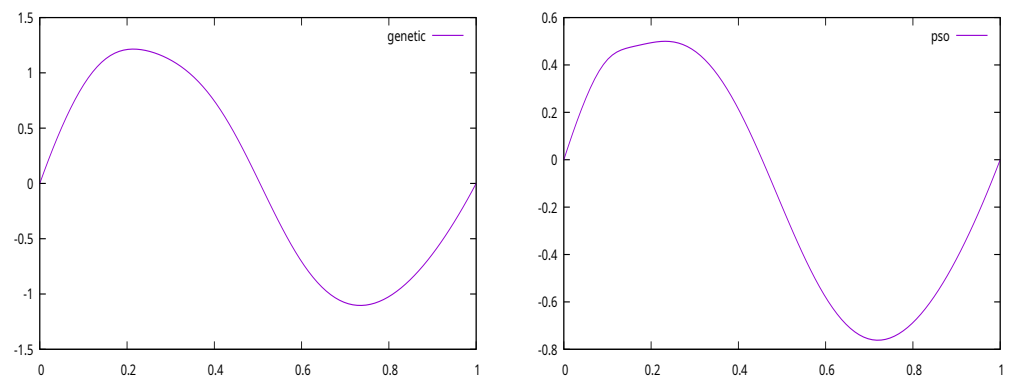


Figure 1. Plots of the suggested model for the Bose–Einstein condensates (BEC) using the two optimization methods. The values $\gamma = 9.1865$ and $\varepsilon_n = 23$ were used.

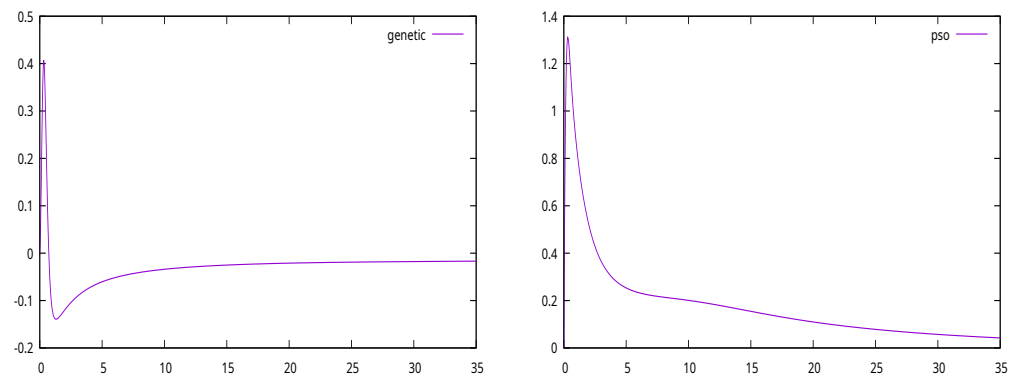


(a) Plot for BEC using genetic algorithm.

(b) Plot for BEC using the PSO algorithm.

Figure 2. Plots of the suggested model for the Bose–Einstein condensates (BEC, using the two optimization methods. The values $\gamma = 9.8271$ and $\varepsilon_n = 54$ were used.

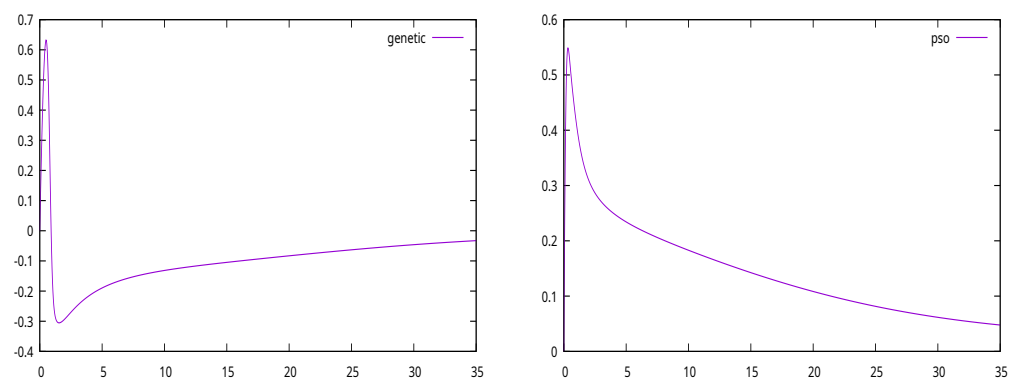
These wavefunctions obey the boundary conditions ($\Psi_n(0) = 0$ and $\Psi_n(\infty) = 0$) for both optimization algorithms (genetic algorithm and the PSO variant).



(a) Plot for gravitational trap using genetic algorithm.

(b) Plot for gravitational trap using the PSO algorithm.

Figure 3. Plots for the gravitational trap using the suggested model and the provided optimization methods. The values $\gamma = 45.4993$ and $\varepsilon_n = 10$ were used.



(a) Plot for gravitational trap using genetic algorithm.

(b) Plot for gravitational trap using the PSO algorithm.

Figure 4. Plots for the gravitational trap using the suggested model and the provided optimization methods. The values $\gamma = 193.6644$ and $\varepsilon_n = 20$ were used.

Furthermore, Figures 5 and 6 demonstrate the application of the proposed method to the harmonic trap potential using the genetic algorithm and the PSO method. For the

first case, the values $\gamma = 218.6985$ and $\varepsilon_n = 30$ were used and the values $\gamma = 203.1434$ and $\varepsilon_n = 30$ were used for the second case. Once again, the approximation of the proposed model is close to the experimental values.

The results fulfill the condition as x goes to ∞ the wavefunctions vanish. From the physical point of view, the above-illustrated results have the expected behavior and magnitudes as reported in earlier works [5]. The proposed model, based on an artificial neural network, was able to numerically solve the equation for a number of different potentials. Moreover, genetic algorithms and the modified version of particle swarm optimization show similar behavior in terms of training the proposed model.

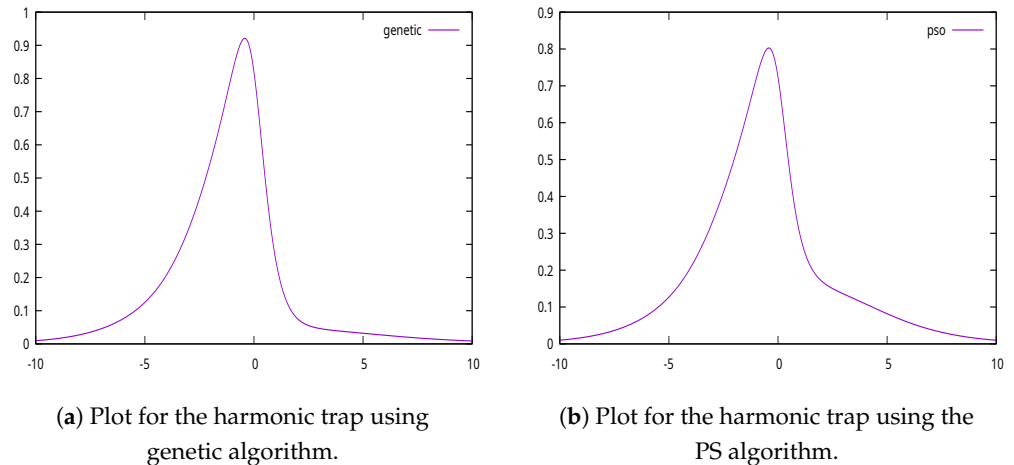


Figure 5. Plots of the proposed method for the harmonic trap potential using the suggested optimization methods. The values $\gamma = 218.6985$ and $\varepsilon_n = 30$ were used.

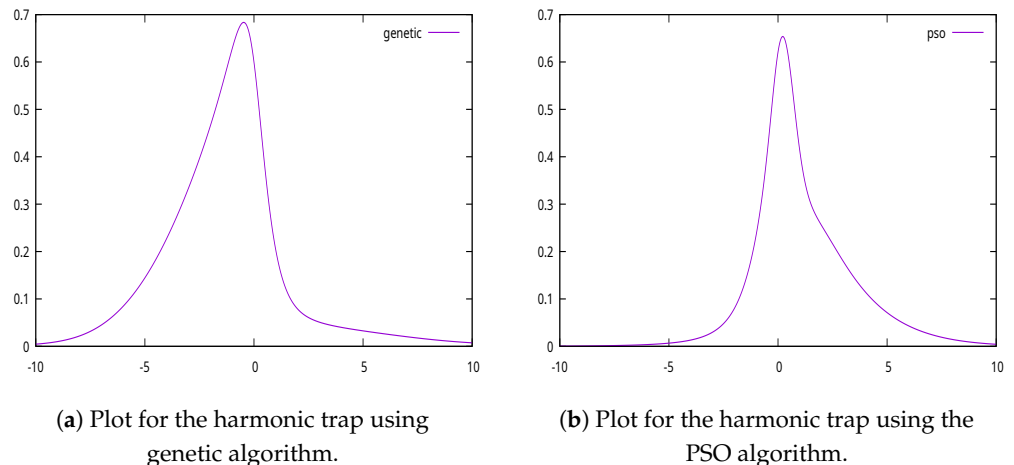


Figure 6. Plots of the proposed method for the harmonic trap potential using the suggested optimization methods. The values $\gamma = 203.1434$ and $\varepsilon_n = 30$ were used.

Moreover, an experimental comparison between various optimization methods for training the proposed model for the first potential is shown in Table 2. The column BEC1 denotes the results for the case $\gamma = 9.1865$ and $\varepsilon_n = 23$ and the column BEC2 represents the results for the values $\gamma = 9.8271$ and $\varepsilon_n = 54$.

The proposed techniques, genetic and PSO in the table, had significantly lower values than the rest without it being clear which of the two is superior in effectiveness.

Furthermore, in order to measure the effectiveness of the proposed methodology, a series of additional experiments were conducted where some parameters were altered. In Table 3, the results from the application of the modified genetic algorithm to train the proposed model are listed. In this table, the number of chromosomes varies from 100 to 600. The execution time is measured in seconds.

Table 2. Experimental results for the BEC potential for two distinct cases, using a series of optimization methods. Numbers in cells denote average value for the minimum value obtained for Equation (12).

Method	BEC1	BEC2
Adam	77.93	12.24
Bfgs	3.35	8.97
Differential Evolution	3.17	8.87
Genetic	2.22×10^{-4}	5.11×10^{-5}
PSO	5.35×10^{-5}	3.8×10^{-3}

Table 3. Experimental results for the experiment with different values for the number of chromosomes. The values $\gamma = 9.1865$ and $\varepsilon_n = 23$ were used and the BEC potential was used.

Chromosomes	Execution Time	Minimum Fitness
100	102.64	0.113
200	187.40	0.0065
400	416.65	0.0057
500	830.81	2.2×10^{-4}
600	967.19	2.1×10^{-5}

The column Execution time outlines the average execution time and the column Minimum fitness the average value for the best fitness obtained. As expected, increasing the number of chromosomes also leads to lower values of the minimum fitness, but at the same time, can significantly increase the execution time required. A satisfactory compromise between the execution time and the accuracy of the generated solution appears to be found at values of 400–600 of the genetic algorithm chromosomes.

Furthermore, for the same potential and for the genetic algorithm, the effect of changing the value of the application rate p_l of the local optimization method on the final result was studied. These results are outlined in Table 4.

Table 4. Experimental results for the experiment with different values for the local search rate parameter p_l . The values $\gamma = 9.1865$ and $\varepsilon_n = 23$ were used and the BEC potential was used.

p_l	Execution Time	Minimum Fitness
0.001	61.54	0.0095
0.002	72.38	0.0073
0.005	255.342	0.0069
0.01	830.81	2.2×10^{-4}
0.02	1091.374	2.93×10^{-6}

This parameter appears to have a more drastic effect on the effectiveness of the genetic algorithm, since even for its lower values the genetic algorithm shows significant effectiveness. Naturally, as this parameter increases, so too does the use of the local search method, and consequently, the required computing time increases significantly.

4. Conclusions

In this work, we have applied the genetic algorithm and the PSO algorithm to estimate the eigenfunctions of the 1D Gross–Pitaevskii equation, which has been used to describe the ground state of several quantum systems of identical bosons. We have used a few models to describe the external potential, e.g., the BEC in a box, the gravitational trap and the harmonic trap. The standard boundary conditions have been applied in our research

system. In our investigation, we have researched the behavior of eigenfunctions of the Gross–Pitaevskii equation on nonlinearity parameters. The numerical results show that the eigenfunctions have the proper dependence on the position x (1-dimensional problem). More specifically, for the case of the BEC in the box, the numerically found eigenfunctions completely agree with previously reported results [5]. In the case of a gravitational trap, the eigenfunction vanishes for $x = 0$ and $x = \infty$. Moreover, in the case of the harmonic trap, the wavefunction vanishes as x reaches ∞ . The two different numerical algorithms have similar predictions for every used potential.

In the future, it would be interesting to study other techniques that make use of machine learning models to solve the equation in question. For example, constructing artificial neural networks could also be applied to this problem. Another method that could be applied is radial basis networks [78], which was presented relatively recently for solving differential Equations [79].

Although the result looks very promising, the techniques used to solve the specific equation nevertheless require significant computing time for the effective training of the artificial neural networks found in the proposed models. For this reason, the use of innovative parallel processing techniques, which take advantage of modern parallel computing structures such as the MPI method [80] and the OpenMP [81] method, is deemed necessary.

Author Contributions: I.G.T. and V.N.S. were responsible for idea conception and the methodology and I.G.T. implemented the corresponding software. I.G.T. conducted the experiments, employing objective functions as test cases, and provided the comparative experiments. V.N.S. and D.T. performed the necessary statistical tests. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been financed by the European Union: Next Generation EU through the Program Greece 2.0 National Recovery and Resilience Plan, under the call RESEARCH—CREATE—INNOVATE, project name “iCREW: Intelligent small craft simulator for advanced crew training using Virtual Reality techniques” (project code:TAEDK-06195).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Ueda, M. *Fundamentals and New Frontiers of Bose–Einstein Condensation*, 1st ed.; Wspc: Singapore, 2010.
2. Rogel-Salazar, J. The Gross–Pitaevskii equation and Bose–Einstein condensates. *Eur. J. Phys.* **2013**, *34*, 247–257. [[CrossRef](#)]
3. Gardiner, C.W.; Anglin, J.R.; Fudge, T.I.A. The stochastic Gross–Pitaevskii equation. *J. Phys. B At. Mol. Opt. Phys.* **2002**, *35*, 1555–1582. [[CrossRef](#)]
4. Yukalov, V.I.; Yukalova, E.P.; Bagnato, V.S. Nonlinear coherent modes of trapped Bose–Einstein condensates. *Phys. Rev. A* **2002**, *66*, 043602. [[CrossRef](#)]
5. Marojević, Z.; Göklü, E.; Lämmerzahl, C. Energy eigenfunctions of the 1D Gross–Pitaevskii equation. *Comput. Phys. Commun.* **2013**, *184*, 1920–1930. [[CrossRef](#)]
6. Pokatov, S.P.; Ivanova, T.Y.; Ivanov, D.A. Solution of inverse problem for Gross–Pitaevskii equation with artificial neural networks. *Laser Phys. Lett.* **2023**, *20*, 095501. [[CrossRef](#)]
7. Zhong, M.; Gong, S.; Tian, S.-F.; Yan, Z. Data-driven rogue waves and parameters discovery in nearly integrable-symmetric Gross–Pitaevskii equations via PINNs deep learning. *Physica D* **2022**, *439*, 133430. [[CrossRef](#)]
8. Holland, M.J.; Jin, D.S.; Chiofalo, M.L.; Cooper, J. Emergence of Interaction Effects in Bose–Einstein Condensation. *Phys. Rev. Lett.* **1997**, *78*, 3801–3805. [[CrossRef](#)]
9. Cerimele, M.M.; Chiofalo, M.L.; Pistella, F.; Succi, S.; Tosi, M.P. Numerical solution of the Gross–Pitaevskii equation using an explicit [®]nite-difference scheme: An application to trapped Bose–Einstein condensates. *Phys. Rev. E* **2000**, *62*, 1382–1389. [[CrossRef](#)]
10. Zou, J.; Han, Y.; So, S.S. Overview of artificial neural networks. In *Artificial Neural Networks: Methods and Applications*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 14–22.
11. Wu, Y.C.; Feng, J.W. Development and application of artificial neural network. *Wirel. Pers. Commun.* **2018**, *102*, 1645–1656. [[CrossRef](#)]
12. Baldi, P.; Cranmer, K.; Faucett, T.; Sadowski, P.; Whiteson, D. Parameterized neural networks for high-energy physics. *Eur. Phys. J. C* **2016**, *76*, 235. [[CrossRef](#)]

13. Valdas, J.J.; Bonham-Carter, G. Time dependent neural network models for detecting changes of state in complex processes: Applications in earth sciences and astronomy. *Neural Netw.* **2006**, *19*, 196–207. [\[CrossRef\]](#) [\[PubMed\]](#)
14. Carleo, G.; Troyer, M. Solving the quantum many-body problem with artificial neural networks. *Science* **2017**, *355*, 602–606. [\[CrossRef\]](#) [\[PubMed\]](#)
15. Shirvany, Y.; Hayati, M.; Moradian, R. Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations. *Appl. Soft Comput.* **2009**, *9*, 20–29. [\[CrossRef\]](#)
16. Malek, A.; Beidokhti, R.S. Numerical solution for high order differential equations using a hybrid neural network—Optimization method. *Appl. Math. Comput.* **2006**, *183*, 260–271. [\[CrossRef\]](#)
17. EKaul, M.; Hill, R.L.; Walthall, C. Artificial neural networks for corn and soybean yield prediction. *Agric. Syst.* **2005**, *85*, 1–18.
18. Dahikar, S.S.; Rode, S.V. Agricultural crop yield prediction using artificial neural network approach. *Int. J. Innov. Res. Electr. Electron. Instrum. Control Eng.* **2014**, *2*, 683–686.
19. Behler, J. Neural network potential-energy surfaces in chemistry: A tool for large-scale simulations. *Phys. Chem. Chem. Phys.* **2011**, *13*, 17930–17955. [\[CrossRef\]](#)
20. Manzhos, S.; Dawes, R.; Carrington, T. Neural network-based approaches for building high dimensional and quantum dynamics-friendly potential energy surfaces. *Int. J. Quantum Chem.* **2015**, *115*, 1012–1020. [\[CrossRef\]](#)
21. Enke, D.; Thawornwong, S. The use of data mining and neural networks for forecasting stock market returns. *Expert Syst. Appl.* **2005**, *29*, 927–940. [\[CrossRef\]](#)
22. Falat, L.; Pancikova, L. Quantitative Modelling in Economics with Advanced Artificial Neural Networks. *Procedia Econ. Financ.* **2015**, *34*, 194–201. [\[CrossRef\]](#)
23. Angelini, E.; Di Tollo, G.; Roli, A. A neural network approach for credit risk evaluation. *Q. Rev. Econ. Financ.* **2008**, *48*, 733–755. [\[CrossRef\]](#)
24. Moghaddam, A.H.; Moghaddam, M.H.; Esfandiyari, M. Stock market index prediction using artificial neural network. *J. Econ. Financ. Adm. Sci.* **2016**, *21*, 89–93. [\[CrossRef\]](#)
25. Amato, F.; López, A.; Peña-Méndez, E.M.; Vañhara, P.; Hampl, A.; Havel, J. Artificial neural networks in medical diagnosis. *J. Appl. Biomed.* **2013**, *11*, 47–58. [\[CrossRef\]](#)
26. Sidey-Gibbons, J.A.; Sidey-Gibbons, C.J. Machine learning in medicine: A practical introduction. *Bmc Med. Res. Methodol.* **2019**, *19*, 64. [\[CrossRef\]](#)
27. Tsoulos, I.; Gavrilis, D.; Glavas, E. Neural network construction and training using grammatical evolution. *Neurocomputing* **2008**, *72*, 269–277. [\[CrossRef\]](#)
28. Lagaris, I.E.; Likas, A.; Fotiadis, D.I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* **1998**, *9*, 987–1000. [\[CrossRef\]](#) [\[PubMed\]](#)
29. Aarts, L.P.; van der Veer, P. Neural Network Method for Solving Partial Differential Equations. *Neural Process. Lett.* **2001**, *14*, 261–271. [\[CrossRef\]](#)
30. Parisi, D.R.; Mariani, M.C.; Laborde, M.A. Solving differential equations with unsupervised neural networks. *Chem. Eng. Process. Process Intensif.* **2003**, *42*, 715–721. [\[CrossRef\]](#)
31. Tsoulos, I.G.; Gavrilis, D.; Glavas, E. Solving differential equations with constructed neural networks. *Neurocomputing* **2009**, *72*, 2385–2391. [\[CrossRef\]](#)
32. Ryan, C.; Collins, J.J.; Neill, M.O. Grammatical evolution: Evolving programs for an arbitrary language. In Proceedings of the Genetic Programming: First European Workshop, EuroGP’98, Paris, France, 14–15 April 1998; Proceedings 1; Springer: Berlin/Heidelberg, Germany, 1998; pp. 83–96.
33. Kumar, M.; Yadav, N. Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: A survey. *Comput. Math. Appl.* **2011**, *62*, 3796–3811. [\[CrossRef\]](#)
34. Zhang, T.; Li, Y. Global exponential stability of discrete-time almost automorphic Caputo–Fabrizio BAM fuzzy neural networks via exponential Euler technique. *Knowl.-Based Syst.* **2022**, *246*, 108675. [\[CrossRef\]](#)
35. Lagaris, I.E.; Likas, A.; Fotiadis, D.I. Artificial neural network methods in quantum mechanics. *Comput. Phys. Commun.* **1997**, *104*, 1–14. [\[CrossRef\]](#)
36. Kolanoski, H. Application of Artificial Neural Networks in Particle Physics. In *Artificial Neural Networks—ICANN 96*; Von der Malsburg, C., von Seelen, W., Vorbrüggen, J.C., Sendhoff, B., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1996; Volume 1112.
37. Cai, Z.; Liu, J. Approximating quantum many-body wave functions using artificial neural networks. *Phys. Rev. B* **2018**, *97*, 035116. [\[CrossRef\]](#)
38. Cai, S.; Wang, Z.; Wang, S.; Perdikaris, P.; Karniadakis, G.E. Physics-Informed Neural Networks for Heat Transfer Problems. *J. Heat Transf.* **2021**, *143*, 060801. [\[CrossRef\]](#)
39. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [\[CrossRef\]](#)
40. Vora, K.; Yagnik, S. A survey on backpropagation algorithms for feedforward neural networks. *Int. J. Eng. Dev. Res.* **2014**, *1*, 193–197.
41. Riedmiller, M.; Braun, H. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP algorithm. In Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, USA, 28 March–1 April 1993; pp. 586–591.

42. Hermanto, R.P.S.; Nugroho, A. Waiting-time estimation in bank customer queues using RPROP neural networks. *Procedia Comput. Sci.* **2018**, *135*, 35–42. [\[CrossRef\]](#)
43. Kingma, D.P.; Ba, J.L. ADAM: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, 7–9 May 2015; pp. 1–15.
44. Törn, A.; Ali, M.M.; Viitanen, S. Stochastic global optimization: Problem classes and solution techniques. *J. Glob. Optim.* **1999**, *14*, 437–447. [\[CrossRef\]](#)
45. Floudas, C.A.; Pardalos, P.M. (Eds.) *State of the Art in Global Optimization: Computational Methods and Applications*; Springer: New York, NY, USA, 2013.
46. Liu, Z.; Liu, A.; Wang, C.; Niu, Z. Evolving neural network using real coded genetic algorithm (GA) for multispectral image classification. *Future Gener. Comput. Syst.* **2004**, *20*, 1119–1129. [\[CrossRef\]](#)
47. Carvalho, M.; Ludermit, T.B. Particle swarm optimization of neural network architectures and weights. In Proceedings of the 7th International Conference on Hybrid Intelligent Systems (HIS 2007), Kaiserslautern, Germany, 17–19 September 2007; pp. 336–339.
48. Kiranyaz, S.; Ince, T.; Yildirim, A.; Gabbouj, M. Evolutionary artificial neural networks by multi-dimensional particle swarm optimization. *Neural Netw.* **2009**, *22*, 1448–1462. [\[CrossRef\]](#)
49. Ilonen, J.; Kamarainen, J.K.; Lampinen, J. Differential evolution training algorithm for feed-forward neural networks. *Neural Process. Lett.* **2003**, *17*, 93–105. [\[CrossRef\]](#)
50. Slowik, A.; Bialko, M. Training of artificial neural networks using differential evolution algorithm. In Proceedings of the 2008 Conference on Human System Interactions, Krakow, Poland, 25–27 May 2008; pp. 60–65.
51. Salama, K.M.; Abdelbar, A.M. Learning neural network structures with ant colony algorithms. *Swarm Intell.* **2015**, *9*, 229–265. [\[CrossRef\]](#)
52. Mirjalili, S. How effective is the Grey Wolf optimizer in training multi-layer perceptrons. *Appl. Intell.* **2015**, *43*, 150–161. [\[CrossRef\]](#)
53. Aljarah, I.; Faris, H.; Mirjalili, S. Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Comput.* **2018**, *22*, 1–15. [\[CrossRef\]](#)
54. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [\[CrossRef\]](#)
55. Stender, J. *Parallel Genetic Algorithms: Theory & Applications*; IOS Press: Amsterdam, The Netherlands, 1993.
56. Haupt, R.L.; Werner, D.H. *Genetic Algorithms in Electromagnetics*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
57. Grady, S.A.; Hussaini, M.Y.; Abdullah, M.M. Placement of wind turbines using genetic algorithms. *Renew. Energy* **2005**, *30*, 259–270. [\[CrossRef\]](#)
58. Oh, I.S.; Lee, J.S.; Moon, B.R. Hybrid genetic algorithms for feature selection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2004**, *26*, 1424–1437.
59. Bakirtzis, A.G.; Biskas, P.N.; Zoumas, C.E.; Petridis, V. Optimal power flow by enhanced genetic algorithm. *IEEE Trans. Power Syst.* **2002**, *17*, 229–236. [\[CrossRef\]](#)
60. Zegordi, S.H.; Nia, M.B. A multi-population genetic algorithm for transportation scheduling. *Transp. Res. Part E Logist. Transp. Rev.* **2009**, *45*, 946–959. [\[CrossRef\]](#)
61. Leung, F.H.F.; Lam, H.K.; Ling, S.H.; Tam, P.K.S. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Trans. Neural Netw.* **2003**, *14*, 79–88. [\[CrossRef\]](#) [\[PubMed\]](#)
62. Sedki, A.; Ouazar, D.; El Mazoudi, E. Evolving neural network using real coded genetic algorithm for daily rainfall—Runoff forecasting. *Expert Syst. Appl.* **2009**, *36*, 4523–4527. [\[CrossRef\]](#)
63. Majdi, A.; Beiki, M. Evolving neural network using a genetic algorithm for predicting the deformation modulus of rock masses. *Int. J. Rock Mech. Min. Sci.* **2010**, *47*, 246–253. [\[CrossRef\]](#)
64. Kaelo, P.; Ali, M.M. Integrated crossover rules in real coded genetic algorithms. *Eur. J. Oper. Res.* **2007**, *176*, 60–76. [\[CrossRef\]](#)
65. Tsoulos, I.G. Modifications of real code genetic algorithm for global optimization. *Appl. Math. Comput.* **2008**, *203*, 598–607. [\[CrossRef\]](#)
66. Powell, M.J.D. A Tolerant Algorithm for Linearly Constrained Optimization Calculations. *Math. Program.* **1989**, *45*, 547–566. [\[CrossRef\]](#)
67. Poli, R.; Kennedy, J.; Blackwell, T. Particle swarm optimization An Overview. *Swarm Intell.* **2007**, *1*, 33–57. [\[CrossRef\]](#)
68. Robinson, J.; Rahmat-Samii, Y. Particle swarm optimization in electromagnetics. *IEEE Trans. Antennas Propag.* **2004**, *52*, 397–407. [\[CrossRef\]](#)
69. Pace, F.; Santilano, A.; Godio, A. A review of geophysical modeling based on particle swarm optimization. *Surv. Geophys.* **2021**, *42*, 505–549. [\[CrossRef\]](#) [\[PubMed\]](#)
70. Call, S.T.; Zubarev, D.Y.; Boldyrev, A.I. Global minimum structure searches via particle swarm optimization. *J. Comput. Chem.* **2007**, *28*, 1177–1186. [\[CrossRef\]](#)
71. Halter, W.; Mostaghim, S. Bilevel optimization of multi-component chemical systems using particle swarm optimization. In Proceedings of the 2006 IEEE International Conference on Evolutionary Computation, Vancouver, BC, Canada, 16–21 July 2006; pp. 1240–1247.
72. Chakraborty, S.; Samanta, S.; Biswas, D.; Dey, N.; Chaudhuri, S.S. Particle swarm optimization based parameter optimization technique in medical information hiding. In Proceedings of the 2013 IEEE International Conference on Computational Intelligence and Computing Research, Enathi, India, 26–28 December 2013; pp. 1–6.

73. Harb, H.M.; Desuky, A.S. Feature selection on classification of medical datasets based on particle swarm optimization. *Int. J. Comput. Appl.* **2014**, *104*, 5.
74. Maschek, M.K. Particle Swarm Optimization in Agent-Based Economic Simulations of the Cournot Market Model. *Intelligent Systems in Accounting. Financ. Manag.* **2015**, *22*, 133–152.
75. Yu, J.; Xi, L.; Wang, S. An improved particle swarm optimization for evolving feedforward artificial neural networks. *Neural Process. Lett.* **2007**, *26*, 217–231. [[CrossRef](#)]
76. Charilogis, V.; Tsoulos, I.G. Toward an Ideal Particle Swarm Optimizer for Multidimensional Functions. *Information* **2022**, *13*, 217. [[CrossRef](#)]
77. Eberhart, R.C.; Shi, Y.H. Tracking and optimizing dynamic systems with particle swarms. In Proceedings of the Congress on Evolutionary Computation, Seoul, Republic of Korea, 27–30 May 2001.
78. Park, J.; Sandberg, I.W. Universal Approximation Using Radial-Basis-Function Networks. *Neural Comput.* **1991**, *3*, 246–257. [[CrossRef](#)]
79. Zhang, Y. An accurate and stable RBF method for solving partial differential equations. *Appl. Math. Lett.* **2019**, *97*, 93–98. [[CrossRef](#)]
80. Gabriel, E.; Fagg, G.E.; Bosilca, G.; Angskun, T.; Dongarra, J.J.; Squyres, J.M.; Sahay, V.; Kambadur, P.; Barrett, B.; Lumsdaine, A.; et al. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In Proceedings of the Recent Advances in Parallel Virtual Machine and Message Passing Interface: 11th European PVM/MPI Users' Group Meeting, Budapest, Hungary, 19–22 September 2004; Proceedings 11; Springer: Berlin/Heidelberg, Germany, 2004; pp. 97–104.
81. Ayguadé, E.; Copt, N.; Duran, A.; Hoeflinger, J.; Lin, Y.; Massaioli, F.; Teruel, F.; Unnikrishnan, P.; Zhang, G. The design of OpenMP tasks. *IEEE Trans. Parallel Distrib. Syst.* **2008**, *20*, 404–418. [[CrossRef](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.