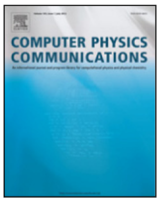
Computer Physics Communications 214 (2017) 31–38

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc



Neural network approach for the calculation of potential coefficients in quantum mechanics

Sebastián Ossandón[a,*], Camilo Reyes[b], Patricio Cumsille[c,d], Carlos M. Reyes[c]

[a] *Instituto de Matemáticas, Pontificia Universidad Católica de Valparaíso, Blanco Viel 596, Cerro Barón, Valparaíso, Chile*
[b] *Departamento de Ciencias Fisicas, Universidad Andres Bello, Avenida Republica 220, Santiago, Chile*
[c] *Departamento de Ciencias Básicas, Universidad del Bío Bío, Casilla 447, Chillán, Chile*
[d] *Centro de Biotecnología y Bioingeniería (CeBiB), Universidad de Chile, Beaucheff 851, Santiago, Chile*

A R T I C L E   I N F O

Article history:
Received 15 December 2015
Received in revised form 5 January 2017
Accepted 9 January 2017
Available online 20 January 2017

Keywords:
Artificial neural network
Radial basis function
Coefficients of the potential function
Inverse problems
Eigenvalues of the Schrödinger operator
Finite element method

A B S T R A C T

A numerical method based on artificial neural networks is used to solve the inverse Schrödinger equation for a multi-parameter class of potentials. First, the finite element method was used to solve repeatedly the direct problem for different parametrizations of the chosen potential function. Then, using the attainable eigenvalues as a training set of the direct radial basis neural network a map of new eigenvalues was obtained. This relationship was later inverted and refined by training an inverse radial basis neural network, allowing the calculation of the unknown parameters and therefore estimating the potential function. Three numerical examples are presented in order to prove the effectiveness of the method. The results show that the method proposed has the advantage to use less computational resources without a significant accuracy loss.

1. Introduction

As known the Schrödinger equation is a partial differential equation developed during the first quarter of the 20th century and its a fundamental part of the quantum mechanics theory. Roughly speaking this equation describes the behavior of a system under the influence of different potentials, and one of the main goals is to find the dynamics of the system determined uniquely by the eigenvalues and eigenvectors. In the following we will review briefly the usage of numerical methods to solve the Schrödinger equation and in particular the use of neural networks.

In general, only in limited cases exact analytic solutions can be obtained, for example the free particle, linear harmonic oscillator or the hydrogen atom. On the other hand, there exist a number of useful approximation methods for more general hamiltonians, such as perturbation methods or the Wentzel–Kramers–Brillouin (WKB) method. Another well established methods for solving numerically the Schrödinger equation, are the ones based in variations of the Numerov's method, as shown in (Pillai et al. [1]) where the wave function is discretized over a lattice. However, to our best knowledge, when applied to practical physical problems, these methods have turn out to be less successful.

In (Braun et al.) [2], the Lanczos method is applied, on a grid, for obtaining eigensolutions of quantum systems. This methodology is used to solve one-, two-, and three-dimensional quantum problems. In (Ishikawa [3]) a numerical method is proposed in order to solve accurately the eigenvalue problem in quantum mechanics. In this case the efficiency is proved through the applications to the harmonic oscillator, in which they achieved 15-digit accuracy with double precision operations. In (Kannan and Masud [4]) two methods are presented in order to stabilize the Schrödinger wave equations, the first one consisting in a Garlekin/least-squares method, whose consistency and convergence was analyzed through potentials which have known analytic solutions. In (Watanabe and Tsukada [5]) the wave function evolution in a magnetic field is analyzed using a numerical method based on the finite elements, improving the accuracy without increasing the computational cost. Among other schemes proposed for solving numerically the Schrödinger equation it can be mentioned the study done by (Simos and Williams [6]) where they use a method based on phase-lag minimization in order to compute the eigenvalues, the method was tested in two types of potentials, an even function with respect of a one dimensional domain, and a general case of the Morse potential.

Other applications of ANN in partial differential equations can be found in (Ossandón and Reyes [7]) and (Ossandón et al. [8]), where the inverse eigenvalue problems for the linear elasticity operator and for the anisotropic Laplace operator are solved respectively. In (Poggio et al. [9]) it is shown that the ill-posed

* Corresponding author.
E-mail addresses: sebastian.ossandon@pucv.cl (S. Ossandón), creyesm70@gmail.com (C. Reyes), pcumsille@ubiobio.cl (P. Cumsille), creyes@ubiobio.cl (C.M. Reyes).

http://dx.doi.org/10.1016/j.cpc.2017.01.006
0010-4655/© 2017 Elsevier B.V. All rights reserved.

problem of function approximation through sparse data can be regularized using an appropriate class of approximation functions.

However, the ANNs have been less explored in quantum mechanics. Some works in this direction are given in (Lagaris et al. [10]), where a feedforward Artificial Neural Network (ANN) is used to find the eigenvalues of integro-differential operators, using the analytic solutions to test the accuracy of the solutions. In this case the neural network proved to be highly accurate, robust and efficient. Another use for the feedforward neural network can be seen in (Shirvany et al. [11]), where an energy term is derived from the boundary conditions which allows to use an unsupervised neural network to solve the equations, also the results given by the neural network were compared with the analytic solutions.

In the present work, our goal is to study the recovering of coefficients associated to the potential function through a finite set of eigenvalues in the quantum system. We assume that the function between the eigenvalues and the potential coefficients is smooth in the sense that two similar inputs correspond to two similar outputs. We emphasize that the error of any designed ANNs cannot have a better performance than the technique used to create the training data. As a consequence, the performance of any ANN is directly related to the training data. In general, all the computation process used by the neural networks, including the training, validation and simulation process, has lower computational time than the finite element method.

Hence, the main issue of this proposed study is to solve the inverse problem associated to the Schrödinger equation, that is to say, calculate a set of coefficients associated to a potential function, through eigenvalues of the Schrödinger operator using ANNs. The ANN proposed is a multilayered Radial-Basis Function (RBF) network (see Ossandón and Reyes [7] and Ossandón et al. [8]). As discussed in (Schilling et al. [12]), a RBF ANN can approximate a function f using nonlinear functions which provides the best fit to the training data. An evaluation of the performance (computational time and accuracy) of the ANN methodology proposed will be done, comparing the results to a classical numerical method based on FEM.

The article is organized as follows. In Section 2, we give an introduction to the eigenvalue problem in quantum mechanics. In Section 3, we present the solution to the direct and inverse problem associated to the calculation of the eigenvalues of the time-independent Schrödinger equation. Numerical results and discussion are given in Section 4. Finally, in Section 5, the conclusions of this work are presented.

2. Eigenvalue problem in quantum mechanics

Let $\Omega \subset \mathbb{R}^k$ ($k \geqslant 1$) be a nonempty, open, connected and bounded domain, with a Lipschitz-continuous boundary $\Gamma := \partial\Omega$. The unit normal vector pointing to the exterior of Ω is denoted by $\boldsymbol{n} = (n_1, n_2, \ldots, n_k)^T \in \mathbb{R}^k$ and $\boldsymbol{x} = (x_1, x_2, \ldots, x_k)^T \in \mathbb{R}^k$.

Let $\widehat{\Lambda}$ be an observable associated to a physical quantity Λ. Let us say that ψ_γ is an eigenfunction of this operator, and λ_γ its associated eigenvalue, if $\psi_\gamma \neq 0$ and

$$\begin{cases} \widehat{\Lambda} \cdot \psi_\gamma = \lambda_\gamma \psi_\gamma & \text{in} \quad \Omega, \\ \psi_\gamma = 0 & \text{on} \quad \Gamma. \end{cases} \tag{1}$$

It is worth noting that the eigenvalues of a hermitian operator lie in the real line. Indeed, if we multiply Eq. (1) by ψ_γ^* and integrate in Ω, we obtain

$$\lambda_\gamma = \frac{\int \psi_\gamma^*(\boldsymbol{x})[\widehat{\Lambda}\psi_\gamma(\boldsymbol{x})]d\boldsymbol{x}}{\int |\psi_\gamma(\boldsymbol{x})|^2 d\boldsymbol{x}} \tag{2}$$

which is real.

In the case that the observable $\widehat{\Lambda}$ is the hamiltonian operator $\widehat{H} = -\frac{\hbar^2}{2m}\Delta\psi(\boldsymbol{x}) + V(\boldsymbol{x}, \boldsymbol{\theta})$, we have the well known time-independent Schrödinger equation $\widehat{H}\psi_\gamma(\boldsymbol{x}) = E_\gamma\psi_\gamma(\boldsymbol{x})$. In this equation $\hbar$ is the Planck's constant, m is the mass associated to the quantum system, and $V(\boldsymbol{x}, \boldsymbol{\theta})$ is the potential function, coming from the potential energy and, in our case, depending on a set of coefficients grouped in the vector $\boldsymbol{\theta} \in \mathbb{R}^l$, $l \geqslant 1$. Thus the energy E_γ is an eigenvalue, and $\psi_\gamma(\boldsymbol{x})$ its related eigenfunction, associated to the hamiltonian operator $\widehat{H}$.

3. The direct and inverse problems

3.1. The direct problem

From now on let us consider only two and three dimensional bounded domains, i.e. $k = 2$ or 3. In addition let us suppose that $V(\cdot, \boldsymbol{\theta}) \in \mathrm{L}^\infty(\Omega)$, $V(\boldsymbol{x}, \boldsymbol{\theta}) \geqslant V_0 > 0$ for a.e. $x \in \Omega$ and $\forall\theta \in \mathbb{R}^l$. Our main goal is to solve the following eigenvalue problem:

Find $E \in \mathbb{R}$ and functions $\psi(\boldsymbol{x}) \not\equiv 0$ which are solution of

$$\begin{cases} -\frac{\hbar^2}{2m}\Delta\psi + V(\boldsymbol{x}, \boldsymbol{\theta})\psi = E\,\psi & \text{in} \quad \Omega, \\ \psi = 0 & \text{on} \quad \Gamma. \end{cases} \tag{3}$$

As known (see [13]) the only non-null solutions of Eqs. (3) are a pair sequence $\{(E_j, \psi_j)\}_{j\geqslant 1}$ of eigenvalues and eigenfunctions.

We define the following function $\mathcal{S}_{\overline{\Omega},N}$ associated to Eq. (3):

$$\mathcal{S}_{\overline{\Omega},N} : \mathbb{R}^l \to \mathbb{R}^N, \qquad \overrightarrow{\boldsymbol{E}} := (E_1, E_2, \ldots, E_N)^{\mathbf{T}} = \mathcal{S}_{\overline{\Omega},N}(\boldsymbol{\theta}). \tag{4}$$

Given the values of the coefficients $\boldsymbol{\theta} \in \mathbb{R}^l$, the potential $V(\boldsymbol{x}, \boldsymbol{\theta})$ is completely well determined and consequently $\mathcal{S}_{\overline{\Omega},N}$ ($N \in \mathbb{N}$), for each domain Ω with regular boundary Γ, solves the direct problem associated to boundary-value problem (3), calculating the first N eigenvalues of the Schrödinger operator.

Let us define the functional space

$$\mathcal{V} = \mathrm{H}_0^1(\Omega) = \left\{ v \in \mathrm{H}^1(\Omega); \quad v = 0 \quad \text{on} \quad \Gamma \right\}, \tag{5}$$

equipped with the usual norm $\|v\|_{1,\Omega}^2 = \int_\Omega |\nabla v|^2 dx + \int_\Omega |v|^2 dx$.

Thus the eigenvalue problem for Schrödinger equation with homogeneous boundary conditions can be formulated as (weak formulation):

Find $(E, \psi) \in (\mathbb{R}, \mathcal{V})$ *such that*

$$a_{\boldsymbol{\theta}}(u, v) = E\,(u, v)_{0,\Omega} \qquad \forall v \in \mathcal{V} \tag{6}$$

where

$$\begin{aligned} a_{\boldsymbol{\theta}}(u, v) &:= \frac{\hbar^2}{2m}\int_\Omega \nabla u \cdot \nabla v dx + \int_\Omega V(\boldsymbol{x}, \boldsymbol{\theta}) uv dx \quad \text{and} \\ (u, v)_{0,\Omega} &= \int_\Omega uv dx. \end{aligned} \tag{7}$$

It is worth noting that the wellposedness of the discrete form of (6) can be guaranteed by the fact that the corresponding approximation space satisfies the Babuska–Brezzi condition (see [13–17] and [18]). Let $\{\mathcal{T}_h\}_{h>0}$ be a regular family of triangulations of Ω, made up of triangles T of diameter h_T, such that $h := \sup\{h_T \mid T \in \mathcal{T}_h\}$ and $\overline{\Omega} = \bigcup\{T : T \in \mathcal{T}_h\}$. In association with the mesh $\mathcal{T}_h$, let us select the finite element space $\mathcal{V}_h \subset \mathcal{V}$ of the continuous functions in Ω which are piecewise polynomials $\mathbb{P}_j$ of degree j, with $j \geq 1$, in each triangle $T \in \mathcal{T}_h$.

Let $(E_h, u_h) \in (\mathbb{R}, \mathcal{V}_h)$ be the eigenpair solution to the discrete form of (6). It is well known that the Rayleigh quotient for each eigenvalue E_h is given by:

$$E_h = \frac{a_{\boldsymbol{\theta}}(u_h, u_h)}{(u_h, u_h)_{0,\Omega}}. \tag{8}$$

3.2. The inverse problem

The inverse problem associated with (3) is the following:
Find $\boldsymbol{\theta} \in \mathbb{R}^l$ such that the following boundary-value problem

$$\begin{cases} -\dfrac{\hbar^2}{2m}\Delta\psi_n + V(\boldsymbol{x},\boldsymbol{\theta})\psi_n = E_n\,\psi_n & \text{in} \quad \Omega, \\ \psi_n = 0 & \text{on} \quad \Gamma \end{cases} \tag{9}$$

is solved for the given sequence $\left\{E_n, \psi_n\right\}_{n=1}^{N}$, with $N < +\infty$. In simple words, our inverse problem consists in finding the potential coefficients $\theta \in \mathbb{R}^l$ provided that we known the first N $\left\{E_n, \psi_n\right\}_{n=1}^{N}$ eigenvalues and eigenfunctions of problem (3).

Thus, it is now possible to define the function $\mathcal{S}_{\overline{\Omega},N}^{-1}$, which is the inverse function of $\mathcal{S}_{\overline{\Omega},N}$, in order to solve the inverse problem (9):

$$\mathcal{S}_{\overline{\Omega},N}^{-1} : \mathbb{R}^N \to \mathbb{R}^l, \quad \boldsymbol{\theta} = \mathcal{S}_{\overline{\Omega},N}^{-1}(\overrightarrow{\boldsymbol{E}}). \tag{10}$$

In order to numerically solve the above inverse problem, first let us consider a direct feed-forward RBF ANN. A typical flow diagram for a feed-forward neural network is shown in Fig. 1. Let $\widehat{\mathcal{S}}_{\overline{\Omega},N}^{\boldsymbol{\omega}_1} : \mathbb{R}^l \to \mathbb{R}^N$ be an approximation of the function $\mathcal{S}_{\overline{\Omega},N}$ (see [12] and [7]), with one hidden layer containing s_1 neurons and one output layer containing N neurons. Let us notice that the activation function associated to each neuron is characterized by $y = \exp\{-x^2\}$.

The function $\widehat{\mathcal{S}}_{\overline{\Omega},N}^{\boldsymbol{\omega}_1}$ has the following form:

$$\widehat{\overrightarrow{\boldsymbol{E}}} = \widehat{\mathcal{S}}_{\overline{\Omega},N}^{\boldsymbol{\omega}_1}(\boldsymbol{\theta}) = \mathcal{L}_W^1 \cdot \exp(-\boldsymbol{y}_1(\boldsymbol{\theta}) \cdot * \boldsymbol{y}_1(\boldsymbol{\theta})) + \boldsymbol{b}_2^1, \tag{11}$$

where $\widehat{\overrightarrow{\boldsymbol{E}}} := (\widehat{E}_1, \widehat{E}_2, \dots, \widehat{E}_N)^{\mathbf{T}}$ is the output vector and $\boldsymbol{y}_1(\boldsymbol{\theta}) = (\mathcal{I}_W^1 \cdot (\boldsymbol{\theta})^T) \cdot * \boldsymbol{b}_1^1$. Furthermore $\boldsymbol{\omega}_1$ is a vector containing all the weights associated to the neural network which must be determined in the training of the network. In other words $\boldsymbol{\omega}_1$ contains all coefficients associated with the design parameters $\mathcal{L}_W^1$ $(N \times s_1)$, $\mathcal{I}_W^1$ $(s_1 \times l)$, $\boldsymbol{b}_1^1$ $(s_1 \times 1)$ and $\boldsymbol{b}_2^1$ $(N \times 1)$. It is important to remark that "$\cdot$" is the classic matrix–vector product, whereas "$\cdot *$" is the element-by-element vector–vector product.

In order to get the trained ANN with $N_t^{(1)}$ input–output vectors $\left\{(\boldsymbol{\theta})^{(i)}, (\overrightarrow{\boldsymbol{E}})^{(i)}\right\}_{i=1}^{N_t^{(1)}}$, where $(\overrightarrow{\boldsymbol{E}})^{(i)} = \mathcal{S}_{\overline{\Omega},N}(\boldsymbol{\theta}^{(i)})$, let us define the following optimization problem:

$$\begin{aligned} \widehat{\boldsymbol{\omega}_1} &= \inf_{\boldsymbol{\omega}_1} J_{N_t^{(1)}}(\boldsymbol{\omega}_1) \\ &= \inf_{\boldsymbol{\omega}_1}\left\{\frac{1}{N_t^{(1)}}\sum_{i=1}^{N_t^{(1)}}\left((\overrightarrow{\boldsymbol{E}})^{(i)} - \widehat{\mathcal{S}}_{\overline{\Omega},N}^{\boldsymbol{\omega}_1}(\boldsymbol{\theta}^{(i)})\right)^2\right\}. \end{aligned} \tag{12}$$

The problem (12) can be iteratively solved using the backpropagation algorithm.

Once determined the optimal value for $\boldsymbol{\omega}_1$, i.e. $\widehat{\boldsymbol{\omega}_1}$, it is possible to consider an inverse RBF ANN $\widehat{\mathcal{S}}_{\overline{\Omega},N}^{\boldsymbol{\omega}_2} : \mathbb{R}^N \to \mathbb{R}^l$, trained with simulated data obtained from the direct network, to calculate the inverse of Eq. (11) in order to obtain an approximation for $\mathcal{S}_{\overline{\Omega},N}^{-1}$ as follows:

$$\begin{cases} \widehat{\boldsymbol{\theta}} = \widehat{\mathcal{S}}_{\overline{\Omega},N}^{\boldsymbol{\omega}_2}(\widehat{\overrightarrow{\boldsymbol{E}}}) = \mathcal{L}_W^2 \cdot \exp(-\boldsymbol{y}_2(\widehat{\overrightarrow{\boldsymbol{E}}}) \cdot * \boldsymbol{y}_2(\widehat{\overrightarrow{\boldsymbol{E}}})) + \boldsymbol{b}_2^2, \\ \boldsymbol{y}_2(\widehat{\overrightarrow{\boldsymbol{E}}}) = (\mathcal{I}_W^2 \cdot \widehat{\overrightarrow{\boldsymbol{E}}}) \cdot * \boldsymbol{b}_1^2, \end{cases} \tag{13}$$

where $\boldsymbol{\omega}_2$ is a parameter vector containing everything that is going to be determined from the network training and associated with the design parameters $\mathcal{L}_W^2$ $(l \times s_2)$, $\mathcal{I}_W^2$ $(s_2 \times N)$, $\boldsymbol{b}_1^2$ $(s_2 \times 1)$ and $\boldsymbol{b}_2^2$ $(l \times 1)$. Let us notice that s_2 is the number of neurons in the hidden layer.

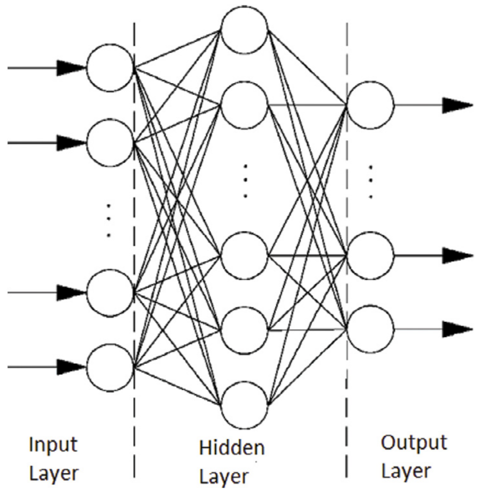


Fig. 1. Feed-forward neural network.

The optimization problem arising in the training of this inverse network can be also solved using the backpropagation algorithm or the Levenberg–Marquardt algorithm with $N_t^{(2)}$ input–output vectors.

As aforementioned, the purpose of this work is to solve the inverse problem described above. However, in order to solve the inverse problem it is necessary to solve many direct problems. Since the FEM is computationally expensive, it is essentially used to solve only a few direct problems, allowing us to have sufficient data ($N_t^{(1)}$) in order to train the direct RBF ANN. Once trained the direct network, it is used to solve many more direct problems ($N_t^{(2)}$, with $N_t^{(2)} \gg N_t^{(1)}$), in a reasonable computational time. This is mainly due to the optimized calculation speed of the designed direct RBF ANN. In this way, we can increase the database if needed, in order to train the inverse RBF ANN, consuming a reasonable calculation time and using reasonable computational resources.

Finally, it is worth noting that the direct problem analyzed in this work is well posed. This means that solutions coming from a potential function given by $V(x) = ax^2$, for $(a > 0)$, will have only positive eigenvalues, and therefore the training set for the direct RBF ANN will only consider positive values for a and positive eigenvalues. This in turn implies that the training set for the inverse RBF ANN will also consist only of positive eigenvalues and positive values of a, because we use the inverse functional relationship in order to train the inverse network (the neural network cannot extrapolate to negative eigenvalues). Thus the solutions of the inverse problems obtained with the inverse RBF ANN will never can consider negative values of a considering, as input data, positive eigenvalues.

4. Numerical results and discussion

In this section, numerical examples are presented in order to show the effectiveness and relevance of the proposed numerical method.

First, let us give a brief description of the numerical procedure used in our examples:

(I) : Generation of the training data set for the direct RBF ANN: $\left\{\boldsymbol{\theta}^{(i)}\right\}_{i=1}^{N_t^1}$ coefficients (N_t^1 sparse initial data).

(II) : Select the value of N (number of eigenvalues to be used).

(III) : For each $1 \leqslant i \leqslant N_t^1$, calculate the N first eigenvalues (energy values): $0 < E_1^{(i)} < E_2^{(i)} \leqslant \cdots \leqslant E_N^{(i)}$ using FEM.

(IV) : Design and train a direct RBF ANN, resulting in the eigenvalues mapping, obtained in **(III)**, as a function of $\boldsymbol{\theta}^{(i)}$, $1 \leqslant i \leqslant N_t^1$.

(V) : New Data Generation: Generate a refined sample $\left\{\boldsymbol{\theta}^{(i)}\right\}_{i=1}^{N_t^2}$ of size $N_t^2 \gg N_t^1$. The generated coefficients will be used as a training data set for the inverse RBF ANN.

(VI) : Calculate, for each $1 \leqslant i \leqslant N_t^2$, the N first eigenvalues (energy values): $0 < E_1^{(i)} < E_2^{(i)} \leqslant \cdots \leqslant E_N^{(i)}$ using the direct RBF ANN designed and trained in **IV)**

(VII) : Design and train an inverse RBF ANN, using the set of coefficients generated in **V)** as a function of the set of eigenvalues obtained in **VI)**

(VIII) : A new set of data is generated using a more refined sample of the last set of coefficients: $\left\{\boldsymbol{\theta}^{(i)}\right\}_{i=1}^{N_s}$, with $N_s \gg N_t^2$.

(IX) : For each $1 \leqslant i \leqslant N_s$, calculate the N first eigenvalues (energy values): $0 < E_1^{(i)} < E_2^{(i)} \leqslant \cdots \leqslant E_N^{(i)}$ using the direct RBF ANN designed and trained in **(IV)**, from coefficients generated in **(VIII)**

(X) : Calculate the estimated coefficients $\left\{\widehat{\boldsymbol{\theta}}^{(i)}\right\}_{i=1}^{N_s}$ from eigenvalues obtained in **(IX)** using the inverse RBF ANN designed and trained in **(VII)**

(XI) : Compare real $\left\{\boldsymbol{\theta}^{(i)}\right\}_{i=1}^{N_s}$ coefficients vs estimated coefficients $\left\{\widehat{\boldsymbol{\theta}}^{(i)}\right\}_{i=1}^{N_s}$ obtained through the RBF ANN in **(X)**

(XII) : Compare the potential coefficients as a function of the first N eigenvalues: (1) calculated using inverse RBF ANN, (2) calculated with FEM using the inverse functional relationship.

In the next two subsections, we are going to validate our methodology through well known benchmark examples.

4.1. Henon–Heiles potential model

Our first numerical example corresponds to the well studied model of Henon–Heiles potential (H-H potential) (see Lagaris et al. [10]). In this case the eigenvalue problem is:

Find $E \in \mathbb{R}$ and functions $\psi(\boldsymbol{x}) \not\equiv 0$ which are solution of

$$-\frac{1}{2}\Delta\psi + V(\boldsymbol{x})\psi = E\,\psi \qquad \text{in} \quad \mathbb{R}^2, \tag{14}$$

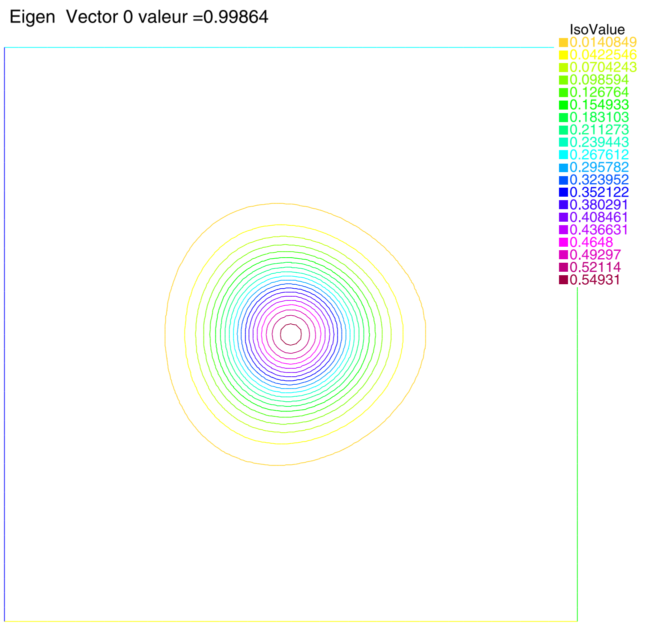
where $V(\boldsymbol{x}) = \frac{1}{2}(x^2+y^2) + \frac{1}{4\sqrt{5}}(xy^2 - \frac{1}{3}x^3)$.

In order to solve the Henon–Heiles eigenproblem (H-H eigenproblem), using the FEM, we have considered the bounded domain $\Omega = (]-6.0, 6.0[\times]-6.0, 6.0[) \subset \mathbb{R}^2$ and imposed homogeneous Dirichlet boundary conditions ($\psi = 0$ on the boundary of the domain).

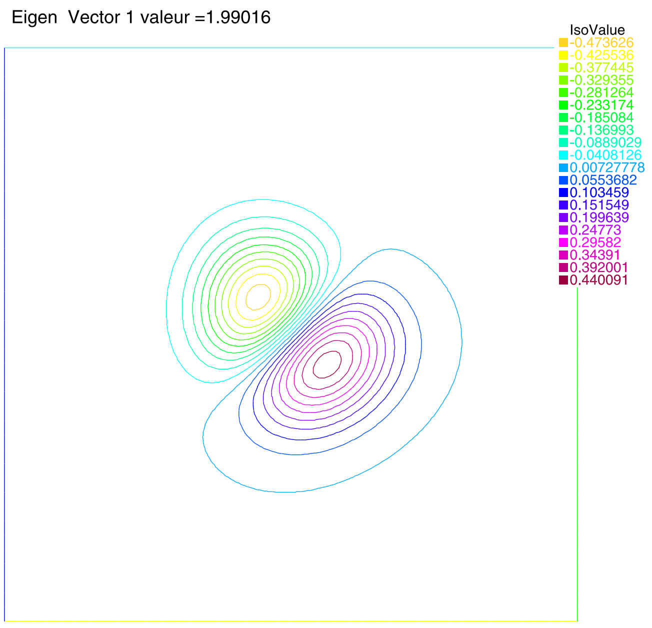
Figs. 2 and 3 qualitatively depict the numerical results associated to the first four states (wave functions and energy levels) for the H-H potential. These results have been widely obtained in the literature using different numerical methods (see [10]).

Let us now consider a generalization of the H-H potential: $V(\boldsymbol{x}, \boldsymbol{\theta}) = V(\boldsymbol{x}, \alpha, \beta, \gamma) = \alpha(x^2+y^2) + \beta xy^2 - \gamma x^3$. The potential coefficients α, β, and γ used for training the direct RBF ANN $\widehat{\mathcal{S}}_{\Omega,N}^{\omega_1}$ are: $\alpha^{(i)} = 0.3 + 80 \times t(i)$, $\beta^{(i)} = 0.08 + 10 \times t(i)$, and $\gamma^{(i)} = 0.0 + 10 \times t(i)$ where $t(i) = 0.0 + 0.0001 \times (i-1)$ with $1 \le i \le N_t^{(1)} = 61$. We employ FEM with $\mathbb{P}_2$ elements on a grid of 20×20 in Ω in order to compute the first four eigenvalues and its corresponding eigenfunctions associated with the previously defined coefficients.

We use this direct network to simulate a more larger amount of data $N_t^{(2)}$, obtaining the following set of training data for the inverse network $\widehat{\mathcal{S}}_{\Omega,N}^{\omega_2}$: $\alpha^{(i)} = 0.3 + 80 \times t(i)$, $\beta^{(i)} = 0.08 + 10 \times t(i)$, and $\gamma^{(i)} = 0.0 + 10 \times t(i)$, where $t(i) = 0.0 + 0.00001 \times (i-1)$ with $1 \le i \le N_t^{(2)} = 601$. The algorithm used to train both networks is the backpropagation algorithm.



(a) First eigenfunction ($E_1 = 0.99864$).



(b) Second eigenfunction ($E_2 = 1.99016$).

Fig. 2. Plots of isovalues of the 2 first eigenfunctions associated to the H-H eigenproblem.

Table 1 shows the application of the direct trained network using the potential coefficients $\alpha = \frac{1}{2}$, $\beta = \frac{1}{4\sqrt{5}} \approx 0.11180$, and $\gamma = \frac{1}{12\sqrt{5}} \approx 0.0372678$ to approximately compute the first four eigenvalues.

Table 2 shows the application of the inverse trained network using the four first eigenvalues documented in Lagaris et al. [10] in order to approximately compute the potential coefficients.

The results observed in Table 1 are in excellent agreement with the results presented in the literature, showing the approximation capability of the direct RBF ANN. On the other hand, it is evident from Table 2 the generalization capability of the inverse RBF ANN

Table 1
Application of the direct RBF ANN in $\varOmega = (]-6.0, 6.0[\times]-6.0, 6.0[) \subset \mathbb{R}^2$ with $N = 4$.

Direct RBF ANN	$\boldsymbol{\theta}$	Calculated eigenvalues $\widehat{E}_i$	Relative error $\frac{\lvert E_i-\widehat{E}_i\rvert}{\lvert E_i\rvert}$
$\widehat{\mathcal{S}}^{\omega_1}_{\varOmega,N}$	$\alpha = \frac{1}{2}$, $\beta = \frac{1}{4\sqrt{5}}$, $\gamma = \frac{1}{12\sqrt{5}}$	0.9995	0.000841
		1.9921	0.001001
		1.9923	0.001102
		2.9602	0.0011006

Table 2
Application of the inverse RBF ANN in $\varOmega = (]-6.0, 6.0[\times]-6.0, 6.0[) \subset \mathbb{R}^2$ with $N = 4$.

Inverse RBF ANN	Lagaris et al. [10] eigenvalues E_i	$\widehat{\boldsymbol{\theta}}$	Relative error $\frac{\lvert\theta_i-\widehat{\theta}_i\rvert}{\lvert\theta_i\rvert}$
$\widehat{\mathcal{S}}^{\omega_2}_{\varOmega,N}$	0.99866	$\widehat{\alpha} \approx 0.4991$	0.0018
	1.990107	$\widehat{\beta} \approx 0.1127$	0.0080
	1.990107	$\widehat{\gamma} \approx 0.0369$	0.0099
	2.957225		

Table 3
Application of the direct RBF ANN in $\varOmega =]-4.0, 4.0[\times]-4.0, 4.0[\subset \mathbb{R}^2$ with $N = 8$.

Direct RBF ANN	$\boldsymbol{\theta}$	Calculated eigenvalues $\widehat{E}_i$	Relative error $\frac{\lvert E_i-\widehat{E}_i\rvert}{\lvert E_i\rvert}$
$\widehat{\mathcal{S}}^{\omega_1}_{\varOmega,N}$	$\alpha = 0.5$, $\beta = 2.0$, $\gamma = 0.5$, $\delta = 1.0$	1.9915	0.000369
		4.3029	0.000520
		4.6897	0.002048
		6.8962	0.000112
		7.8299	0.001017
		7.9591	0.000025
		9.9989	0.001760
		10.5855	0.000065

when the network is applied to data outside the training set (note that the considered eigenvalues used to calculate $\widehat{\boldsymbol{\theta}}$ are the same as those reported by Lagaris et al. [10]).

4.2. 2D and 3D coupled anharmonic oscillators

Our second numerical example corresponds to the well studied models of coupled anharmonic oscillators (see Braun et al. [2]).

4.2.1. Case 2D: two coupled anharmonic oscillators

In this case the eigenvalue problem is:

Find $E \in \mathbb{R}$ and functions $\psi(\boldsymbol{x}) \not\equiv 0$ which are solution of

$$-\frac{1}{2}\Delta\psi + V(\boldsymbol{x})\psi = E\,\psi \qquad \text{in} \quad \mathbb{R}^2, \tag{15}$$

where $V(\boldsymbol{x}) = \frac{1}{2}(x^2+y^2) + 2(x^4+y^4) + \frac{1}{2}(x^6+y^6) + xy$.

In order to solve the Two Coupled Anharmonic Oscillators (2CAO) eigenproblem, using the FEM approach, we have considered a 32 ×32 grid in the square domain $\varOmega =]-4.0, 4.0[\times]-4.0, 4.0[\subset \mathbb{R}^2$ and imposed a homogeneous Dirichlet condition, $\psi = 0$ on the boundary of the domain.

Let us consider a generalization of the 2CAO potential: $V(\boldsymbol{x},\boldsymbol{\theta}) = V(\boldsymbol{x},\alpha,\beta,\gamma,\delta) = \alpha(x^2+y^2)+\beta(x^4+y^4)+\gamma(x^6+y^6)+\delta xy$. In this case the potential coefficients α, β, γ, and δ used for training the direct RBF ANN $\widehat{\mathcal{S}}^{\omega_1}_{\varOmega,N}$ are chosen as follows: $t(i)$ is chosen uniformly in [0, 1], $\alpha^{(i)} = t(i)$, $\beta^{(i)} = 1.5 + t(i)$, $\gamma^{(i)} = 0.2 + t(i)$, and $\delta = 0.5+t(i)$, where $1 \le i \le N_t^{(1)} = 81$. The eight first eigenvalues $0 < E_1^{(i)} < E_2^{(i)} \leqslant \cdots \leqslant E_8^{(i)}$ associated to the generated coefficients are calculated using FEM with $\mathbb{P}_2$ elements in $\varOmega$.

We use the designed direct network in order to simulate a larger amount of data $N_t^{(2)}$, obtaining the set of training data for the inverse network $\widehat{\mathcal{S}}^{\omega_2}_{\varOmega,N}$. In this case $t(i)$ is also chosen uniformly in $[0,1]$, $\alpha^{(i)} = t(i)$, $\beta^{(i)} = 1.5+t(i)$, $\gamma^{(i)} = 0.2+t(i)$, and $\delta = 0.5+t(i)$, where $1 \le i \le N_t^{(2)} = 801$. The algorithm used to train both networks is the backpropagation algorithm.

Table 3 shows the results when we apply the direct neural network (trained) to calculate the eigenvalues provided the potential coefficients, whereas Table 4 shows the results of applying the inverse neural network (trained) to calculate the potential coefficients provided the eight first eigenvalues documented in [2]. It can be observed that our direct and inverse calculations are in accordance with results reported in [2].

In addition, it is worth noting that the neural networks designed cannot have a better convergence order than FEM with $\mathbb{P}_2$ elements, since this is the methodology used to train the direct RBF ANN. However, the error can be improved by using, for training the direct network, methods with better convergence order.

4.2.2. Case 3D: three coupled anharmonic oscillators

Let us now consider $V(\boldsymbol{x}) = \frac{1}{2}(x^2+y^2+z^2)+2(x^4+y^4+z^4)+\frac{1}{2}(x^6+y^6+z^6)+xy+yz+zx$.

We solve the Three Coupled Anharmonic Oscillators (3CAO) eigenproblem in the bounded domain $\varOmega =]-4.0, 4.0[\times]-4.0, 4.0[\times]-4.0, 4.0[\subset \mathbb{R}^3$, where we have imposed again a homogeneous Dirichlet boundary condition. We have considered a 28 ×28 ×28 grid in $\varOmega$.

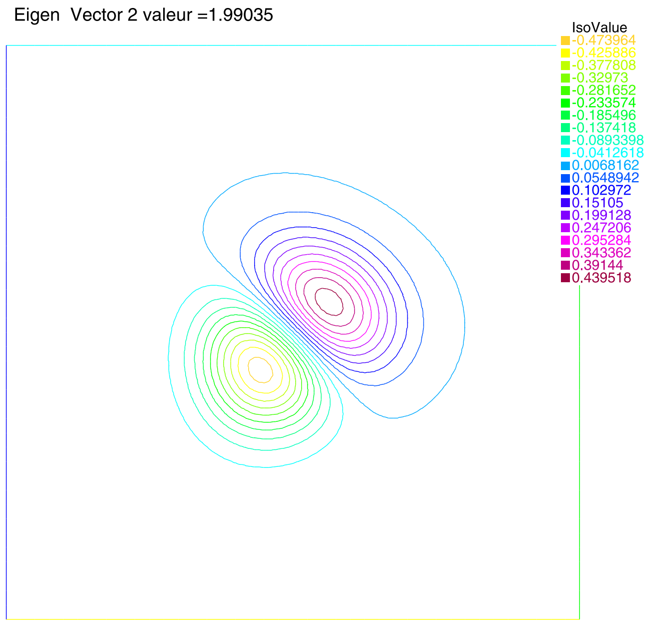
A generalization of the 3CAO potential is considered: $V(\boldsymbol{x},\boldsymbol{\theta}) = V(\boldsymbol{x},\alpha,\beta,\gamma,\delta) = \alpha(x^2+y^2+z^2)+\beta(x^4+y^4+z^4)+\gamma(x^6+y^6+z^6)+\delta(xy+yz+zx)$.

For training the direct and inverse networks, we use the same procedure employed for the 2CAO case. However, we employ FEM with $\mathbb{P}_2$ tetrahedral elements in $\varOmega$ in this case.

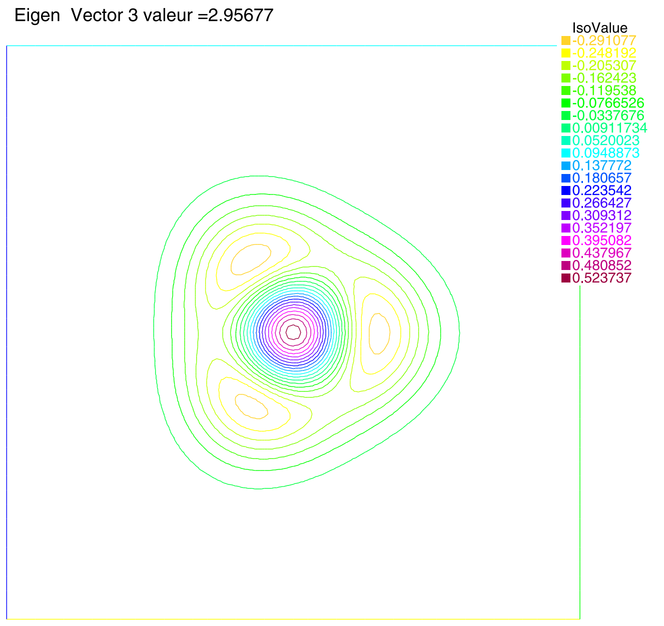
The application of the direct trained network in order to compute the two first eigenvalues, provided the potential coefficients, as well as, the application of the inverse trained network in order to compute the potential coefficient, provided the two first eigenvalues documented in Braun et al. (see [2]), are presented in Tables 5 and 6, respectively.

It can be observed from Tables 5 and 6 that the performance achieved by the direct and inverse RBF ANN is good. However, the accuracy of the calculation made by both networks undergoes a small decrease, in relation to the 2D case, which we attribute to the 3D geometry of this example. Consequently, the numerical results presented above show the validity of our approach, and also shows its limitations when considering 3D geometries.

It is well known that the FEM is a numerical technique that requires a bounded domain in order to calculate the approximate



(a) Third eigenfunction ($E_3 = 1.99035$).



(b) Fourth eigenfunction ($E_4 = 2.95677$).

Fig. 3. Plots of isovalues of the third and fourth eigenfunctions associated to the H-H eigenproblem.

solution. Thus it is necessary to truncate the domain when working with unbounded domains. In the first and second examples, we have truncated the domains in accordance with the literature (see Lagaris et al. [10] and Braun et al. [2]). However, in practice the domains can be truncated using analytical techniques, mainly based on Fourier Analysis, especially customized to each problem. In addition, in some cases, the boundary could vary depending on the values of the potential coefficients.

4.3. Assessing the performance of our methodology

Once validated our methodology through benchmark examples, we must evaluate its performance. Let us define for this $\Omega =]0.0, 1.0[\times]0.0, 1.0[\subset \mathbb{R}^2$ and consider the following potential function:

$$V(\boldsymbol{x}, \boldsymbol{\theta}) = \frac{\beta \exp(-\alpha x^2)}{|(x + 0.0000001)||(x - 1.0000001)|} + \frac{(20 - \beta)\exp(-\alpha y^2)}{|(y + 0.0000001)||(y - 1.0000001)|} \qquad (16)$$

where $\boldsymbol{\theta} = (\alpha, \beta)^T$.

For simplicity let us put $\frac{\hbar^2}{2m} = 1$.

The potential coefficients used for training the direct RBF ANN are: $\alpha^{(i)} = t(i)$ and $\beta^{(i)} = 20/(1+t(i))$ where $t(i) = 1.0+0.1\times(i-1)$ with $1 \le i \le N_t^{(1)} = 51$.

Once trained the network $\widehat{\mathcal{S}}^{\omega_1}_{\Omega,N}$, using the FEM technique with $\mathbb{P}_2$ elements on a grid of 20×20 in Ω (see Section 3), and calculated the associated vector $\widehat{\boldsymbol{\omega}}_1$, we use this direct network to simulate a larger amount of data $N_t^{(2)}$, obtaining a set of training data for the inverse network $\widehat{\mathcal{S}}^{\omega_2}_{\Omega,N}(\boldsymbol{\theta})$. In this case $1 \le i \le N_t^{(2)} = 501$, $\alpha^{(i)} = t(i)$, $\beta^{(i)} = 20/(1 + t(i))$ and $t(i) = 1.0 + 0.01 \times (i - 1)$. This last training data gives us the value of $\widehat{\boldsymbol{\omega}}_2$. We have used in this case the Levenberg–Marquardt algorithm to train both networks.

Fig. 4 shows a comparison of the coefficients evolution as a function of the first $N = 3$ eigenvalues of the Schrödinger operator, when the value of the simulated data is $N_s = 5001$, $\alpha^{(i)} = t(i)$, $\beta^{(i)} = 20/(1 + t(i))$ and $t(i) = 1.0 + 0.001 \times (i - 1)$. It can be observed in this figure that the coefficients calculated from the neural network method approach quite well to those calculated using the inverse functional relationship obtained with FEM. Fig. 5 depicts the relative errors of the calculated coefficients associated to the comparison shown in Fig. 4.

Finally, Table 7 summarizes the computational performance using the mean squared error (MSE), the computational time, in seconds, using RBF ANN (CT ANN) directly applied to the set of eigenvalues, and also the computational time, in seconds, using the inverse functional relationship between the coefficients associated to a potential function and eigenvalues obtained with the FEM technique (CT FEM), required for simulations. The computer used to obtain the above results has a 2.4 GHz Intel Core 2 Duo processor with 3GB 800 MHz DDR2 SDRAM.

The CT ANN is obtained taking into account the computational time required to calculate the training data, through FEM, needed by the first network in each example. We observe from Table 7 the excellent computational time obtained by using the RBF ANN compared with the computational time obtained by using the FEM procedure, as well as, the good computational performance which is measured using the MSE.

Finally, the main advantage that we want to emphasize about our approach is the computational time (CT) required for the calculations. The MSE is shown (see Table 7) in order to quantitatively demonstrate that the accuracy is not lost using the ANN method. In addition, it is worth noting that the CT ANN considers the total time, including the time employed for training the direct network using FEM, allowing a comparison of the techniques used.

5. Conclusion

In this article, we have proposed a numerical method based on an artificial neural network which allows to compute the coefficients of an unknown potential function in quantum mechanics, using a finite set of eigenvalues of the Schrödinger operator. Once trained a direct RBF ANN, using FEM in Ω, an inverse RBF ANN has been trained, using as a training set the data calculated by computational simulation of the direct network and considering the inverse functional relationship between the calculated eigenvalues and the potential coefficients.

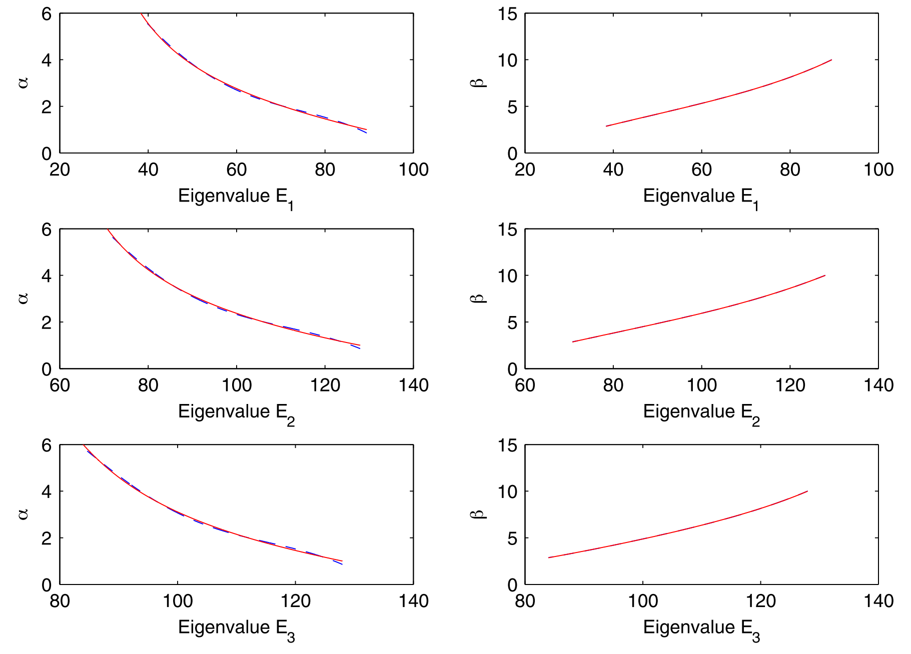


Fig. 4. Coefficients of the potential as a function of the first $N = 3$ eigenvalues: (1) calculated using inverse RBF ANN (in dashed line), (2) calculated with FEM using the inverse functional relationship (in solid line).

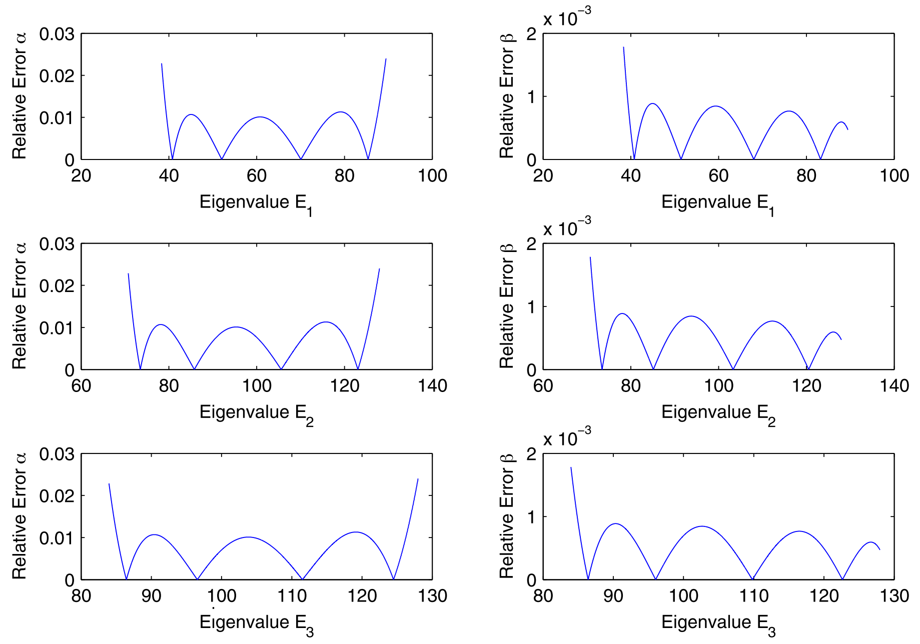


Fig. 5. Relative error of the calculated coefficients.

Table 4
Application of the inverse RBF ANN in $\Omega =]-4.0, 4.0[\times]-4.0, 4.0[\subset \mathbb{R}^2$ with $N = 8$.

Inverse RBF ANN	Braun et al. [2] eigenvalues E_i	$\widehat{\boldsymbol{\theta}}$	Relative error $\frac{\lvert\theta_i-\widehat{\theta}_i\rvert}{\lvert\theta_i\rvert}$
	1.99223576		
	4.30513845		
	4.69932314	$\widehat{\alpha} \approx 0.5017$	0.0034
$\widehat{\mathcal{S}}^{\omega_2}_{\Omega,N}$	6.89542638	$\widehat{\beta} \approx 1.9824$	0.0088
	7.83787029	$\widehat{\gamma} \approx 0.5004$	0.0008
	7.95930124	$\widehat{\delta} \approx 1.0115$	0.0115
	10.01652920		
	10.58618828		

We have tested our method with three examples, showing well known results for the direct problem in the first two models, hence validating our method, and then evaluating the performance for the inverse problem in the third model. The numerical results

Table 5
Application of the direct RBF ANN in $\Omega =]-4.0, 4.0[\times]-4.0, 4.0[\times]-4.0, 4.0[\subset \mathbb{R}^3$ with $N = 2$.

Direct RBF ANN	$\boldsymbol{\theta}$	Calculated eigenvalues $\widehat{E}_i$	Relative error $\frac{\lvert E_i-\widehat{E}_i\rvert}{\lvert E_i\rvert}$
$\widehat{\mathcal{S}}^{\omega_1}_{\Omega,N}$	$\alpha = 0.5$, $\beta = 2.0$, $\gamma = 0.5$, $\delta = 1.0$	2.9921 5.3018	0.004633 0.001097

Table 6
Application of the inverse RBF ANN in $\Omega =]-4.0, 4.0[\times]-4.0, 4.0[\times]-4.0, 4.0[\subset \mathbb{R}^3$ with $N = 2$.

Inverse RBF ANN	Braun et al. [2] eigenvalues E_i	$\widehat{\boldsymbol{\theta}}$	Relative error $\frac{\lvert\theta_i-\widehat{\theta}_i\rvert}{\lvert\theta_i\rvert}$
$\widehat{\mathcal{S}}^{\omega_2}_{\Omega,N}$	2.97830266 5.29599234	$\widehat{\alpha} \approx 0.4895$ $\widehat{\beta} \approx 1.9656$ $\widehat{\gamma} \approx 0.5143$ $\widehat{\delta} \approx 1.009$	0.0210 0.0172 0.0286 0.0090

Table 7
Summary of computational performance and time for the numerical example using $N_s = 5001$ simulation data.

N_s	MSE α	MSE β	CT ANN	CT FEM
5001	0.0026	4.2947e−05	14.15	423.94

show that the calculation of these unknown potential coefficients, through the eigenvalues, using our neural network approach is very efficient. In other words the relative error is acceptable, and the computation time is significantly smaller compared to the finite element technique (see Table 7). However as the geometry of the domain Ω (see Second example, 3D case) becomes more complex more training data will be needed for the direct RBF ANN, and therefore more computational time will be used in our approach, in order to keep the accuracy of the calculation acceptable. Finally, our neural network based method has shown that it can be successfully implemented, as an approximate method for obtaining: 1.- The energy levels for a given particle; 2.- The unknown potential coefficients associated to a quantum system.

In summary the main advantage of our methodology is that all the computation process using neural networks, including the training process, the validation process and the simulation process, requires a notably lower computational time than the FEM technique, with a good calculation error performance.

Acknowledgments

S. Ossandón acknowledges support from the European Unions Horizon 2020, research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 644202: Geophysical Exploration using Advanced Galerkin Methods (GEAGAM). C. Reyes thanks the partial support from Project Innova-Chile CORFO No. 10CEII-9007: "CSIRO-CHILE International Centre of Excellence in Mining and Mineral Processing", Program 3-Project 1. The work of P. Cumsille was founded by PIA-CONICYT under grant PFBasal-01. Finally, C. M. Reyes acknowledges support from Grant Fondecyt No. 1140781, DIUBB No. 141709 4/R and the group of Fisica de Altas Energias of the Universidad del Bio-Bio.

References

[1] M. Pillai, J. Goglio, T.G. Walker, Amer. J. Phys. 80 (11) (2012) 1017–1019.
[2] M. Braun, S.A. Sofianos, D.G. Papageorgiu, I.E. Lagaris, J. Comput. Phys. 126 (1996) 315–327.
[3] H. Ishikawa, J. Phys. A: Math. Gen. 35 (2002) 4453–4476.
[4] R. Kannan, A. Masud, J. Appl. Mech. 76 (2) (2009).
[5] N. Watanabe, M. Tsukada, J. Phys. Soc. Japan 69 (9) (2000).
[6] T. Simos, P. Williams, J. Comput. Appl. Math. 79 (1997) 189–205.
[7] S. Ossandón, C. Reyes, C. R. Méch. 344 (2016) 113–118.
[8] S. Ossandón, C. Reyes, C.M. Reyes, Comput. Math. Appl. 72 (2016) 1153–1163.
[9] T. Poggio, F. Girosi, Neural Comput. 10 (6) (1998) 1445–1454.
[10] I. Lagaris, A. Likas, D. Fotiadis, Comput. Phys. Comun. 104 (1997) 1–14.
[11] Y. Shirvany, M. Hayati, R. Moradian, Commun. Nonlinear Sci. Numer. Simul. 13 (10) (2008) 2132–2145.
[12] R.J. Schilling, J.J. Carroll Jr., A.F. Al-Ajlouni, IEEE Trans. Neural Netw. 12 (1) (2001) 1–15.
[13] I. Babuska, J. Osborn, in: P.G. Lions (Ed.), Finite Element Methods (Part 1), in: Handbook of Numerical Analysis, vol. II, North-Holland, Amsterdam, 1991, pp. 641–787.
[14] D. Boffi, Acta Numer. 19 (2010) 1–120.
[15] D. Boffi, F. Brezzi, M. Fortin, Mixed Finite Element Methods and Applications, in: Springer Series in Computational Mathematics, vol. 44, Springer, Heidelberg, 2013.
[16] F. Brezzi, M. Fortin, Mixed and Hybrid Finite Element Methods, in: Springer Series in Computational Mathematics, vol. 15, Springer-Verlag, 1991.
[17] P.G. Ciarlet, The Finite Element Method for Elliptic Problems, North-Holland, Amsterdam, 1978.
[18] B. Mercier, J. Osborn, J. Rappaz, P.A. Raviart, Math. Comp. 36 (1981) 427–453.