AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Informatyki

Praca dyplomowa

Neural networks in solving differential equations

Sieci neuronowe w rozwiązywaniu równań różniczkowych

Autor: *Piotr Wróbel*
Kierunek studiów: *Informatyka*
Opiekun pracy: *dr inż. Marcin Kuta*

Kraków, 2023

Contents

List of abbreviations and symbols

CNN	Convolutional Neural Network
Δ	Laplace operator, for n variable function: $\Delta = \sum_{i=1}^{n} \frac{\partial^2}{\partial i^2}$
DE	Differential Equation
DFT	Density Functional Theory
DFTB	Density Functional based Tight-Binding
EC	ethylene carbonate
F1EC	monofluoroethylene carbonate
F2EC	*trans*-difluoroethylene carbonate
IR	infrared
L_2	L_2 norm of the error
MAE	mean absolute error
MD	molecular dynamics
MSE	mean squared error
N_u	number of training points
N_f	number of collocation points
NN	neural network
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PDE	Partial Differential Equation
PINN	physics informed neural network
t^n	n-th time step
u	the solution of a differential equation

Chapter 1

Introduction

The following chapter presents the basics and motivation for the topic of this thesis, the research hypothesis, and the goals of the work.

1.1 Motivation

Today, simulation techniques are very influential for people. Obtained results are widespread and have a wide range of applications, from weather forecasting, the prediction of vulnerable points in buildings, to the design of new drugs or materials. A successful simulation needs an appropriate model of physics involved in the described phenomenon. In most cases, this description is made by a single differential equation or a system of them.

Since the development of quantum theory, solving Schrödinger equation has become one of the fundamental problems in modern chemistry. Proper electron structure of a given system obtained by solving this equation is the basis in studying macroscopic properties, such as excitation energy, crystaline structure, energy effects of chemical reactions or the most efficient structure of a drug [1].

One of the recently studied topics in chemistry is the development of novel electrolytes for batteries [2]. The background is mostly environmental: lithium, which is commonly used today in batteries, has a rather low abundance in the Earth's crust and is very hard to recycle [3, 4]. Electrolytes that contain widespread elements (such as sodium or magnesium) instead of lithium are now considered potential alternatives [5, 6, 7]. Experimental work could be supported by theoretical research that allows cheaper modeling of new system properties such as conductivity or influence of sodium/magnesium compounds addition to a solution [8, 9, 10]. One of such systems, which were studied, are sodium salt solutions in ethylene carbonate (EC) [11] and its fluorinated derivatives [12, 13, 14, 15].

The primary method for conducting theoretical research in this case is molecular dynamics (MD), which enables the calculation of dynamic properties and interactions present in the condensed phase. The simulation results also in data that could be used to calculate the infrared (IR) spectrum, giving important information about interactions within the system. The most crucial part of such simulation is the numerical solving of the Schrödinger equation for

multielectron system, which is also the most time-consuming step. A desirable improvement would be to implement a faster method without a significant loss of accuracy. Systems with EC were recently studied with standard MD methods [16].

A recently introduced concept of Physics Informed Deep Neural Networks (PINNs) [17, 18, 19] is one of the possible new approaches to such problems. By learning from data obtained by computationally expensive methods, it is possible to predict new results in a much faster way. Thus, incorporating PINNs and other Neural Networks (NNs) methods for this purpose is the topic of this thesis.

1.2 Aims of the work

Neural networks (NNs) are models built in a way inspired by biological systems, including the human brain. They are, as all machine learning models, trained on existing data. Among their advantages are uncomplicated architecture and way of training, short prediction times, and the ability to generalize. Therefore, NNs are commonly used today in a wide range of applications [20]. Some results of their application to solving differential equations are described in the next chapter of this thesis. The limitations of standard numerical methods involve time of obtaining a solution and also concentration only on solving one given unique problem. NNs are capable of training on data from many problems, so they allow using them for a range of problems instead of the only one. Recently, an approach involving usage of NNs for solving differential equations has been proposed, such NNs are called Physics Informed Neural Networks (PINNs).

In this work, as a preliminary task, PINNs have been used to solve the Schrödinger equation for four standard quantum chemical problems with a known analytical form of the exact solution: particle in a potential box, harmonic oscillator, rigid rotor and hydrogen atom. The main aim of the work is to apply NNs to accelerate the process of performing MD simulation for one of the potential electrolytes, which are ethylene carbonate solutions (EC) of metal salts. For this purpose, the NN is trained on the data set that contains results obtained from one of the commonly used methods in computational chemistry, the Density Functional based Tight-Binding (DFTB) [21] method. The obtained model is used to predict atomic charges, as in the standard DFTB. On the basis of these charges, forces acting on atoms are calculated. Then, the atoms are moved in a given period of time. The whole process is repeated, what results in MD simulation. This simulation is then used to calculate the theoretical infrared (IR) spectrum, which is compared with the one obtained from standard DFTB as a form of verification of the NN approach.

Usually, NNs were considered as a pure computer science problem. Even now, their usage in computational chemistry is a novel and non-standard approach [22]. Thus, this work inter-disciplinary deploys computer science methods in the field where they were not very common. Furthermore, the idea of using predicted partial charges to accelerate MD simulation is a new approach, which has not been used before.

1.3 Research hypothesis

The preliminary hypothesis of this work is that the Schrödinger equation can be effectively solved with Physics Informed Deep Neural Networks (PINN) for standard model systems. The influence of some parameters on the quality of the results is examined. The parameters involve the number of training points, their distribution, and the architecture of the PINN.

The main hypothesis is that for small and medium quantum chemical systems (up to approximately 500 atoms), molecular dynamics (MD) simulations with NN enhanced DFTB can be effectively performed with small losses of accuracy. For accuracy verification, the IR spectrum obtained from the proposed approach is compared with the one obtained from the standard DFTB method.

Chapter 2

Differential equations

This chapter introduces concepts from differential equations theory, which are crucial for this work.

2.1 Definition

The differential equation (DE) is an equation that involves functions and their derivatives, where at least one of them is unknown [23]. In general, except for basic cases, it is difficult to obtain an analytical solution. Thus, for practical reasons, numerical methods of solving them are usually applied.

Depending on the character of the unknown function, differential equations could be divided into two groups:

- ordinary differential equations (ODE), where an equation involves a function of only one variable (here $u(x)$), for example,

$$\frac{\mathrm{d}u}{\mathrm{d}x} - \sin x = 0, \tag{2.1}$$

- partial differential equations (PDE), where multivariate functions (here $u(x, y)$) are involved together with their partial derivatives, for example.

$$\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0. \tag{2.2}$$

In terms of linearity, two groups of differential equations could be defined:

- linear differential equations, where all dependencies on the unknown function and its derivatives are linear, for example,

$$\frac{\mathrm{d}u}{\mathrm{d}x} + 3u - 4 = 0, \tag{2.3}$$

- non-linear differential equations, where there is at least one nonlinearity in the dependence

on the unknown function or its derivatives, for example

$$\left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)^2 - 2u + 3x = 0. \tag{2.4}$$

Additionally, differential equations are described by their order. It is defined as the highest derivative order present in the equation. All equations presented above are first order equations, while the equation

$$\frac{\mathrm{d}^3u}{\mathrm{d}x^3} + \frac{\mathrm{d}u}{\mathrm{d}x} - 3u = 0 \tag{2.5}$$

is the third order equation.

Without additional conditions, the solution of a differential equation is a family of functions. To obtain a unique solution, it is needed to provide boundary conditions in number equal to the order of the equation. There are two basic types of boundary conditions:

- Dirichlet (first-type) boundary conditions - here, the exact values for a solution at a boundary of its domain are given. For example, for the equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \tag{2.6}$$

  Dirichlet boundary conditions force that:

$$u(x, y) = \sin x \cos y, \quad \forall (x, y) \in \partial\Omega, \tag{2.7}$$

  where Ω is the solution domain and $\partial\Omega$ is its boundary.

- Neumann (second-type) boundary conditions - here, instead of values of a solution at a boundary, the values of its derivative are given. For example, for the equation

$$\frac{\mathrm{d}^2u}{\mathrm{d}x^2} - u + 2 = 0, \tag{2.8}$$

  Neumann boundary conditions force that:

$$\frac{\mathrm{d}u}{\mathrm{d}x}\Big|_{x=0} = u_0, \quad \frac{\mathrm{d}u}{\mathrm{d}x}\Big|_{x=2} = u_2. \tag{2.9}$$

The existence of the solution of the DE is assured by the Pickard-Lindelöf theorem. Let D be the domain of the problem such that: $D \subseteq \mathbb{R} \times \mathbb{R}^n$ with given point (t_0, y_0) inside D. Let $f(t, y)$ be a function such that: $f(t, y) : D \to \mathbb{R}^n$ and f is continuous in t and Lipschitz continuous in y, i.e., there exists constant value L that satisfies the condition for every (t, y_1), (t, y_2) inside D:

$$|f(t, y_1) - f(t, y_2)| \leq L\,|y_1 - y_2|\,. \tag{2.10}$$

Then, there exists $\varepsilon > 0$ for which the Cauchy problem:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f\left(t, y(t)\right), \tag{2.11}$$

$$y(t_0) = y_0, \tag{2.12}$$

has a unique solution $y(t)$ defined on the interval $[t_0 - \varepsilon, t_0 + \varepsilon]$.

2.2 Importance of differential equations

Differential equations are quite widespread in describing many physical phenomena, starting from engineering issues like thermal conductivity and ending with the evolution of states of quantum particles. In general, computational methods for predicting various properties of materials are cheaper than performing experiments; therefore, with increasing computing power, these methods have become more and more essential for modern science. Usually the crucial issue in simulation is solving some particular kind of differential equation. For example

- Newton's equation of motion [24]:

$$\vec{F} = m\frac{\mathrm{d}^2\vec{r}}{\mathrm{d}t^2}, \tag{2.13}$$

  where $\vec{F}$ stands for force, m stands for mass, $\vec{r}$ stands for position, and t stands for time. This equation describes motion of a point particle with non-zero mass when it is influenced by an external force, crucial for, i.e., the description of the movements of planets [24] or the motion of atomic nuclei in molecular dynamics [25].

- Poisson's equation [26]:

$$\Delta\varphi = \frac{\rho}{\epsilon_0}, \tag{2.14}$$

  where Δ is the Laplace operator, φ stands for electrical potential, ρ stands for electric charge density, and ϵ_0 stands for vacuum electric permittivity. This is crucial in any problem involving the movement of charged particles in an electric field, because knowing the potential φ allows one to calculate the electrical component for the force in equation 2.13.

- Schrödinger's equation [27]:

$$i\hbar\frac{\partial\Psi(x,t)}{\partial t} = \left(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x)\right)\Psi(x,t), \tag{2.15}$$

  where i is the imaginary unit, $\hbar$ is the reduced Planck constant, Ψ represents the wave function of the system and $V(x)$ is the potential description function. This equation is basic in non-relativistic quantum mechanics, it describes the time evolution of the system wavefunction (module square of which is the probability density of finding a particle at position x). Almost all problems considered in quantum chemistry are related to solving it, i.e. the calculation of ionization energies, the potential energy surface [28] or the determination of the potential for ab initio molecular dynamics [25].

2.3 Standard numerical methods

The importance of DEs has led to the development of many numerical methods to solve them. Among the most popular are methods based on the Taylor series approximation or Runge-Kutta methods of various orders [29].

One of the simplest methods is the first order Runge-Kutta method, also known as the Euler method. In this approach, for a differential equation in the form:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y(t)), \tag{2.16}$$

with a given initial condition:

$$y(t_0) = y_0, \tag{2.17}$$

and arbitrarily chosen time step h, the value of the function y in the (n+1)-th step is calculated as:

$$y_{n+1} = y_n + hf(t_n, y_n). \tag{2.18}$$

For a particular example of simulating atomic motions in a molecular system, step h could be interpreted as the amount of time, for which it is assumed that the potential of the system remains unchanged. This is the reason why this method is sensitive to the magnitude of h, which is one of the standard issues in molecular dynamics [25].

Chapter 3

Related works

At this moment, deep learning methods are commonly used in a wide range of applications, for instance, in object recognition, voice synthesis, prediction of sale trends or optical character recognition. In recent years, works considering the usage of neural networks as a new computational method to solve (mostly partial) differential equations have been published. This chapter briefly describes three proposed approaches in [17, 18, 19] and also some other available works on this topic. In addition, an overview of the software available utilizing these methods is included, and some open problems in this field are mentioned.

3.1 Physics informed deep learning

In this subsection, the following notation for partial derivatives is used:

$$u_x = \frac{\partial u}{\partial x}, \quad u_{xx} = \frac{\partial^2 u}{\partial x^2}, \quad u_{xt} = \frac{\partial^2 u}{\partial x \partial t}, \quad ... \tag{3.1}$$

In [17, 18, 19] physics informed deep learning method is introduced. It relies on the use of a neural network (NN) to predict the solution of a partial differential equation (PDE) in general form:

$$u_t + \mathcal{N}[u, u_x, u_{xx}, ...; \lambda] = 0, \tag{3.2}$$

where u is the seeked function and $\mathcal{N}$ is a non-linear differential operator with hyperparameters λ. NN training is performed using the PDE itself and some initial or boundary conditions. Thus, this approach is an example of unsupervised learning. Data completed from an analytically solved equation or measured experimentally is used for evaluation of results. Depending on the knowledge of $\mathcal{N}$, three approaches could be distinguished: white box, gray box, and black box. The authors also describe two different cases with respect to treatment of time: continuous and discrete. Such NN is called Physics Informed deep Neural Network (PINN).

3.1.1 White-box

In [17] the white-box approach has been described in which the representation of $\mathcal{N}$ and parameters λ are known. In the following, an example with Burger's equation is presented for both

continuous- and discrete-time models.

Continuous time model

In this example, a partial differential equation

$$u_t + uu_x - \frac{0.01}{\pi} u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1], \tag{3.3}$$

with boundary conditions

$$u(0, x) = -\sin(\pi x), \tag{3.4}$$

$$u(t, -1) = u(t, 1) = 0, \tag{3.5}$$

is considered.

We want to obtain a neural network that gets t and x as input and predicts the value $u(t, x)$. Next, the $f(t, x)$ function is defined as:

$$f(t, x) = u_t + uu_x - \frac{0.01}{\pi} u_{xx}. \tag{3.6}$$

The neural network is trained using N_u training points from an experiment or a known solution $u(t_i, x_i)$, N_{initial} initial points in a form $u(0, x_i)$, N_b boundary points in a form $u(t_i, \pm 1)$ as well as N_f points for which $f(t, x)$ vanishes (collocation points). The loss function has the following form:

$$\begin{aligned} MSE &= MSE_{\text{initial}} + MSE_b + MSE_f, \\ MSE_{\text{initial}} &= \frac{1}{N_{\text{initial}}} \sum_{i=1}^{N_{\text{initial}}} \left| u(0, x_i) + \sin(\pi x_i) \right|^2, \\ MSE_b &= \frac{1}{N_b} \sum_{i=1}^{N_b} \left| u(t_i, \pm 1) \right|^2, \\ MSE_f &= \frac{1}{N_f} \sum_{i=1}^{N_f} \left| f(t_i, x_i) \right|^2, \end{aligned} \tag{3.7}$$

where MSE_{initial} is responsible for satisfying the initial condition (in the example Equation 3.4), MSE_b is responsible for satisfying boundary conditions (Equation 3.5), and MSE_f enforces that (3.3) is satisfied.

To measure accuracy of a solution predicted by PINN, relative L_2 errors are calculated, which are defined as

$$L_2 = \sqrt{\frac{\sum_i^N \left(u_{\text{pred}}(t_i, x_i) - u_{\text{exact}}(t_i, x_i) \right)^2}{\sum_i^N u_{\text{exact}}^2(t_i, x_i)}}, \tag{3.8}$$

where u_{pred} is the predicted function and u_{exact} is the exact function.

Figure 3.1 presents the results obtained for NN with 9 layers with 20 neurons each trained with $N_u = 100$ and $N_f = 10000$. The relative error L_2 is equal to $6.7 \cdot 10^{-4}$ for this case. The authors also examined cases with a larger number of neurons, larger N_u or N_f, which showed that increasing each of them lowers the error.

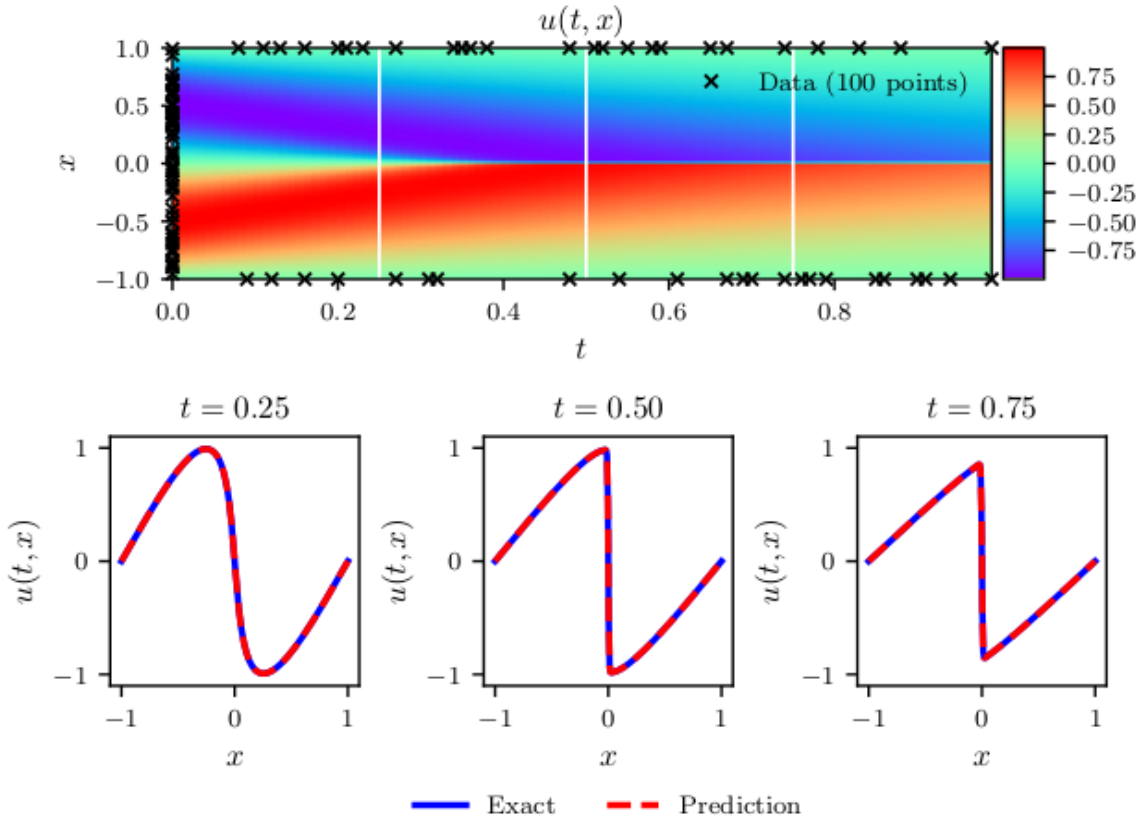


Figure 3.1: Results obtained for Burger's equation in white-box approach with continuous time, top: predicted solution $u(t, x)$ with training points marked with crosses, bottom: comparison between predicted and exact solution for three snapshots marked with vertical white lines in the top picture [17].

This approach has also been successful for complex functions with periodic boundary conditions, which was verified by solving the Schrödinger equation.

The continuous time model provides accurate solutions in the mentioned cases, however, its main disadvantage is a requirement for large N_f to work correctly. For the 1D example described above, it was not a significant issue, but for 2D and 3D functions the necessary number of points would be enormous. This issue led to the development of a discrete time model.

Discrete time model

In this method, the Runge-Kutta scheme with q stages is applied:

$$\begin{aligned} u_i^n &= u^{n+c_i} + \Delta t \sum_{j=1}^{q} a_{ij} \mathcal{N}[u^{n+c_j}], \quad i = 1, ..., q, \\ u_{q+1}^n &= u^{n+1} + \Delta t \sum_{j=1}^{q} b_j \mathcal{N}[u^{n+c_j}], \end{aligned} \tag{3.9}$$

where $u^{n+c_j}(x) = u(t_n + c_j \Delta t, x)$ for $j = 1, ..., q$. The neural network takes x as an input and outputs a vector:

$$\left[u_1^n(x), ..., u_q^n(x), u_{q+1}^n(x)\right]. \tag{3.10}$$

The nonlinear operator $\mathcal{N}$ is given by:

$$\mathcal{N}[u^{n+c_j}] = u^{n+c_j} u_x^{n+c_j} - \frac{0.01}{\pi} u_{xx}^{n+c_j}. \tag{3.11}$$

In this approach, data is available for time t_n, and NN predicts the solution at t_{n+1}. This scheme could also be used with NN training on the solution for t_{n+1} to obtain the output for t_{n+2} and repeated for the number of times requested.

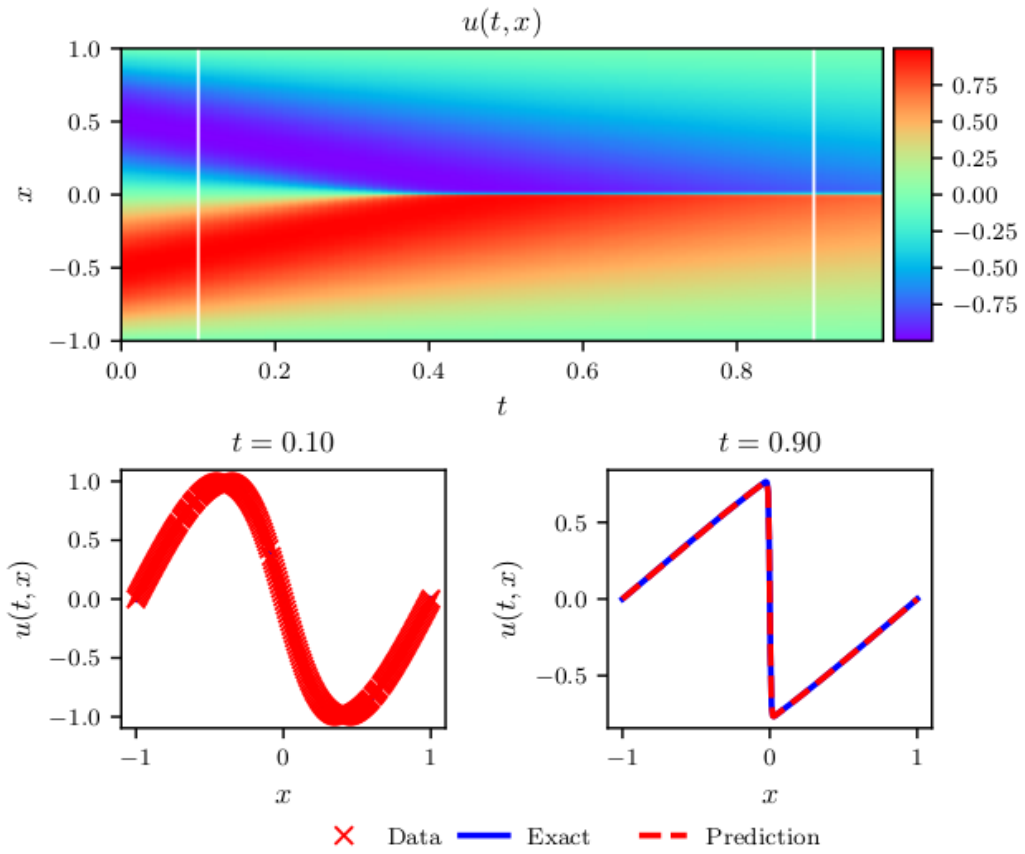


Figure 3.2: Results obtained for Burger's equation in white-box approach with discrete time, top: solution $u(t, x)$ with initial snapshot with training data at $t = 0.1$ and prediction snapshot at $t = 0.9$ marked with white vertical lines, bottom: comparison between predicted and exact solution for $t = 0.1$ and $t = 0.9$ [17].

Figure 3.2 shows the result obtained for $q = 500$, 500 training points, and $\Delta t = 0.8$. PINN consisted of 4 layers with 50 neurons each. The relative L_2 error in this case is equal to $8.2 \cdot 10^{-4}$.

The authors stress that in the Runge-Kutta method Δt usually needs to be small to obtain reasonable solutions, while $\Delta t = 0.8$ used here is a rather large value. In addition, an increasing number of stages q is claimed to allow the use of larger values of Δt.

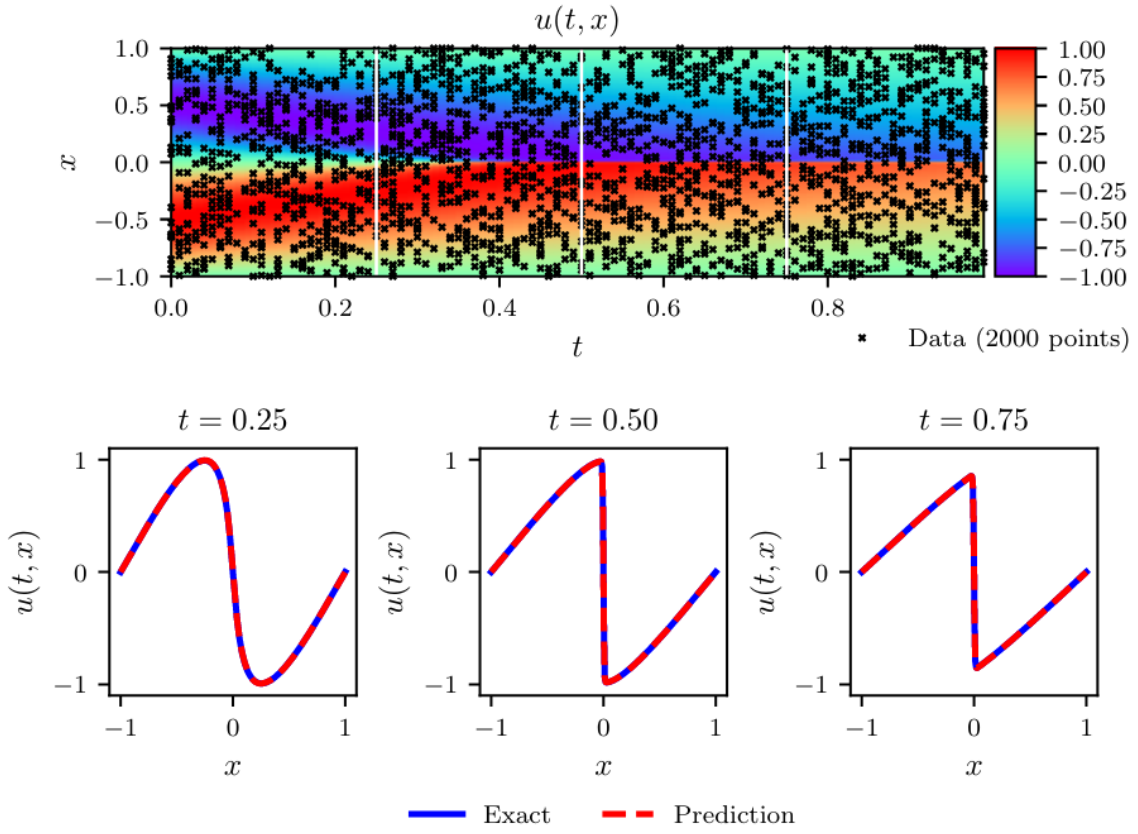
3.1.2 Gray-box

The gray-box approach is described in [18]. In that case, the general form of $\mathcal{N}$ is known, but it contains unknown parameters λ, which need to be predicted by the neural network, in addition to the solution $u(t, x)$.

As an example, Burger's equation with continuous time model will be presented. The equation has the following form:

$$u_t + \lambda_1 u u_x - \lambda_2 u_{xx} = 0 \tag{3.12}$$

The loss function is defined similarly as in (3.7) but the number of collocation points is taken the same as the number of training points.



Correct PDE	$u_t + uu_x - 0.0031831u_{xx} = 0$
Identified PDE (clean data)	$u_t + 0.99915uu_x - 0.0031794u_{xx} = 0$
Identified PDE (1% noise)	$u_t + 1.00042uu_x - 0.0032098u_{xx} = 0$

Figure 3.3: Results obtained for Burger's equation in gray-box approach with continuous time, top: predicted solution $u(t,x)$ with training points marked with black points, middle: comparison between predicted and exact solution for three snapshots marked with vertical white lines in the top picture, bottom: predicted PDEs for clean and noised data compared to the correct one [18].

In Figure 3.3 results obtained for $N_u = N_f = 2000$ and for NN with 9 layers with 20 neurons each are presented. It is clear that in this case this method works properly for data with small noise (1%). Furthermore, the authors show that predictions are reasonable even for 10% noise and that the λ_2 parameter is more vulnerable to noise than λ_1.

In the same work, the authors show that this approach is successful in predicting pressure from 2D Navier-Stokes equations without its presence in the training data. They also indicate that continuous time model suffers from the need for a large number of training points. In this case, the discrete time model works correctly for an equation with higher order derivatives as it

is shown for the Korteweg–de Vries equation, which involves a third-order derivative.

3.1.3 Black-box

Finally, paper [19] presents the most general approach in which the form of $\mathcal{N}$ remains unknown. Before training NN, one must make an assumption about $\mathcal{N}$, for example:

$$\mathcal{N}(x, u, u_x) = \alpha_{0,0} + \alpha_{1,0}u + \alpha_{0,1}u_x + \alpha_{1,1}uu_x. \tag{3.13}$$

Here, in fact, two NNs are used, one which is learning the form of the differential equation given by $\mathcal{N}$ and the other that provides the solution $u(t, x)$.

For example, for Burger's equation:

$$u_t = -uu_x + 0.1u_{xx}, \tag{3.14}$$

one of possible assumptions is that:

$$u_t = \mathcal{N}(u, u_x, u_{xx}). \tag{3.15}$$

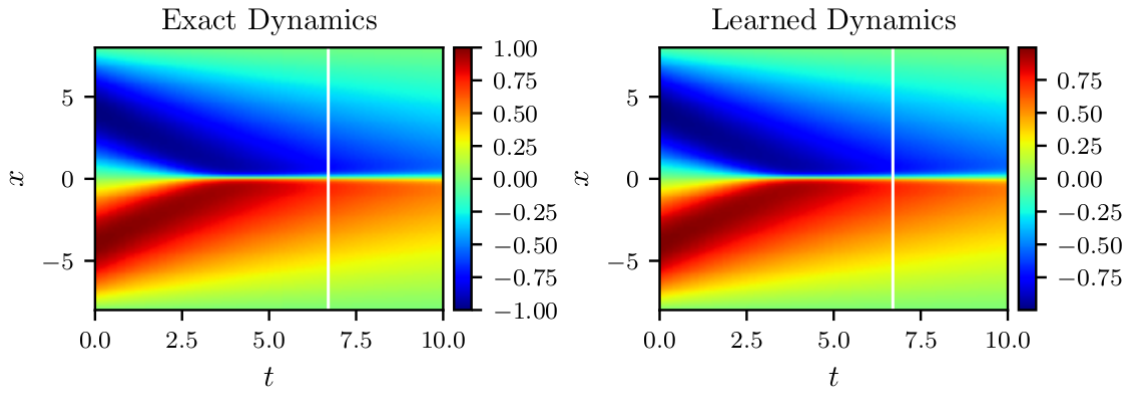


Figure 3.4: Results obtained for Burger's equation in black-box approach, white vertical line indicates the maximum time value for training points [19].

From the data points generated for the initial condition $u(0, x) = -\sin(\pi x/8)$, $x \in [-8, 8]$, for time values between 0 and 6.7, PINN containing 5 layers with 50 neurons each has been trained. The results are shown in Figure 3.4. It is clear that the predicted solution remains in acceptable accuracy with the exact one, even for the region with time greater than 6.7 from which no point was taken during the NN training. Furthermore, Figure 3.5 shows the results obtained with PINN simulating the differential equation 3.14 for the data set with different initial condition and it still is a successful approach.

Although this method had an acceptable accuracy for the cases mentioned, it is important to emphasize that the quality of the results strongly depends on the assumed form of $\mathcal{N}$. The best predictions are obtained when it takes into account only the variables and derivatives that are present in the original equation. In other cases, for example, making $\mathcal{N}$ also dependent on t

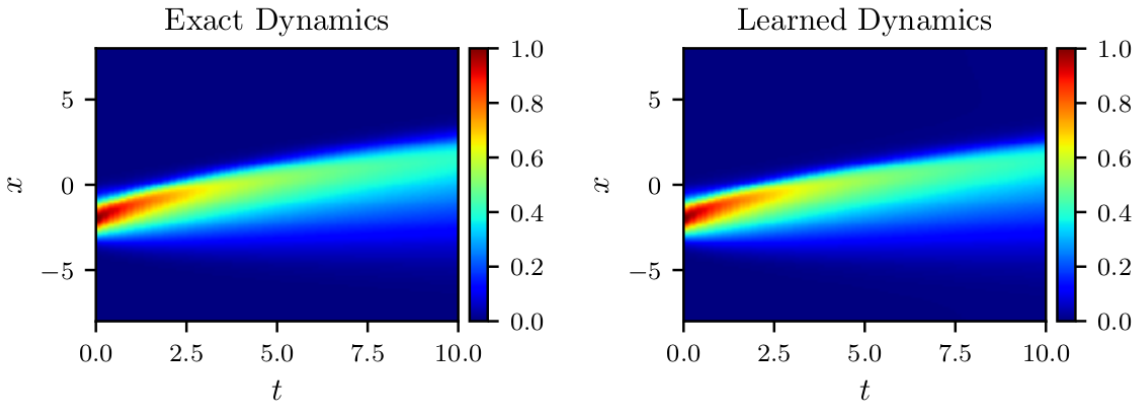


Figure 3.5: Results obtained for Burger's equation in the black-box approach, for a different initial conditions with conservation of learned form of the differential equation from an earlier case [19].

in Burger's equation, the errors are greater. Another disadvantage is related to the character of NNs - one does not get the exact form of the differential equation but only a black-box predicting values of $\mathcal{N}$ for given arguments.

3.2 Alternative approaches

3.2.1 Training without experimental or exact solution data

In work [30] the authors compare the results obtained from NN and classical numerical methods to solve Euler's equation for the Sod shock tube in the following form:

$$\frac{\partial}{\partial t}U + \frac{\partial}{\partial x}F(U) = 0, \tag{3.16}$$

where U is the state vector $U = (\rho, \rho u, E)^T$ and F is the flux vector $F = (\rho u, \rho u^2 + p, u(p + E))^T$ for the density ρ, velocity u, pressure p and energy $E = \frac{p}{\gamma - 1} + \frac{\rho}{2}u^2$ for the adiabatic index γ equal to $\frac{5}{3}$ in this case. Discontinuous initial condition is applied:

$$U(t = 0) = U_0, \tag{3.17}$$

where:

$$U_0 = \begin{cases} U_l, & x < 0, \\ U_r, & x \geqslant 0, \end{cases} \qquad U_l \neq U_r. \tag{3.18}$$

Using NN for this problem appeared to be successful, as shown in the top part of Figure 3.6 that presents the obtained and exact density. However, there are some differences between the predicted and exact solutions, mostly in discontinuous regions. It has been corrected by repeating the NN training, but for the equation with added diffusion term parameterized by

friction coefficient τ. For this case, the results are better and are presented in the bottom part of Figure 3.6. It is interesting that the authors did not use here exact solution values to train NN, but started from randomly generated values that fit during training to satisfy the differential equation.

The authors propose some improvements to this method: dissipative annealing, which allows us to shorten the NN training time, simultaneous parameter scans to have at once solutions for more than one configuration of the adiabatic coefficient, and residual NNs, which, however, were not too successful and this proposal needs to be further investigated. Another possible modification is to use exact or experimental data as in Raissi's et al. works [17, 18, 19].

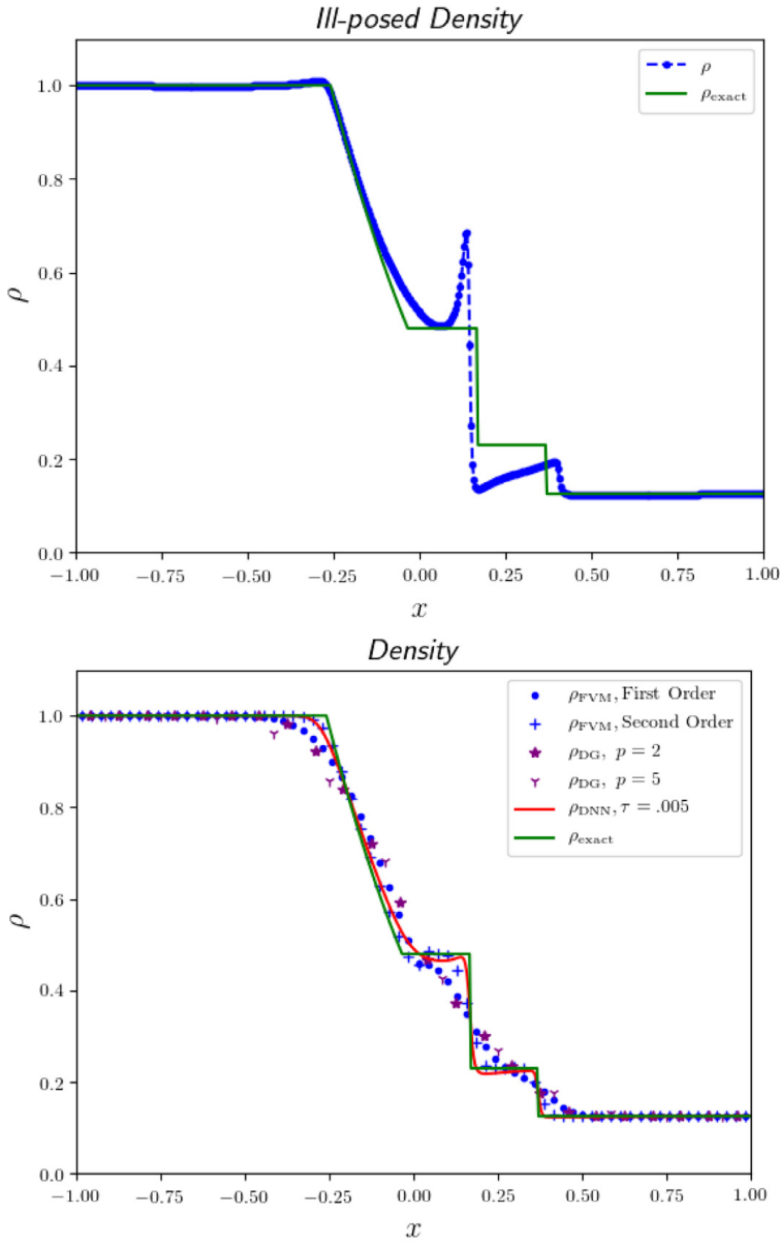


Figure 3.6: Results obtained for Euler equation: without (top) and with (bottom) adding diffusion term in the equation. Bottom picture also shows comparison of NN approach with classical numerical methods for solving differential equations [30].

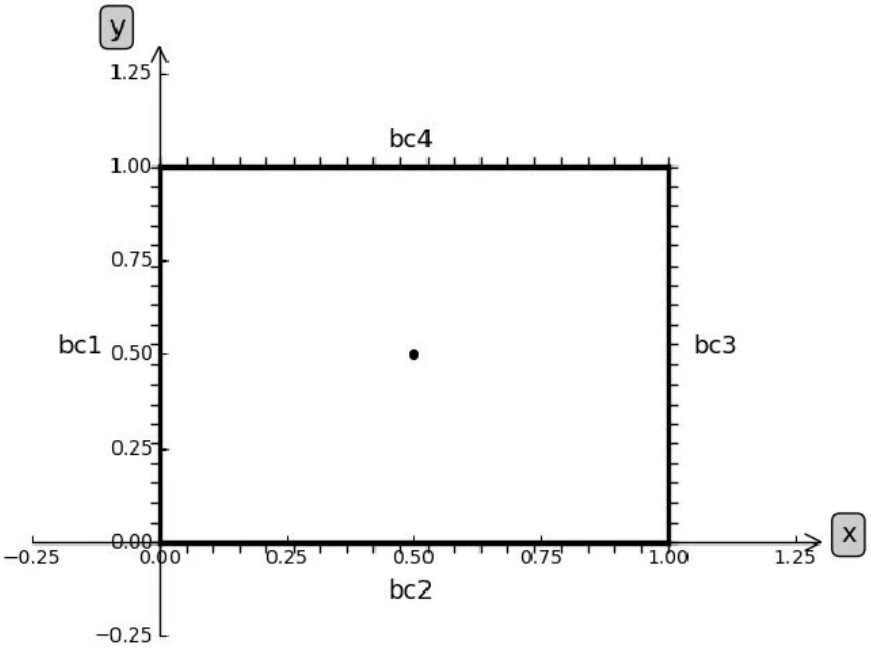


Figure 3.7: Example domain for defining weights in the DFS-net approach [31]

Another extension of the original PINN idea is presented in [31]. Here, the authors observe that standard PINN models tend to create unstable predictions in different subdomains. They propose to use weights for training points, defined, for example, in the domain from Figure 3.7 for points inside the boundaries as follows:

$$w(p) = \begin{cases} \frac{1}{0.5-d_t} d_l, & \text{if } d_l > 0.5 - d_t, \\ 1, & \text{otherwise}, \end{cases} \tag{3.19}$$
$$p = (x, y),$$

where d_t is an empirical parameter and d_l denotes the shortest distance between a given point p and one of the boundaries. The graphical representation of weights for d_t equal to 0.1 is presented in Figure 3.8. The weights for the boundary points are defined differently:

$$w(p) = \begin{cases} 1 - y^2, & p \in \{bc1, bc3\}, \\ 1 - x^2, & p \in \{bc2, bc4\}, \end{cases} \tag{3.20}$$

and are presented in Figure 3.9.

These weights are then used in the training process as multipliers in the standard loss function defined in 3.7. This model is called the data-free surrogate model (DFS-net). One of the equations solved successfully by DFS-net is the Helmholtz equation in the following form:

$$\Delta u(x, y) + k^2 u(x, y) = q(x, y). \tag{3.21}$$

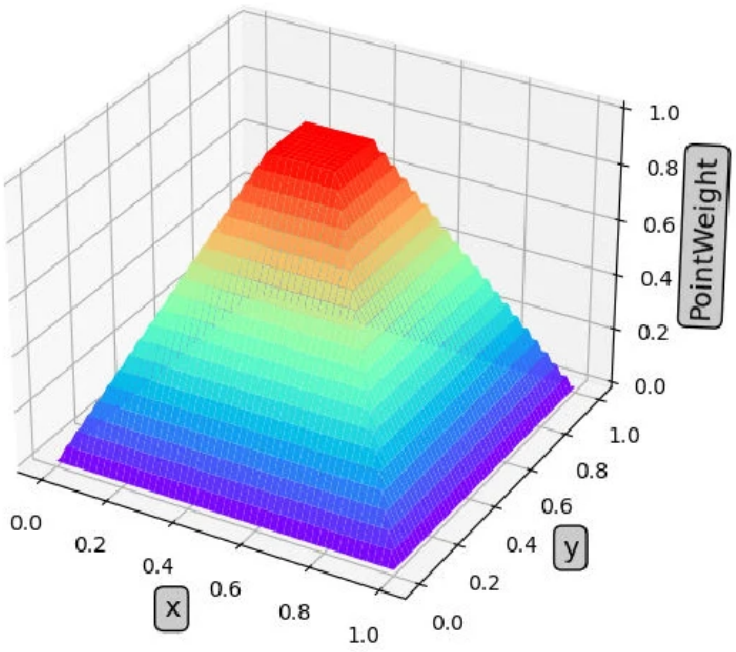


Figure 3.8: Weights for inner points with d_t equal 0.1 [31]

The problem is solved in the domain $\Omega = [0, 1] \times [0, 1]$ with $q(x, y)$ defined as follows:

$$q(x, y) = \sin(\alpha_1 \pi x) \sin(\alpha_2 \pi y) \left(k^2 - (\alpha_1 \pi)^2 - (\alpha_2 \pi)^2\right). \tag{3.22}$$

The results are presented in Figure 3.10. Similar concept with a mathematical proof of correctness of this approach is presented in work [32].

3.2.2 DeepONet

One of the well-known applications of neural networks is the approximation of continuous functions, as was described in previous examples. In a recent work [33], Lu Lu et al. show results for a successful accurate approximation of the nonlinear continuous operator by a neural network with a single hidden layer, called DeepONet. It consists of two subnetworks: a branch network used to encode the input function at a fixed number of points and a trunk network used to encode the locations of points for the output functions. Schematically, it is presented in Figure 3.11.

3.2.3 Hybrid model

Buchaniec et. al. [34] propose an approach in which only a part of the solved equation is modeled by a neural network while the rest is explicitly taken into account through a mathematical model, Figure 3.12 shows the scheme of the Integrated Mathematical Modeling - Artificial Neural Network (IMANN) method.

The authors describe an example of modeling polynomial functions:

$$f(x) = \frac{x^5 - 16x^3 + 5x^2}{2}. \tag{3.23}$$

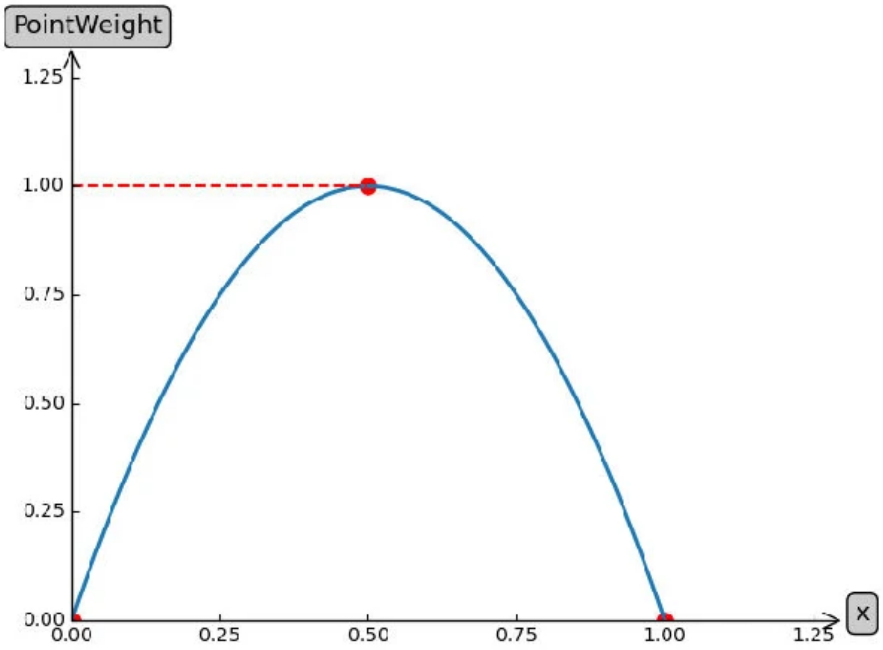


Figure 3.9: Weights for boundary points [31]

In this method, some parts of the function f are substituted with functions $a(x)$ or $b(x)$ that have been modeled by NN. The function f has one of the following forms:

$$\begin{aligned} f_1 &= \frac{a(x)x^5 - 16x^3 + 5x^2}{2}, \\ f_2 &= \frac{a(x)x^4 - 16x^3 + 5x^2}{2}, \\ f_3 &= \frac{a(x)x^3 - 16x^3 + 5x^2}{2}, \\ f_4 &= \frac{a(x)x^5 - b(x)x^3 + 5x^2}{2}. \end{aligned} \tag{3.24}$$

The results of this work show that the best precision is obtained when the degree of functions $a(x)$ and $b(x)$ is low, that is, when they are constant or linear functions. This approach is successful for rather small learning datasets of no more than 30 data points.

3.2.4 Other works

Before using PINNs, Raissi et al. tried to incorporate the Gaussian process into solving differential equations [35, 36, 37, 38], but this approach was only successful for linear equations and suffered from high computational complexity. There was also an attempt to use the Bayesian approach to this problem [39].

There was a postulate to create libraries of known solutions to differential equations to have reasonable data sets for NN training [40].

In [41] authors use a series of NNs to achieve consecutive approximations of the desired solution related to financial problems. In other work [42], Nourani and Mousavi used NN as an

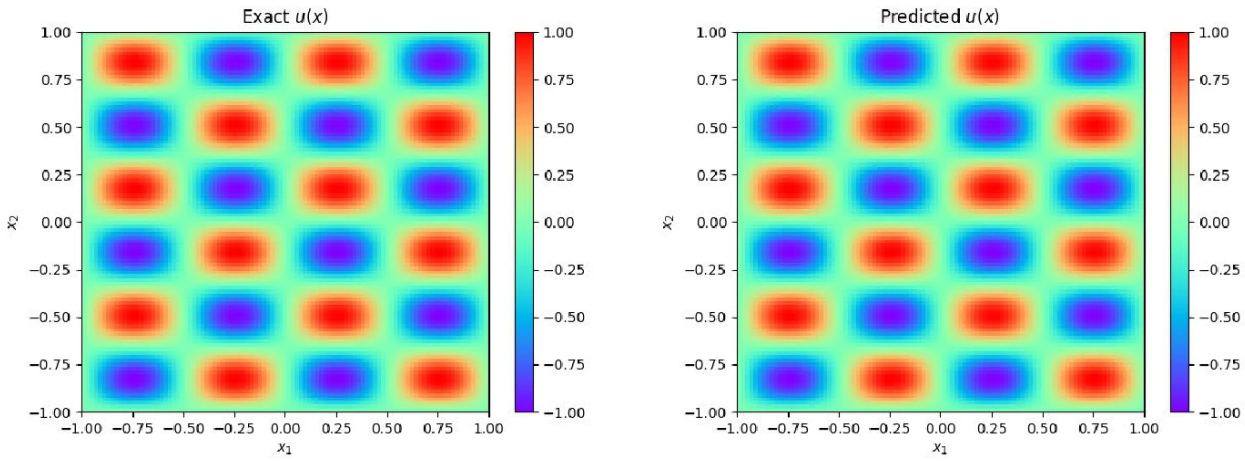


Figure 3.10: Solution for Helmholtz equation with $\alpha_1 = 1$ and $\alpha_2 = 4$ [31]

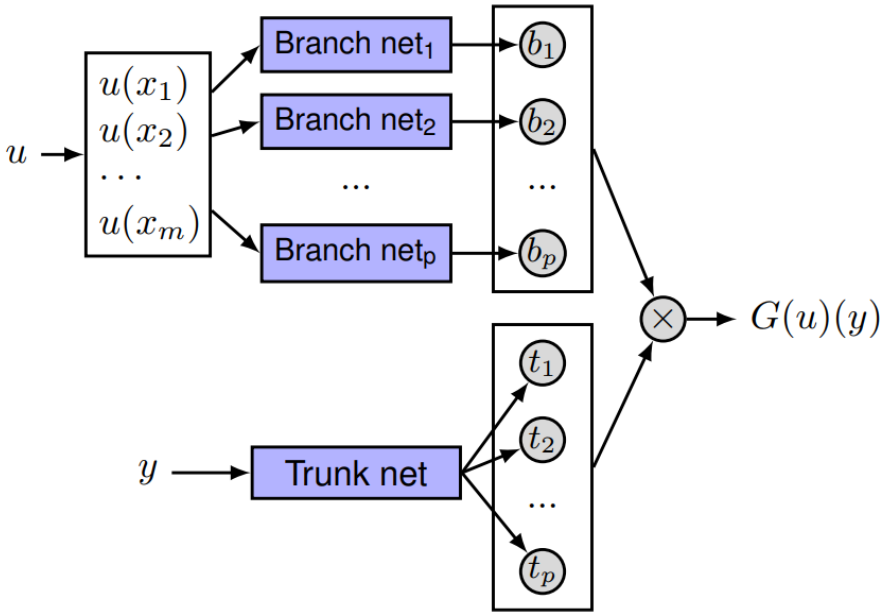


Figure 3.11: Scheme of DeepONet [33]

intermediate step. The neural network was trained on known values of the solution at $t = 0$, and then its predictions were used as an input to another method of solving PDEs.

Attempts have also been made to use NNs to solve differential equations other than PDEs. In [43, 44] a method to solve fractional order differential equations is described. The method works correctly between training points, however, it is unsuccessful in extrapolating the solution. The fuzzy differential equations have been solved in a hybrid way [45] - one part of the model satisfied the initial conditions, while NN solved the equation to obtain consecutive time steps. In this case there was also a problem with extrapolation of the predicted solution outside the training points. Neural networks were also successful in solving integral equations such as the Fredholm equation [46].

In the work [47] the authors investigated the structure of the trained NN to solve the Poisson equation and found that the first and second layers of the NN are general, while further layers

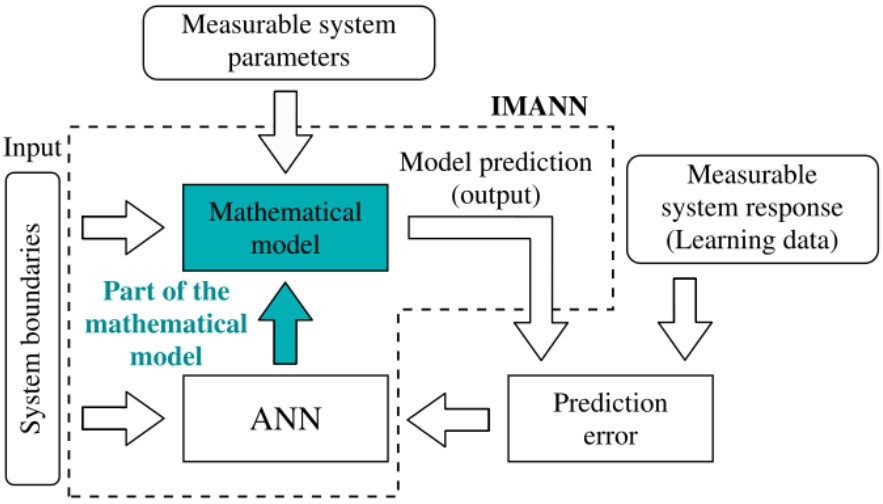


Figure 3.12: Scheme of IMANN method [34].

are more specific to a given problem.

The authors of the work [48] generalize Raissi's white-box approach and approximate unknown parts of the model with NNs. It is similar to gray- and black-box approaches.

In other work [49], the authors developed an improvement of the back-propagation algorithm to work with complex domain geometries such as unusual polygons.

Some problems with chaotic systems are described in work [50], where the authors successfully use NNs to solve Burger's equation in different initial conditions: rarefaction, shock and smooth.

To make PINNs a more powerful tool in comparison to classical numerical methods, some approaches involving transfer learning and the integration of PINNs into traditional linear solvers are discussed in the article [51].

3.3 Libraries for solving PDEs with PINNs

For Julia language, there exists one library using PINNs – DiffEqFlux [52]. It allows a user to apply NNs in solving differential equations, but its usage is limited only to basic cases.

Currently, there are libraries available for Python to use NNs to solve differential equations. Elvet library [53] gives the possibility to define custom boundary conditions and also solve variational problems, but is limited to only the white-box approach. NeuroDiffEq [54] uses a specific method to satisfy boundary conditions, which results in a function u_r modeled via a neural network and a factor $A(t, x; t_0, x_b)$:

$$\tilde{u}(t, x) = A(t, x; t_0, x_b) u_r(t, x), \tag{3.25}$$

where A is chosen to satisfy:

$$\tilde{u}(t_0, x_b) = u_0. \tag{3.26}$$

DeepXDE library [55] is based on Raissi's works [17, 18, 19] and is capable of solving PDEs in white- and gray-box approaches. During training it uses boundary conditions of different kinds

(especially Dirichlet and Neumann) in the loss function. It applies full implicit Runge-Kutta method, does need time step, is mesh-free and has no problems with stability. However, as all machine learning methods it often fails to generalize results beyond the training domain. Due to its capabilities and convenience in use, this one is used in this work for solving Schrödinger equation for standard model systems.

There are libraries available to perform molecular dynamics simulations enhanced by NNs, among them Schnetpack [56, 57] and PND [58], where neural networks are usually trained on forces and energies. Schnetpack uses Convolutional Neural Networks (CNNs) in a special architecture developed by its authors, named SchNet. In Schnetpack, it is possible to train NN to predict atomic charges, so it is used in this work for replacement of charge prediction in the standard DFTB method. Training is performed with loss function involving true data, therefore this is an example of supervised learning.

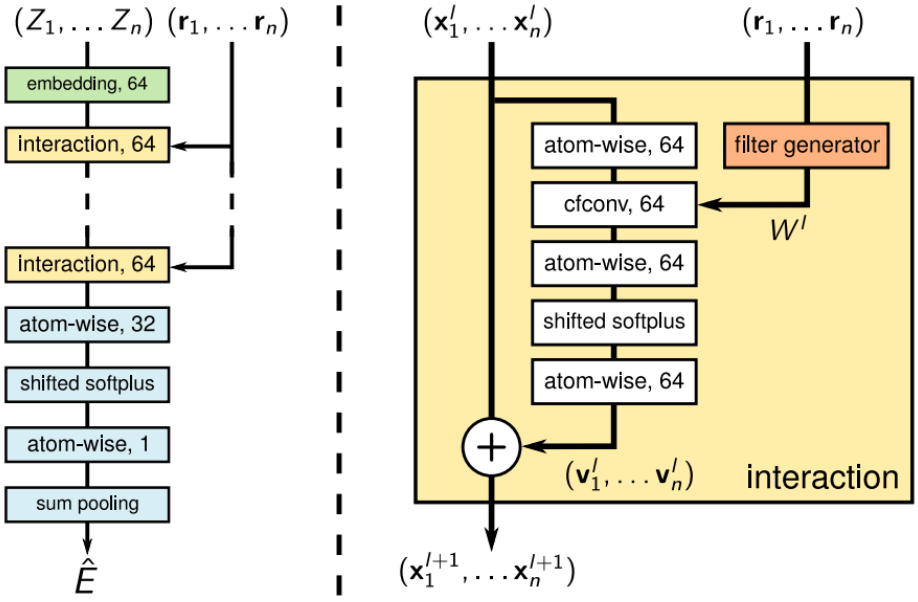


Figure 3.13: Typical architecture of the NN used by Schnetpack [56], where Z_i are atomic charges, $\mathbf{r}_i$ are atomic positions and $\mathbf{x}_i^l$ are the outputs of the l-th layer of the NN (and inputs to the (l+1)-th layer). $\hat{E}$ represents the predicted property, here it is the system energy. For each layer, the number of neurons is given.

For readability in this section vectors are marked in bold font (not arrows due to the usage of upper indices). A system containing n atoms can be uniquely described by their nuclear charges $Z = (Z_1, ..., Z_n)$ and positions $R = (\mathbf{r}_1, ..., \mathbf{r}_n)$. In SchNet architecture, there are defined specific layers, utilizing these values [56]:

- atom embeddings – these layers transform the nuclear charges Z to a tuple of features used internally in the NN, at the l-th layer: $X^l = (\mathbf{x}_1^l, ..., \mathbf{x}_n^1)$ where $\mathbf{x}_i^l \in \mathbb{R}^F$ with the number of features F defined by the user. The representation $\mathbf{x}_i^0$ of atom i is randomly initialized with a value a_{Z_i} depending on the atom type Z_i and further optimized during training
- atomwise layers – dense layers which apply transformations separately to representations

$\mathbf{x}_i^l$, with weights W^l and biases $\mathbf{b}^l$ independent of the atom type:

$$\mathbf{x}_i^{l+1} = W^l \mathbf{x}_i^l + \mathbf{b}^l. \tag{3.27}$$

- interaction blocks – these fragments of the SchNet include interactions between atoms (they take atom positions as an input), presented on the right-hand side of Figure 3.13. While atoms are not located on the regular grid like image pixels, a generalization of discrete convolutional layers is used, named continuous-filter convolutional layers (cfconv). These layers use the output of the previous layer $\mathbf{x}_i^l$ as well as the generated filter $W^l(\mathbf{r}_j - \mathbf{r}_i)$ to produce the output $\mathbf{x}_i^{l+1}$:

$$\mathbf{x}_i^{l+1} = \sum_{j=0}^{n} \mathbf{x}_j^l \circ W^l(\mathbf{r}_j - \mathbf{r}_i), \tag{3.28}$$

  where $\circ$ denotes the element-wise multiplication. Furthermore, this layer uses shifted softplus activation function in the form:

$$\mathrm{ssp}(x) = \ln\left(0.5e^x + 0.5\right). \tag{3.29}$$

- filter-generating networks – fully-connected NN taking the vector pointing from atom i to its neighbor j as an input to produce the filter values $W(\mathbf{r}_j - \mathbf{r}_i)$. It incorporates Gaussian functions to represent atomic distances.

To limit the computational effort during training the SchNet model, the additional parameter named *cutoff* is defined: it is the biggest distance between atoms at which they are treated as interacting, atoms more distant than the *cutoff* are considered as non-interacting.

The prediction of the desired property of the system is finally done by the atomwise layers giving atomwise contributions. For intensive properties (independent of the system size) the final prediction is calculated as an average of atomwise contributions, while for extensive properties (dependent on the system size - additive for subsystems) it is calculated as the sum of atomwise contributions.

Schematically, the SchNet architecture is presented in Figure 3.13. Briefly, the process of generating the output could be divided into a few steps: atom embedding for extracting internal features (marked in green in Figure 3.13), calculation of interactions (marked in yellow) and final prediction of the atom-wise features and their summing or averaging (layers marked in blue).

3.4 Open problems

From the works reviewed in this chapter, some open problems appear. It is still unclear how to estimate the maximal possible error of NN prediction [17]. Another issue is development of rules for selecting training points to obtain the best accuracy. Convolutional NNs have not yet been used as PINNs and seem to be a promising improvement [19]. The use of residual NNs also needs further investigation.

The development of similar methods for other types of differential equations (e.g., stochastic DEs, fractional PDEs) is a possible next step in the field of PINNs. Furthermore, it would be useful to provide a general method for equations the form of which depends on auxiliary conditions, for example, flux equations depending on the Reynolds number [19].

Chapter 4

Standard quantum chemistry problems

This chapter provides introduction into quantum mechanics, theoretical description and formulation with PDEs of four model problems: particle in a box, harmonic oscillator, rigid rotor and hydrogen atom. The analytical solutions of these models are known and can be used to verify the solutions obtained in this thesis (Chapter 6) with PINNs. Next, the DFTB method used to solve the Schrödinger equation for multielectron systems is explained. In the end, the method of obtaining simulations of atomic motions is described: molecular dynamics (MD) along with basics of infrared (IR) spectroscopy.

4.1 Basics of quantum mechanics

Quantum mechanics is the theory developed in XX century for description of the physics at the microscale. It is usually formulated using a set of axioms called postulates. Here, they will be presented as in [28].

4.1.1 State representation

Quantum mechanical state of the system at the moment t with n particles is described by the wavefunction ψ which is a vector in the Hilbert space. Here, for convenience its position representation $\psi(\vec{r}_1, ..., \vec{r}_n, t)$ will be used.

Only the square of modulus of the wavefunction has physical interpretation. It is the probability density $\rho(\vec{r}_1, ..., \vec{r}_n, t)$ of finding the particles in positions $(\vec{r}_1, ..., \vec{r}_n)$ at the time t:

$$\rho(\vec{r}_1, ..., \vec{r}_n, t) = |\psi(\vec{r}_1, ..., \vec{r}_n, t)|^2 . \tag{4.1}$$

Due to the connection with probability, the wavefunction should be normalized:

$$\int_{\mathbb{R}^n} |\psi(\vec{r}_1, ..., \vec{r}_n, t)|^2 \, \mathrm{d}^3\vec{r}_1 ... \mathrm{d}^3\vec{r}_n = 1. \tag{4.2}$$

To properly describe the physical state, the wavefunction should also be bounded (probability cannot be infinite), continuous and unambiguous.

If $\psi_1, ..., \psi_m$ are the allowed wavefunctions of the given system, then also their linear combination ψ is the possible state of the system:

$$\psi = \sum_{i=1}^{m} c_i \psi_i, \tag{4.3}$$

with the probability P_j of observing the state ψ_j equal:

$$P_j = |c_j|^2, \tag{4.4}$$

what leads to another limitation:

$$\sum_{i=1}^{m} P_i = 1. \tag{4.5}$$

Subjective labelling of the identical particles in the system should not change the value of the probability density, for example with swap between particle 1 and particle 3:

$$\rho(\vec{r}_1, \vec{r}_2, \vec{r}_3, ..., \vec{r}_n) = \rho(\vec{r}_3, \vec{r}_2, \vec{r}_1, ..., \vec{r}_n), \tag{4.6}$$

thus:

$$\psi(\vec{r}_1, \vec{r}_2, \vec{r}_3, ..., \vec{r}_n) = \pm\psi(\vec{r}_3, \vec{r}_2, \vec{r}_1, ..., \vec{r}_n). \tag{4.7}$$

Depending on the sign on the right hand side of the above equation, the particles can be divided into two groups: bozons (+ sign) and fermions (− sign).

4.1.2 Physical properties

Every mechanical physical property A is represented by a hermitian and linear operator $\hat{A}$. In the position representation, the basic operators for positions $\hat{x}_i$ and momenta $\hat{p}_i$ have the following forms:

$$\hat{x}_i = x_i, \tag{4.8}$$

$$\hat{p}_i = -i\hbar\nabla. \tag{4.9}$$

In a single measurement of the physical property A only one of the eigenvalues of the operator $\hat{A}$ can be observed. Since these operators are hermitian, it is possible to find a complete set of their eigenfunctions, i.e. each wavefunction can be described as a linear combination of these eigenfunctions.

The average value of $\langle A \rangle$ after many measurements in state ψ is expressed as:

$$\langle A \rangle = \int_{\mathbb{R}^n} \psi^*(\vec{r}_1, ..., \vec{r}_n, t) \hat{A} \psi(\vec{r}_1, ..., \vec{r}_n, t) \mathrm{d}^3\vec{r}_1 ... \mathrm{d}^3\vec{r}_n. \tag{4.10}$$

4.1.3 System dynamics

The time evolution of quantum mechanical state is described by the time-dependent Schrödinger equation:

$$\hat{H}\Psi(\vec{r},t) = i\hbar\frac{\partial\Psi(\vec{r},t)}{\partial t}, \tag{4.11}$$

where $\hat{H}$ is the operator of the system total energy, the Hamiltonian:

$$\hat{H} = \frac{\hat{p}^2}{2m} + V(\vec{r},t) = -\frac{\hbar^2}{2m}\Delta + V(\vec{r},t). \tag{4.12}$$

If the potential $V(x,t)$ is time independent and the wavefunction is described in the form:

$$\Psi(\vec{r},t) = A(t)\psi(\vec{r}), \tag{4.13}$$

equation 4.13 can be rewritten in the following form:

$$i\hbar\frac{1}{A(t)}\frac{\partial A(t)}{\partial t} = \frac{1}{\psi(\vec{r})}\left[-\frac{\hbar^2}{2m}\Delta + V(\vec{r})\right]\psi(\vec{r}). \tag{4.14}$$

The left-hand side contains only functions of time, while the right-hand side is only position dependent; this equality enforces that both sides are constant and equal E and thus the equation can be separated:

$$i\hbar\frac{\partial A(t)}{\partial t} = EA(t), \tag{4.15}$$

$$\left[-\frac{\hbar^2}{2m}\Delta + V(\vec{r})\right]\psi(\vec{r}) = E\psi(\vec{r}). \tag{4.16}$$

Equation 4.16 is called the time-independent Schrödinger equation. The model systems (particle in a box, harmonic oscillator, rigid rotor and hydrogen atom) are described with this time-independent form.

4.2 Particle in a box

This section presents solutions for different variations of the so-called particle in a box problem.

4.2.1 One-dimensional infinite box

The particle in an infinite box is a system in which the potential is discontinuous and has the following form [28]:

$$V(x) = \begin{cases} 0, & -\frac{L}{2} < x < \frac{L}{2}, \\ \infty, & \text{for other } x. \end{cases} \tag{4.17}$$

For this case, the Schrödinger equation has the following form:

$$-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2\psi_n(x)}{\mathrm{d}x^2} = E\psi_n(x). \tag{4.18}$$

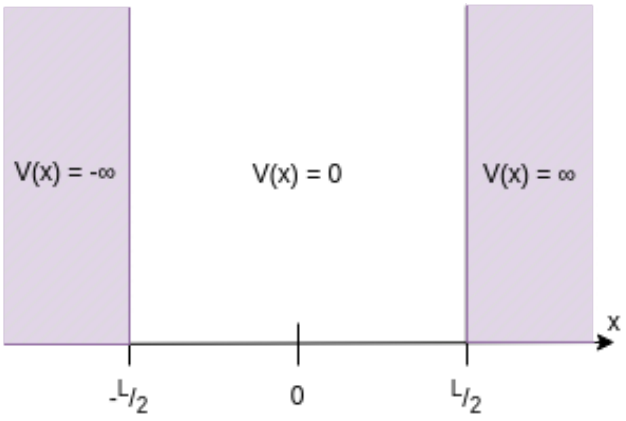


Figure 4.1: Infinite potential box - potential dependence on position.

With the assumption of disappearing of a wavefunction on the box borders:

$$\psi\left(-\frac{L}{2}\right) = \psi\left(\frac{L}{2}\right) = 0, \tag{4.19}$$

the solution has the following form:

$$\psi_n(x) = \begin{cases} \sqrt{\frac{2}{L}} \sin\left(k_n\left(x + \frac{L}{2}\right)\right), & -\frac{L}{2} < x < \frac{L}{2}, \\ 0 \quad \text{for other } x \end{cases}, \tag{4.20}$$

with k_n given as:

$$k_n = \frac{n\pi}{L}, \quad n = 1, 2, 3, ... \tag{4.21}$$

As could be seen, the allowed solutions (having physical meaning) are quantized by consecutive positive natural numbers.

The energy is dependent on the value of k_n, thus not all values are allowed:

$$E_n = \frac{\hbar^2 k_n^2}{2m} = \frac{\hbar^2 n^2 \pi^2}{2mL^2}, \quad n = 1, 2, 3, ... \tag{4.22}$$

where m stands for the mass of a particle.

This model problem, despite being simple, has found some applications, e.g. to describe beta-caroten electron states [59].

4.2.2 One-dimensional triangular box

Particle in a triangular box is a special case of this problem, where one side of the box is limited by an infinite potential while the other side has constantly growing finite value potential:

$$V(x) = \begin{cases} \infty, & x < 0, \\ q\varepsilon x, & x \geqslant 0. \end{cases} . \tag{4.23}$$

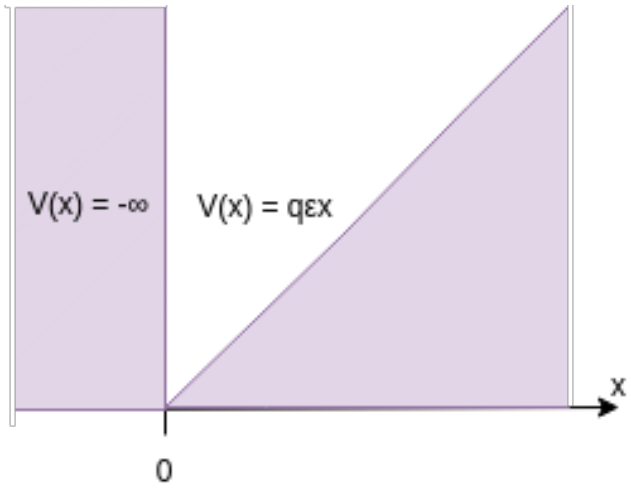


Figure 4.2: Triangular potential box - potential dependence on position.

The physical system which could be described by this model is a particle with charge q moving in a constant electric field ε, e.g. between plates of the capacitor.

For this case, the Schrödinger equation has the following form:

$$-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2\psi_n(x)}{\mathrm{d}x^2} + q\varepsilon x\psi_n(x) = E_n\psi_n(x). \tag{4.24}$$

After a few manipulations and an introduction of a new variable z:

$$z = \beta\left(x - \frac{E_n}{q\varepsilon}\right), \tag{4.25}$$

where β is a normalization factor:

$$\beta = \sqrt[3]{\frac{2mq\varepsilon}{\hbar^2}}, \tag{4.26}$$

the equation takes the following form:

$$\frac{\mathrm{d}^2\psi_n(z)}{\mathrm{d}z^2} = \psi_n(z)z, \tag{4.27}$$

which is known as the Airy equation. Its general solution has the following form:

$$\psi_n(z) = A\mathrm{Ai}(z) + B\mathrm{Bi}(z), \tag{4.28}$$

with A and B as constants, and $\mathrm{Ai}(z)$ and $\mathrm{Bi}(z)$ as the Airy's functions of the first and second kind, respectively. Since Bi has no finite limit at infinity, B value is forced to be equal to 0.

Boundary value $\psi_n(0) = 0$ leads to quantized values of E_n:

$$E_n = -\alpha_n q\varepsilon\sqrt[3]{\frac{\hbar^2}{2mq\varepsilon}}, \tag{4.29}$$

where α_n is the n-th zero of the Airy's function of the first kind, Ai.

From the normalization constraint:

$$\int_0^{+\infty} |\psi_n(x)|^2 = 1, \tag{4.30}$$

the constant value A is determined as:

$$A = \left(\frac{\sqrt[3]{\frac{2mq\varepsilon}{\hbar^2}}}{\mathrm{Ai}'^2\left(-\sqrt[3]{\frac{2mq\varepsilon}{\hbar^2}}\frac{E_n}{q\varepsilon}\right)} \right)^{\frac{1}{2}}, \tag{4.31}$$

and finally:

$$\psi_n(x) = \left(\frac{\sqrt[3]{\frac{2mq\varepsilon}{\hbar^2}}}{\mathrm{Ai}'^2\left(-\sqrt[3]{\frac{2mq\varepsilon}{\hbar^2}}\frac{E_n}{q\varepsilon}\right)} \right)^{\frac{1}{2}} \mathrm{Ai}\left[\sqrt[3]{\frac{2mq\varepsilon}{\hbar^2}}\left(x - \frac{E_n}{q\varepsilon}\right)\right]. \tag{4.32}$$

4.2.3 Two-dimensional infinite rectangular box

In this case, the domain Ω if defined as follows:

$$\Omega = \left\{ (x,y) : |x| < \frac{L_x}{2} \wedge |y| < \frac{L_y}{2} \right\}, \tag{4.33}$$

with the potential $V(x,y)$ defined as follows:

$$V(x,y) = \begin{cases} 0, & (x,y) \in \Omega, \\ \infty, & \text{for other } (x,y). \end{cases} \tag{4.34}$$

The Schrödinger equation in this case takes the following form:

$$-\frac{\hbar^2}{2m}\Delta\Psi(x,y) = E\Psi(x,y). \tag{4.35}$$

The assumption that $\Psi(x,y)$ has the product form $\Psi(x,y) = \psi_{n_x}(x)\phi_{n_y}(y)$ leads to separation of the PDE into two one-dimensional differential equations, which are solved in subsection 4.2.1. Thus, the solution has the following form:

$$\Psi_{n_x n_y}(x,y) = \begin{cases} \frac{2}{\sqrt{L_x L_y}} \sin\left(k_x\left(x + \frac{L_x}{2}\right)\right) \sin\left(k_y\left(y + \frac{L_y}{2}\right)\right), & (x,y) \in \Omega, \\ 0 \quad \text{for other } (x,y) \end{cases}, \tag{4.36}$$

with k_x, k_y defined as follows:

$$k_x = \frac{n_x \pi}{L_x}, \quad n_x = 1, 2, 3, ..., \tag{4.37}$$

$$k_y = \frac{n_y \pi}{L_y}, \quad n_y = 1, 2, 3, ..., \tag{4.38}$$

and energy with values quantized by two quantum numbers, n_x and n_y:

$$E_{n_x n_y} = \frac{\hbar^2 \pi^2}{2m} \left(\frac{n_x^2}{L_x^2} + \frac{n_y^2}{L_y^2} \right). \tag{4.39}$$

4.2.4 Two-dimensional infinite circular box

In this case, the domain Ω has the following form:

$$\Omega = \left\{ (x, y) : x^2 + y^2 < a^2 \right\}, \tag{4.40}$$

and the potential $V(x, y)$ again takes the form:

$$V(x, y) = \begin{cases} 0, & (x, y) \in \Omega, \\ \infty, & \text{for other } (x, y). \end{cases} \tag{4.41}$$

For convenience, the polar coordinates are used here, defined as follows:

$$\begin{cases} r = \sqrt{x^2 + y^2} \\ \theta = \text{atan2}(y, x), \end{cases} \tag{4.42}$$

and the Laplace operator takes the following form:

$$\Delta = \frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial}{\partial r} + \frac{1}{r^2} \frac{\partial^2}{\partial \theta^2}. \tag{4.43}$$

The Schrödinger equation:

$$-\frac{\hbar^2}{2m} \left(\frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial}{\partial r} + \frac{1}{r^2} \frac{\partial^2}{\partial \theta^2} \right) \psi(r, \theta) = E \psi(r, \theta), \tag{4.44}$$

after substitution of $\psi(r, \theta) = R(r)\Theta(\theta)$, separates into two equations:

$$\frac{\partial^2 \Theta(\theta)}{\partial \theta^2} = -l^2 \Theta(\theta), \tag{4.45}$$

$$\frac{\mathrm{d}^2 R(r)}{\mathrm{d}r^2} + \frac{1}{r} \frac{\mathrm{d}R(r)}{\mathrm{d}r} + \left(k^2 - \frac{l^2}{r^2} \right) R(r) = 0, \tag{4.46}$$

with $k = \frac{\sqrt{2mE}}{\hbar}$.

The normalized solution of equation 4.45 has the form:

$$\Theta_l(\theta) = \frac{1}{\sqrt{2\pi}} e^{il\theta}, \quad l = 0, \pm 1, \pm 2, ... \tag{4.47}$$

while the equation 4.46 is the Bessel equation with the general solution:

$$R_{nl}(r) = A J_l(kr) + B N_l(kr). \tag{4.48}$$

Here, the B constant is forced to be 0, because the Neumann function N_l has a singularity for r equal to 0. From the normalization condition and boundary condition $R(a) = 0$ the final form of the wavefunction is obtained:

$$\psi_{nl}(r,\theta) = \frac{1}{\sqrt{\pi}a\left|J_{l+1}\left(\alpha_{nl}\right)\right|} J_l\left(\alpha_{nl}\frac{r}{a}\right) e^{il\theta}, \tag{4.49}$$

where α_{nl} is the n-th zero of the l-th order Bessel function J_l. The energy of the system is equal:

$$E_{nl} = \frac{\hbar^2\alpha_{nl}^2}{2ma^2}. \tag{4.50}$$

4.3 Harmonic oscillator

Harmonic oscillator is a system in which the potential is given by [28]:

$$V(x) = \frac{1}{2}m\omega^2x^2, \tag{4.51}$$

where ω is the oscillator frequency. This potential is shown in Figure 4.3. The Schrödinger equation takes the following form:

$$-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2\psi(x)}{\mathrm{d}x^2} + \frac{1}{2}m\omega^2x^2\psi(x) = E\psi(x). \tag{4.52}$$

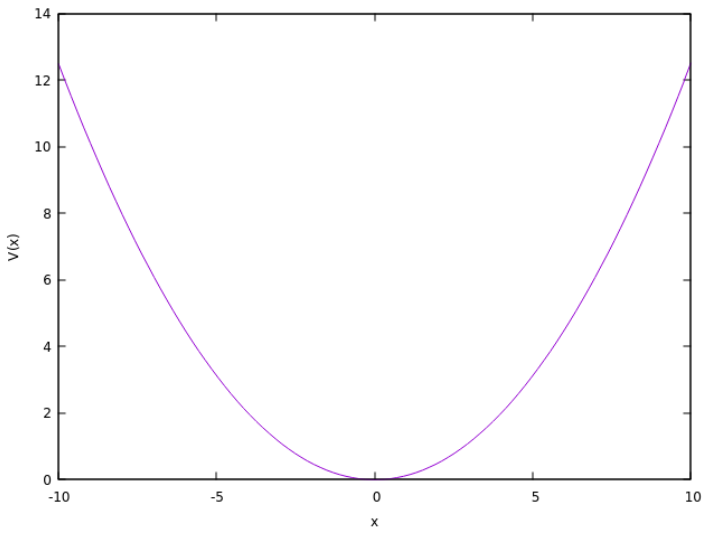


Figure 4.3: Classical potential for harmonic oscillator with frequency ω equal to 0.25. In classical mechanics points beyond the curve (determining the turning points) are not reachable

In this case, physically allowed wavefunctions are also quantized by consecutive natural numbers and have the following form:

$$\psi_n(x) = \frac{1}{\sqrt{2^n n!}}\left(\frac{m\omega}{\pi\hbar}\right)^{\frac{1}{4}} e^{-\frac{m\omega x^2}{2\hbar}} H_n\left(\sqrt{\frac{m\omega}{\hbar}}x\right), \quad n = 0,1,2,..., \tag{4.53}$$

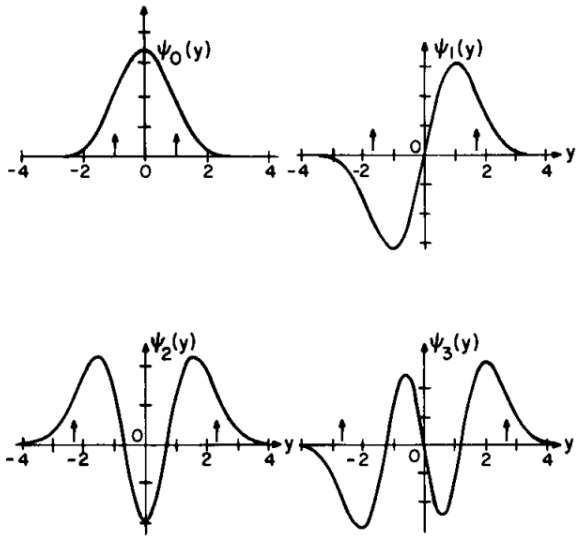


Figure 4.4: First four eigenfunctions of quantum harmonic oscillator [60]. Arrows indicate classical turning points.

where H_n denotes the Hermit polynomials:

$$H_n(x) = (-1)^n e^{x^2} \frac{\mathrm{d}^n}{\mathrm{d}x^n} \left(e^{-x^2} \right). \tag{4.54}$$

Energy is quantized and equals:

$$E_n = \hbar\omega \left(n + \frac{1}{2} \right), \quad n = 0, 1, 2, \ldots . \tag{4.55}$$

The first four eigenstates are shown in Figure 4.4. It is worth mentioning that the quantum oscillator has a nonzero probability of reaching points beyond classical turning points, which does not happen in the classical case.

This model system has been found useful in describing lattice vibrations in solid states (phonons) and as a first approximation of vibrations in molecules (however, here it needs an anharmonic correction).

4.4 Rigid rotor

A rigid rotor is a system consisting of two point masses m_1 and m_2 at positions $\vec{r}_1$ and $\vec{r}_2$ with a constant distance r between them: $r = |\vec{r}_1 - \vec{r}_2| = \text{const}$. For relative movement of these masses the Hamiltonian is defined as:

$$\hat{H} = \frac{\hat{L}^2}{2I}, \tag{4.56}$$

where I is the moment of inertia and $\hat{L}$ is the angular momentum operator, the square of which is defined in the spherical coordinate system (r, θ, ϕ) as follows:

$$\hat{L}^2(\vartheta, \varphi) = -\hbar^2 \left[\frac{1}{\sin\vartheta} \frac{\partial}{\partial\vartheta} \left(\sin\vartheta \frac{\partial}{\partial\vartheta} \right) + \frac{1}{\sin^2\vartheta} \frac{\partial^2}{\partial\varphi^2} \right]. \tag{4.57}$$

After factorization of the wavefunction $\psi(\vartheta, \varphi)$ into two parts:

$$\psi(\vartheta, \varphi) = \Theta(\vartheta)\Phi(\varphi), \tag{4.58}$$

and substitution of this form into the Schrödinger equation, after some manipulations the equation can be expressed in a separable form:

$$\frac{\sin\vartheta}{\Theta(\vartheta)}\frac{\partial}{\partial\vartheta}\left(\sin\vartheta\frac{\partial\Theta(\vartheta)}{\partial\vartheta}\right) + \frac{2IE}{\hbar^2}\sin^2\vartheta = -\frac{1}{\Phi(\varphi)}\frac{\partial^2\Phi(\varphi)}{\partial\varphi^2} = \text{const.} \tag{4.59}$$

Solutions are parameterised by two quantum numbers m and l and have the following forms:

$$\Phi_m(\varphi) = \frac{1}{\sqrt{2\pi}}e^{im\varphi}, \tag{4.60}$$

$$\Theta_{lm}(\vartheta) = N_{lm}P_l^{|m|}(\cos\varphi), \tag{4.61}$$

where N_{lm} is the normalization constant:

$$N_{lm} = (-1)^m\sqrt{\frac{2l+1}{2}\frac{(l-m)!}{(l+m)!}}, \tag{4.62}$$

and $P_l^{|m|}$ are associated Legendre polynomials:

$$P_l(x) = \frac{1}{2^l l!}\frac{\mathrm{d}^l}{\mathrm{d}x^l}\left(x^2-1\right)^l, \tag{4.63}$$

$$P_l^{|m|}(x) = (-1)^{|m|}\left(1-x^2\right)^{\frac{|m|}{2}}\frac{\mathrm{d}^{|m|}P_l(x)}{\mathrm{d}x^{|m|}}. \tag{4.64}$$

The energy of relative movement is equal:

$$E_l = \frac{l(l+1)}{2I}\hbar^2. \tag{4.65}$$

Values of quantum numbers are constrained as follows:

$$m = -l, -l+1, ..., l-1, l, \tag{4.66}$$

$$l = 0, 1, 2, \dots . \tag{4.67}$$

The wavefunctions of rigid rotor are called spherical harmonics and are denoted as $Y_l^m(\vartheta, \varphi)$.

4.5 Hydrogen atom

Hydrogen is the lightest of chemical elements and its most abundant isotope consists of 1 electron and 1 proton. Furthermore, a hydrogen-like atom is an atom with any number of protons in the atomic nuclei (here denoted by Z) and only 1 electron. For simplicity, in quantum chemistry

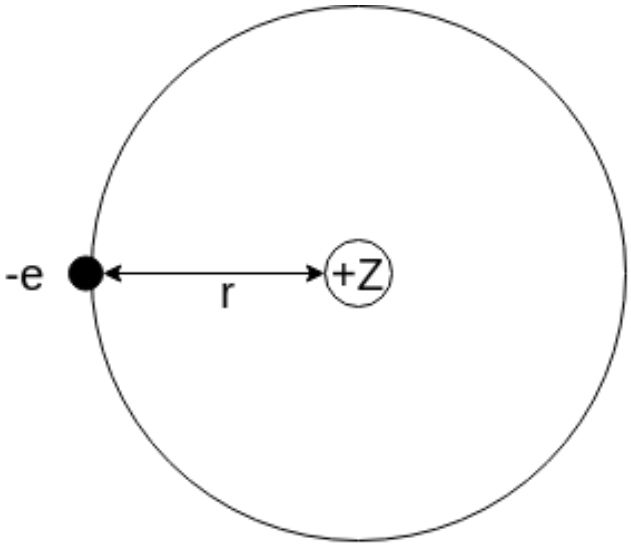


Figure 4.5: Hydrogen-like atom

Hartree atomic units are used, with basic values equal to 1:

- charge unit - elementary charge e (charge of the proton, the absolute value of the electron charge),
- distance unit - average distance between electron and proton in hydrogen atom (Bohr radius) a_0,
- action unit - reduced Planck constant $\hbar$,
- mass unit - mass of the electron m_e.

In this unit system and in spherical coordinates with atomic nuclei put in its origin, the potential has the following form [28]:

$$V(r) = -\frac{Z}{r}. \tag{4.68}$$

Solution of the Schrödinger equation can be expressed as a product of the radial part $R_{nl}(r)$ and the angular part $Y_{lm}(\vartheta, \varphi)$ [27]:

$$\psi_{nlm} = R_{nl}(r) Y_{lm}(\vartheta, \varphi). \tag{4.69}$$

Solutions are quantized by three integer numbers: n — the principal quantum number (consecutive positive natural numbers), l — azimuthal quantum number (values from 0 to $n-1$) and m — magnetic quantum number (values from $-l$ to $+l$).

The radial part of Schrödinger equation has the following form:

$$\left[-\frac{1}{r^2}\frac{\mathrm{d}}{\mathrm{d}r}\left(r^2\frac{\mathrm{d}}{\mathrm{d}r}\right) + 2\mu\left(V(r) - E\right) + \frac{l(l+1)}{r^2}\right] R_{nl}(r) = 0, \tag{4.70}$$

where μ is the reduced mass of the system:

$$\mu = \frac{m_n m_e}{m_n + m_e}, \tag{4.71}$$

where m_n is the mass of the nucleus and m_e is the mass of the electron. Due to the fact that m_n is approximately 2000 times higher than m_e, the reduced mass becomes:

$$\mu \approx \frac{m_n m_e}{m_n} = m_e, \tag{4.72}$$

in atomic units: $\mu = 1$.

The radial part has the following form:

$$R_{nl}(r) = N_{nl} e^{-\frac{Zr}{na_\mu}} \left(\frac{2Zr}{na_\mu}\right)^l L^{2l+1}_{n-l-1}\left(\frac{2Zr}{na_\mu}\right), \tag{4.73}$$

where a_μ is the Bohr radius a_0 divided by the reduced mass of the system, N_{nl} is the normalization constant:

$$N_{nl} = \sqrt{\left(\frac{2}{n}\right)^3 \frac{(n-l-1)!}{2n\,(n+l)!}}, \tag{4.74}$$

and L stands for the generalized Laguerre polynomial:

$$L^0_p(x) = L_p(x) = e^x \frac{\mathrm{d}^p}{\mathrm{d}x^p}\left(x^p e^{-x}\right), \tag{4.75}$$

$$L^q_p(x) = (-1)^q \frac{\mathrm{d}^q}{\mathrm{d}x^q} L_{q+p}(x). \tag{4.76}$$

The angular part is described by spherical harmonics:

$$Y_{ml}(\vartheta, \varphi) = N_{ml} P_l^{|m|}(\vartheta) e^{im\varphi}. \tag{4.77}$$

The energy for a hydrogen-like atom is equal to:

$$E_n = -\frac{Z^2}{2n^2}. \tag{4.78}$$

It is worth noting that energy depends only on the principal quantum number, so that states with equal n values but different azimuthal or magnetic quantum numbers have the same energy; they are therefore degenerate. This is not the case with multielectron systems. Figure 4.6 shows visualizations of some eigenstates for the hydrogen-like atom. This model is crucial to quantum chemistry. It is the only atomic system for which it is possible to solve the Schrödinger equation analytically. It is also the starting point for the development of theories going beyond standard non-relativistic quantum mechanics, such as Dirac's theory or quantum electrodynamics.

4.6 Density Functional Based Tight-Binding Method

Since it is impossible to solve analytically the Schrödinger equation for atomic systems with more than one electron, approximate methods have been developed. One of the first was the Hartree-Fock method [27] and its extension with corrections based on perturbation theory - the MPx (especially MP2) methods [62].

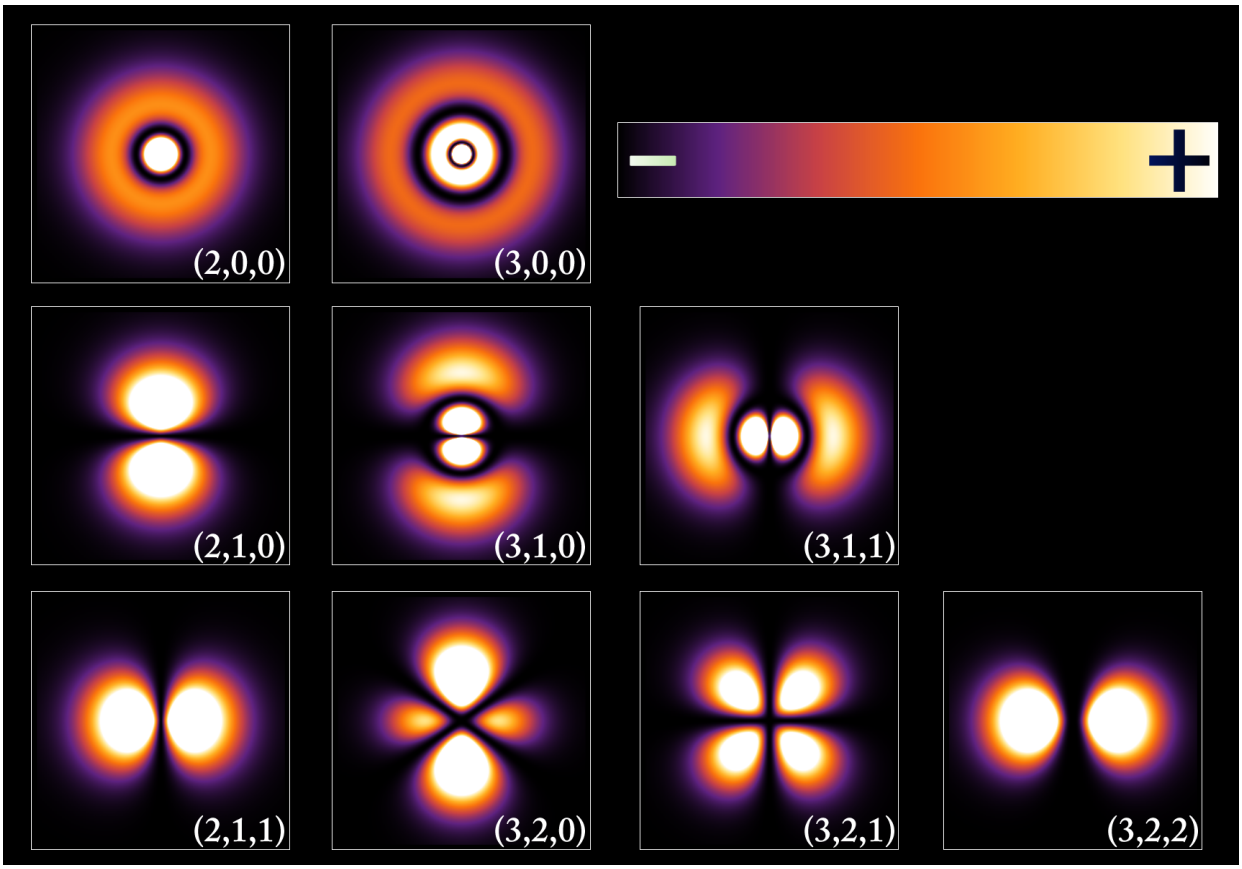


Figure 4.6: Visualisations of some hydrogen-like atom eigenstates, described in form (n, l, m) [61]

The mentioned methods concentrate on the use of wavefunctions. As has been shown by Hohenberg and Kohn [63], it is possible to use an equivalent theory based on electronic density (square of modulus of the wavefunction) called Density Functional Theory (DFT). Computational methods based on DFT are a standard approach in the modern computational chemistry.

In such calculations, the so-called basis set of functions marked as $\{\chi_i\}$ is used. The resulting wavefunctions $\{\psi_i\}$ are obtained as a linear combination of basis functions:

$$\psi_i = \sum_j c_j^i \chi_j. \tag{4.79}$$

Coefficients c_j are calculated in the way that minimizes the obtained energy for the system. Usually, the overlap matrix S_{ij} is also used. It is defined as:

$$S_{ij} = \int_{\mathbb{R}^6} \chi_i^*(\vec{r}_1)\chi_j(\vec{r}_2)\mathrm{d}^3\vec{r}_1\mathrm{d}^3\vec{r}_2. \tag{4.80}$$

Although DFT provides an acceptable accuracy for many problems, it is slow when applied to large systems. This led to the development of simplified approximate methods. One of them was proposed by Slater and Koster in the 1950s [64] and was further extended in Density Functional Based Tight Binding (DFTB) [21] method. In this approach, the energy of the system of N atoms is expressed as [65]:

$$E = E_{\mathrm{BS}} + E_{\mathrm{fluct}} + E_{\mathrm{rep}}. \tag{4.81}$$

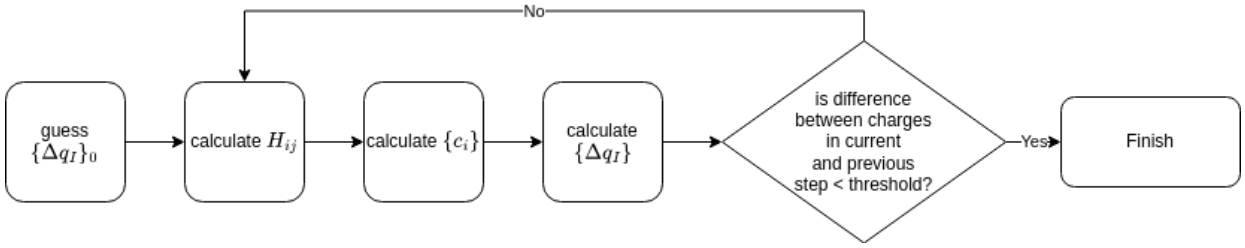


Figure 4.7: Scheme of the SCC-DFTB method

The last term is the repulsion energy, which is expressed as:

$$E_{\text{rep}} = \sum_{I<J} V_{\text{rep}}^{IJ}(R_{IJ}), \tag{4.82}$$

where R_{IJ} is the distance between atom I and atom J, $V_{\text{rep}}^{IJ}(R_{IJ})$ is the function of repulsion between atoms of type I and J, and is one of the parameters of the DFTB method.

The second term in Equation 4.81 is the electron density fluctuation term:

$$E_{\text{fluct}} = \frac{1}{2} \sum_{I,J} \gamma_{IJ}(R_{IJ}) \Delta q_I \Delta q_J, \tag{4.83}$$

where $\gamma_{IJ}(R_{IJ})$ is another parameter of the DFTB method and Δq_I is the partial charge on atom I.

The term E_{BS} is the band-structure energy:

$$E_{\text{BS}} = \sum_{i=0}^{N} \sum_{jm} c_j^{i*} c_m^i H_{jm}^0, \tag{4.84}$$

where H_{jm}^0 are parameters of DFTB method.

By finding the minimum of energy described by Equation 4.81, the set of equations for the optimal coefficients c_j^i is obtained in the form:

$$\sum_j c_j^i \left(H_{mj} - \varepsilon_i S_{mj}\right) = 0, \tag{4.85}$$

where ε_i are undetermined Lagrange coefficients and H_{mj} is expressed as:

$$H_{mj} = H_{mj}^0 + \frac{1}{2} S_{mj} \sum_K \left(\gamma_{IK} + \gamma_{JK}\right) \Delta q_K, \quad m \in I, j \in J. \tag{4.86}$$

As mentioned, above Equations 4.82, 4.83 and 4.84 include parameters V_{rep}^{IJ}, γ_{IJ} and H_{jm}^0. They are usually available in the format of the so-called Slater-Koster files. There are some publicly available sets of these parameters. One of them, which has been used in this work, is 3ob-3-1 [66, 67, 68] developed for modeling organic and biological systems.

Equation 4.86 is solved iteratively: an initial guess of charges $\{\Delta q_i\}$ is made, from which elements H_{mj} are calculated. From them coefficients $\{c_j^i\}$ are calculated and then, new char-

ges $\{\Delta q_i\}$ are obtained. If the difference between the charges in the current and previous step is less than an assumed threshold, the procedure ends, otherwise the loop repeats. This is called the Self-Consistent-Charge [69] variant of the DFTB method (SCC-DFTB). This procedure is presented schematically in Figure 4.7. In the SCC variant, the most time-consuming step is the determination of partial charges for each atom, so it seems a promising step to be replaced by the use of a neural network.

There are available some computational packages in which calculations with the DFTB method are possible, among them there are DFTB+ [70] and CP2K [71].

4.7 Molecular dynamics

Protons and neutrons are about 2000 times heavier than electrons. This leads to one of the basic approximations beyond most applications of quantum chemistry, that there is a significant difference in the speed of their motion. It is therefore justified to separate their motions, which is called the Born-Oppenheimer approximation [28].

Molecular dynamics is the name given to methods used to determine the dependence of atomic positions on time based on the extension of the Born-Oppenheimer approximation — the assumption that atomic nuclei are classical objects moving in the potential V, which comes from electrons (their movements and repelling forces). Thus, it is possible to determine the forces acting on the nucleus at the position $\vec{R}_\alpha$:

$$\vec{F}_\alpha = -\nabla_\alpha V(\vec{Q}), \tag{4.87}$$

where $\vec{Q}$ denotes nuclei positions, $\vec{F}_\alpha$ denotes the force acting on α-th atom and ∇_α is the gradient at the position of α-th atom.

When forces are known, it is possible to determine acceleration from the Newton equation:

$$M_\alpha \ddot{\vec{R}}_\alpha = \vec{F}_\alpha. \tag{4.88}$$

In standard molecular dynamics, it is assumed that the forces do not change during an arbitrarily set timestep Δt (usually small, less than 1.0 fs), and atoms are moving with constant acceleration during this time. Next, a new potential is calculated and the procedure continues.

Determination of positions $\vec{R}_\alpha$ of atoms after the timestep Δt involves numerical solving of Newton equation. Usually, Velocity Verlet algorithm [72] is used. In the implementation available in Schnetpack, at first new atom position $\vec{R}_\alpha(t + \Delta t)$ is determined on the basis of its momentum $\vec{p}_\alpha(t)$ and mass M_α:

$$\vec{R}_\alpha(t + \Delta t) = \vec{R}_\alpha(t) + \frac{\vec{p}_\alpha(t)}{M_\alpha}\Delta t, \tag{4.89}$$

and then, new momentum $\vec{p}_\alpha(t+\Delta t)$ is calculated using force $\vec{F}_\alpha(t)$:

$$\vec{p}_\alpha(t+\Delta t) = \vec{p}_\alpha(t) + \frac{1}{2}\vec{F}_\alpha(t)\Delta t. \tag{4.90}$$

In the ab initio approach, V is determined by solving the Schrödinger equation for electrons in the first step [25]:

$$\hat{H}_e\psi(\vec{q};\vec{Q}) = E_e\psi(\vec{q};\vec{Q}). \tag{4.91}$$

Here, $\vec{q}$ denotes positions of N electrons and $\vec{Q}$ denotes nuclei positions and $\hat{H}_e$ is the Hamiltonian for electrons.

Next, the forces are calculated from the average electron energy value:

$$M_\alpha\ddot{\vec{R}}_\alpha = -\nabla_\alpha \min_\psi \int_{\mathbb{R}^{3N}} \psi^*(\vec{q};\vec{Q})\hat{H}_e\psi(\vec{q};\vec{Q})\,\mathrm{d}\vec{q}. \tag{4.92}$$

In most cases, the equation 4.91 is solved using DFT methods (Hartree-Fock is too inaccurate, while MP2 is too slow). It could also be accelerated with DFTB, as it is implemented in DFTB+ [70].

Usually, the simulation needs to be performed at a constant temperature. This requires additional mechanisms for system control. Without these mechanisms, atoms could accelerate their movements, what would lead to a rapid increase in temperature. Such mechanisms are called thermostats. One of them if the Langevin thermostat [73] used in simulations carried out in the research for this thesis. In this approach, Equation 4.88 is modified in the following way:

$$M_\alpha\ddot{\vec{R}}_\alpha = \vec{F}_\alpha - \gamma_\alpha M_\alpha\dot{\vec{R}}_\alpha + f_\alpha, \tag{4.93}$$

where γ_α is the friction coefficient having time unit and f_α is a random force chosen from a Gaussian distribution with variance equal:

$$\sigma_\alpha^2 = \frac{2M_\alpha\gamma_\alpha k_B T}{\Delta t}, \tag{4.94}$$

where k_B is the Boltmzann constant, T is the system temperature and Δt is the timestep used in MD simulation. This modification of the Newton's equation prevents molecules from too rapid acceleration and keeps the average temperature of the system at a nearly constant level.

Molecular dynamics methods are widespread in modern chemistry, as they allow one to determine many properties of materials, e.g., infrared spectra, diffusion coefficients, electrical conductivity, lattice vibrations and proton transfers [25].

4.8 Infrared spectroscopy

In molecules, their rotational, vibrational, and electronic energetic states are discrete, which makes possible to observe transfers between them only when the energy delivered fits the difference between the initial and final state. One of the most popular ways to deliver energy to

the system is to use light at a given frequency range. Measuring the amount of light absorbed as a function of frequency gives a spectrum, which is a characteristic property of a system. For example, infrared is able to excitate vibrational states in molecules, and infrared spectra are commonly used in organic chemistry to identify functional groups by the presence of specific bands. It is also commonly used to study the interactions between molecules in a given system. For example, in the work [74] oscillation spectra were measured for systems with increasing concentration of sodium salt, and changes in band positions related to interactions between the solvent and sodium cations were observed. The sample IR spectrum is presented in Figure 4.8.

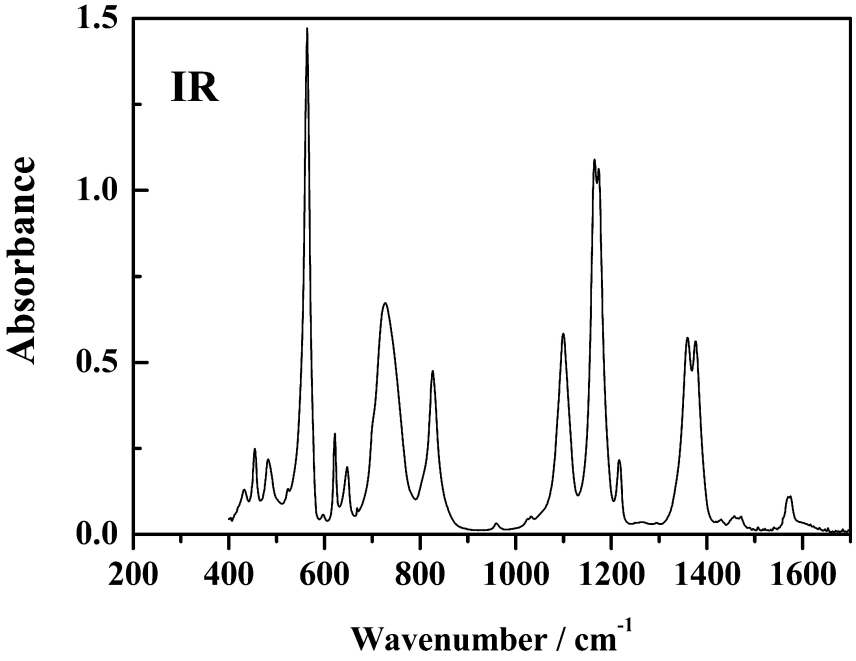


Figure 4.8: Sample IR spectrum of 1-Ethyl-3- methylimidazolium Bis(fluorosulfonyl)imide ionic liquid [75]

Vibrations present in a molecular system are usually described in the system of normal coordinates $\vec{q}$, where the Hessian of the system energy is diagonal.

Non-zero intensity of a band appearing in an IR spectrum is connected with a vibration, which changes the total dipole moment of a system [76]:

$$I_{\mathrm{IR}} \sim \left| \left(\frac{\partial \vec{\mu}}{\partial \vec{q}} \right)_{\vec{q}=\vec{q}_e} \right|^2, \tag{4.95}$$

where I_{IR} stands for the intensity of a band in an IR spectrum, $\vec{\mu}$ stands for the dipole moment, $\vec{q}$ stands for the normal coordinate which changes during the vibration, and $\vec{q}_e$ stands for the equilibrium position for this vibration.

The frequency of the vibration could be related to the bond strength in the case of bond stretching vibrations. This relationship could be approximated as [77]:

$$\nu \approx \frac{1}{2\pi} \sqrt{\frac{\kappa}{\mu_{\mathrm{red}}}}, \tag{4.96}$$

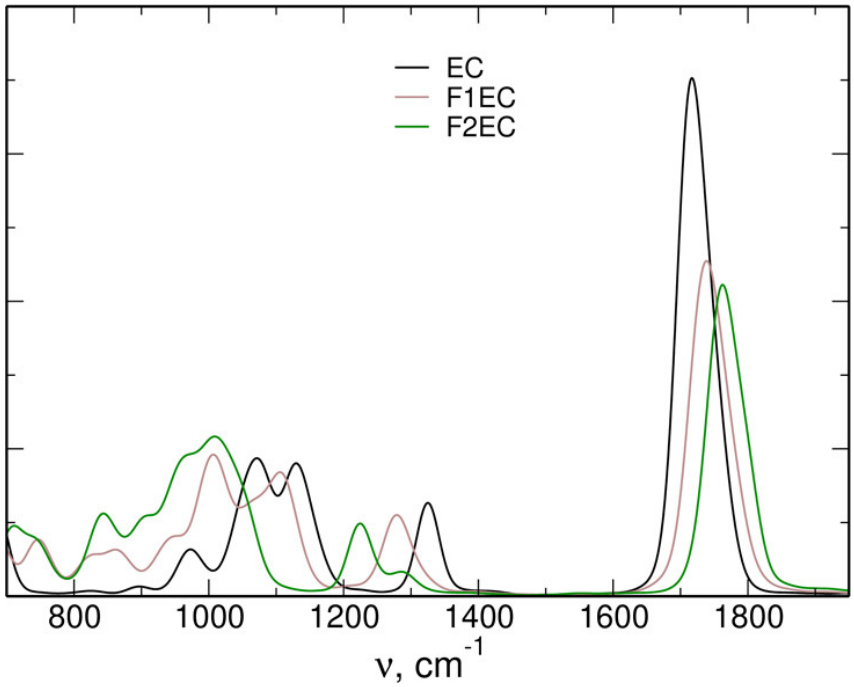


Figure 4.9: IR spectra for ethylene carbonate (EC) and its mono- (F1EC) and difluorinated (F2EC) derivatives obtained with the use of DFTB-based molecular dynamics [16].

where κ is the force constant of the vibration, and μ_{red} is the reduced mass of the vibrating system. Thus, the frequency is proportional to the square root of κ, which leads to the assumption, that in most cases the higher the frequency of the vibration, the stronger the vibrating bond.

In order to reproduce the IR spectrum from the molecular dynamics simulation, the total dipole moment of the system is calculated at each time step. Next, the dependence of the shape function $I(\nu)$ on the frequency ν is calculated [78, 79]:

$$I(\nu) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \langle \vec{\mu}(0) \cdot \vec{\mu}(t) \rangle \, \mathrm{e}^{-2\pi i \nu t} \, \mathrm{d}t, \tag{4.97}$$

which is used to determine the absorption coefficient $\alpha(\nu)$ [78, 80]:

$$\alpha(\nu) = \left[\frac{16\pi^4 \nu}{3\Omega h c n(\nu)} \right] \left(1 - \mathrm{e}^{-\frac{h\nu}{k_B T}} \right) Q(\nu) I(\nu), \tag{4.98}$$

where: h is the Planck constant, k_B is the Boltzmann constant, T is the temperature of the system, $n(\nu)$ is the refractive index, c is the light velocity, Ω is the volume of the simulated system and $Q(\nu)$ is the correction function. One of the commonly used $Q(\nu)$ is the harmonic correction [80, 81]:

$$Q(\nu) = \frac{\frac{h\nu}{k_B T}}{1 - \mathrm{e}^{-\frac{h\nu}{k_B T}}}. \tag{4.99}$$

Usually, $n(\nu)$ and its dependence on light frequency is not known, thus IR spectra obtained from MD simulations are commonly presented as the frequency dependence of the product $\alpha(\nu)n(\nu)$. In the literature, there are many examples of systems for which such spectra have

been successfully obtained, including research done by the autor of this thesis [16]. Sample theoretical IR spectra are presented in Figure 4.9.

The most important effect observed for these spectra is the change in frequency of the band near 1700 cm^{-1}. The analysis described in [16] has shown that this band is associated with the C=O bond stretching vibration. The position of the maximum moves towards higher frequencies with increasing degree of fluorination of the solvent molecule. This indicates that the strength of the C=O bond increases as fluorine atoms are added to the structure of ethylene carbonate.

Chapter 5

Solving quantum chemistry problems with PINNs

This chapter presents the proposed in this thesis solution of quantum chemical problems described in Chapter 4. The first section describes the approach used to the preliminary problem of solving the Schrödinger equation for model systems: particle in a box, harmonic oscillator and hydrogen atom. The second section describes the molecular dynamics simulations carried out using neural networks to solve the main problem mentioned in this thesis.

5.1 Systems with analytical solutions of Schrödinger equation

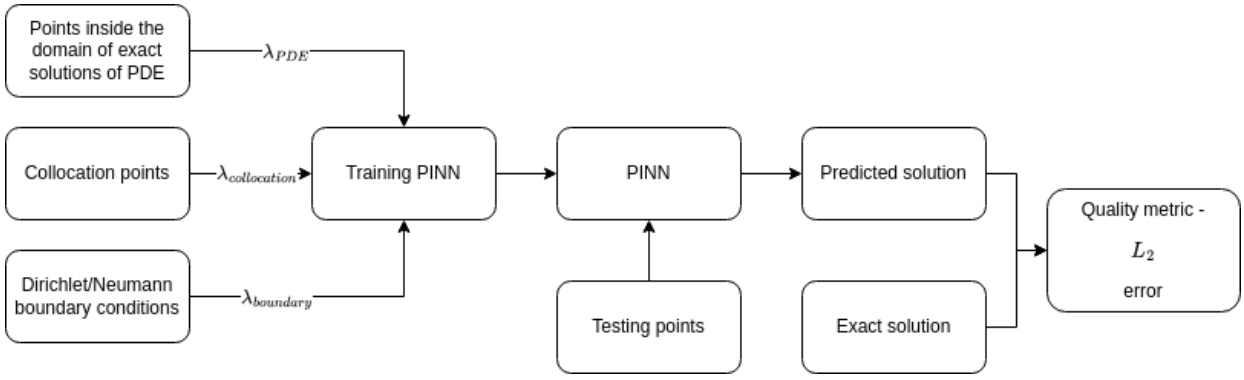


Figure 5.1: Data flow for solving the Schrödinger equation for model systems

For all the model quantum systems described in the previous chapter, a PINN is constructed and trained using a loss function with data points extracted from the exact known solution (weight λ_{PDE}) with highlighting some of them as collocation points (with different weight in the loss function $\lambda_{\text{collocation}}$) and with Dirichlet/Neumann boundary conditions for some cases (weight $\lambda_{\text{boundary}}$). In the next step, the accuracy of the neural network is verified by comparing the values predicted by PINN with exact values for test points not present in the training set. As a metric, the L_2 relative error is calculated. The data flow is presented in Figure 5.1.

Different approaches to the problem of solving Schrödinger equation for systems with known exact solution are described. They start with a basic approximation of the known function by the neural network, next they use the ability of PINN to learn from PDE for fixed quantum numbers and end with approaches generalised in terms of quantum numbers. One approach treats the whole domain of spatial variables and quantum numbers as continuous variables and the second approach defines its own domain with points having only integer values of quantum numbers. The example of the PINN architecture for a 1D function and non-fixed quantum number n is shown in Figure 6.1.

The abilities of PINNs to generalize problems are also studied in terms of the use of different quantum numbers. The influence of the structure of the neural network (number of layers, number of neurons per layer, choice of collocation points, weights in the loss function) on the accuracy of the prediction is also examined.

5.2 Molecular dynamics

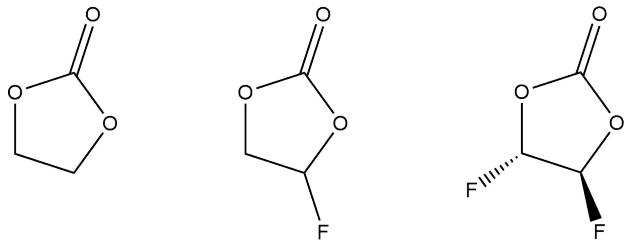


Figure 5.2: Structures of carbonates modeled by MD: left - ethylene carbonate (EC), middle - monofluoroethylene carbonate (F1EC), right - *trans*-difluoroethylene carbonate (F2EC)

As it was described in Section 4.6, the most time-consuming part of DFTB-based MD simulations is the determination of the atomic charges. In order to accelerate this process, the neural network approach is used in this work with the Schnetpack package [57]. This package is suitable for molecular systems, as it uses for each atom its neighbourhood as the input to the NN. The typical architecture of the NN used in Schnetpack is presented in Figure 3.13. Specific layers are described in Section 3.3.

One of the systems which are nowadays considered as potential electrolytes for the new generation of non-lithium batteries are systems based on liquid ethylene carbonate solutions [11], due to their stability. In addition, these systems have been chosen because they have already been studied with the DFTB method [16]. Structures of the modeled carbonates are presented in Figure 5.2.

The typical usage of Schnetpack involves training the NN to predict the forces acting on atoms in the system, and the system energy, and on their basis to perform molecular dynamics (MD) simulation. However, the preliminary study for systems with EC or water molecules [82] has shown that this approach has problems with the stability of the simulation. Thus, an alternative has been developed for this thesis.

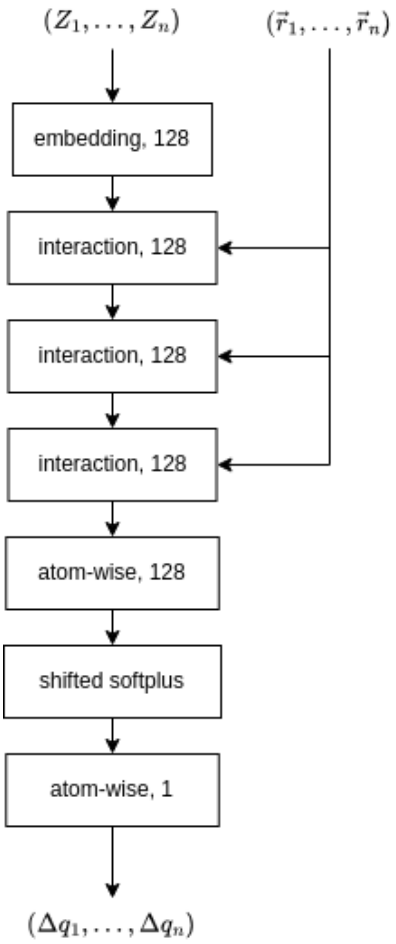


Figure 5.3: Structure of SchNet used for predictions of partial charges $(\Delta q_1, ..., \Delta q_n)$ on the basis of atomic numbers $(Z_1, ..., Z_n)$ and their positions $(\vec{r}_1, ..., \vec{r}_n)$. For each layer, the number of neurons is given. Layers are described in Section 3.3.

To perform MD simulation, in the first step a proper dataset is prepared. For this purpose, charges obtained during DFTB simulations are used (results from [16]). Next, a SchNet neural network, which is capable of predicting atomic charges based on the atomic numbers and atomic positions, is trained on this data. The structure of the SchNet is presented in Figure 5.3. The detailed description of specific layers is available in Section 3.3. The mean absolute error (MAE) is used as a training metric.

After training, the NN is used as an intermediate step in solving the Schödinger equation. The neural network receives as an input the initial geometry of the system $(\vec{r}_1, ..., \vec{r}_n)$ and atomic numbers $(Z_1, ..., Z_n)$ and predicts from them partial charges $(\Delta q_1, ..., \Delta q_n)$. This replaces the SCC procedure of the DFTB method, which is presented schematically in Figure 4.7 and described in Section 4.6. The partial charges, are used by DFTB+ API [70] as a starting point for the calculation of system energy E and the forces acting on atoms $(\vec{F}_1, ..., \vec{F}_n)$. Forces are used by Newton's equation integrator (available in Schnetpack) to determine new atom positions $(\vec{r}'_1, ..., \vec{r}'_n)$ after the timestep Δt. These new positions are used as an input to SchNet for the next iteration of the molecular dynamics simulation.

For each step, the positions of atoms, as well as their partial charges are saved. From the

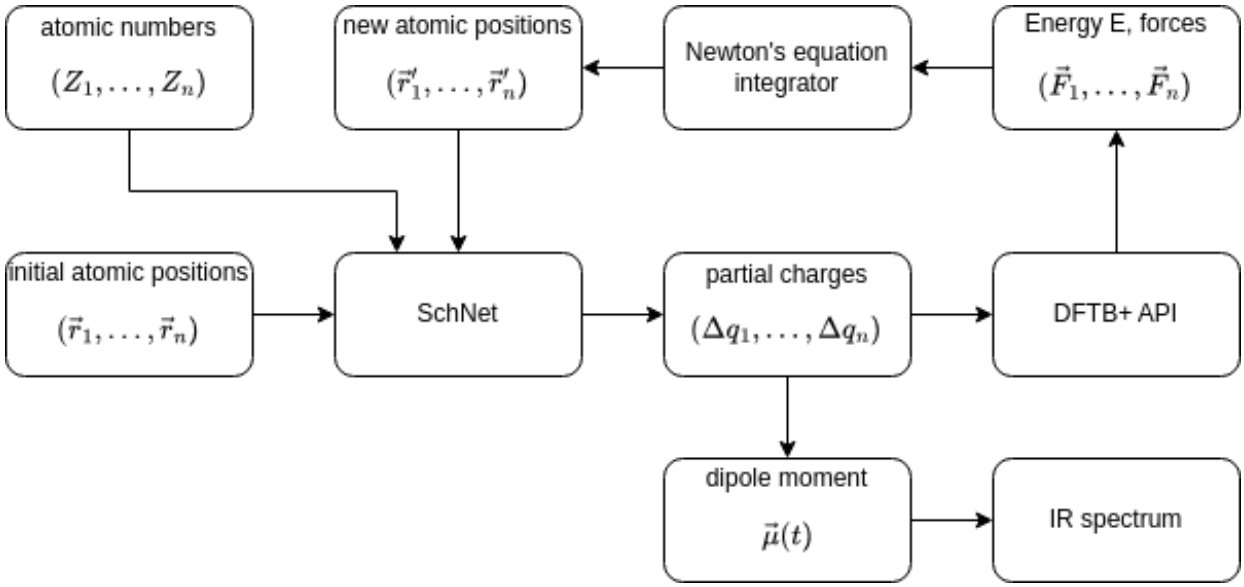


Figure 5.4: Data flow for performing molecular dynamics with the NN-enhanced DFTB method

partial charges, for each step the total dipole moment of the system $\vec{\mu}(t)$ at time t is calculated as the following sum:

$$\vec{\mu}(t) = \sum_{i}^{n} \Delta q_i(t) \vec{r}_i(t). \tag{5.1}$$

From dipole moment dependence on time, the theoretical IR spectrum is calculated using Equation 4.98. As an additional verification of the efficiency of this method, the obtained spectrum is compared with spectra obtained by the standard DFTB method.

Schematically, the procedure of DFTB-based MD enhanced with SchNet is presented in Figure 5.4. It should be highlighted, that this approach has not been used before and is a novel idea developed in this thesis.

Chapter 6

Experiments

This chapter contains results of experiments performed to verify the research hypothesis proposed in Section 1.3. At the beginning, computational details of experiments are described, and in the following sections particular results are presented. The sample architecture of the PINN used for the model quantum systems is presented in Figure 6.1.

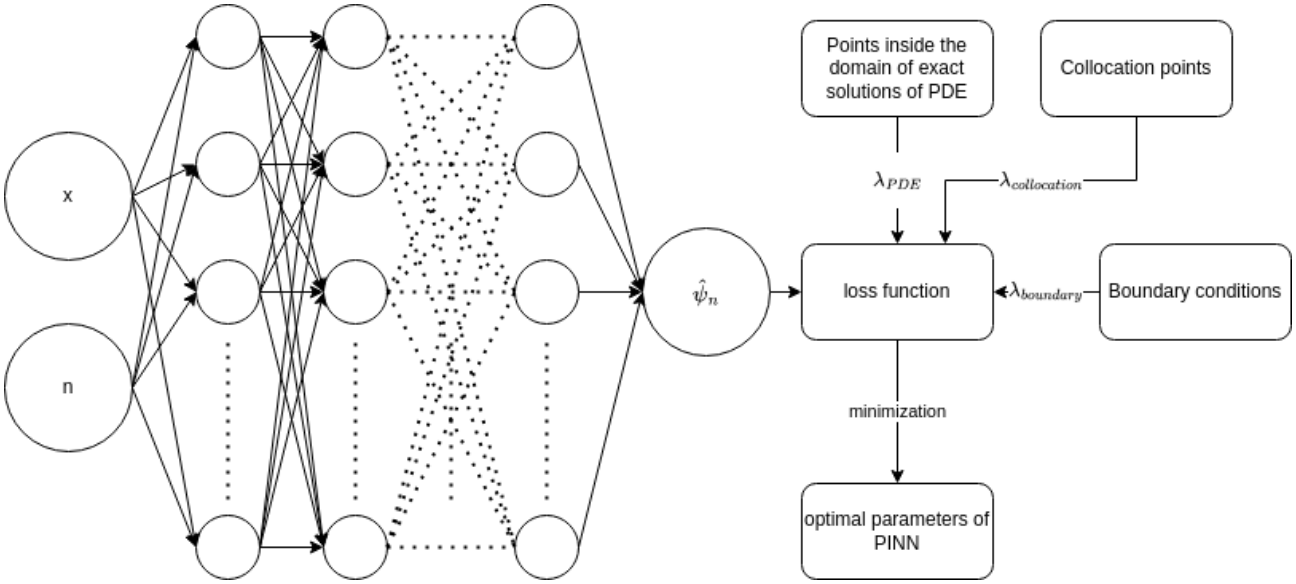


Figure 6.1: Example of PINN for prediction of a solution $\hat{\psi}_n$ for a model quantum system with non-fixed quantum number n

6.1 Details of experiments

Experiments for model systems of quantum chemistry were performed with use of the 1.8.2 version of the DeepXDE library [55], Python 3.10.6, Tensorflow v1 backend and GPU acceleration based on CUDA version 11.8. The older version of Tensorflow (at the time of writing this thesis the v2 version was available) was used due to the best performance among all backends utilized in DeepXDE at the time of writing this text. This part could be implemented without the use of external resources, on a personal computer with a 64-bit Intel Core i5-7200 U processor with

clock frequency equal to 2.50 GHz, 12 GB of RAM, and an NVIDIA GeForce 940MX graphics card. For some problems, implementation of special functions from Scipy [83] were used.

For model problems, the hyperparameters of experiments (the number of training points, test points, boundary points, the number of training epochs, boundary conditions, conditions based on collocation points, loss weights, and the structure of the PINN) are different for particular examples and are defined in the corresponding sections. The common hyperparameters were the L-BFGS optimizer, hyperbolic tangent as the activation function and Glorot uniform option for initialization. L-BFGS optimizer does not need learning rate, so it was not a hyperparameter of training NNs.

The general form of the loss function is the following:

$$\begin{aligned}\mathcal{L} = &\frac{\lambda_{\text{PDE}}}{N_{\text{inside}}} \sum_{i=1}^{N_{\text{inside}}} \left| \mathcal{N}\left(x_i, \hat{\psi}(x_i), \nabla\hat{\psi}(x_i), \Delta\hat{\psi}(x_i)\right)\right|^2 + \\ &+ \frac{\lambda_{\text{boundary}}}{N_{\text{boundary}}} \sum_{i=1}^{N_{\text{boundary}}} \left| \mathcal{B}\left(x_i, \hat{\psi}(x_i)\right)\right|^2 + \\ &+ \frac{\lambda_{\text{collocation}}}{N_{\text{collocation}}} \sum_{i=1}^{N_{\text{collocation}}} \left|\hat{\psi}(x_i) - \psi(x_i)\right|^2,\end{aligned} \tag{6.1}$$

where λ_{PDE}, $\lambda_{\text{boundary}}$, $\lambda_{\text{collocation}}$ are weights of the particular parts of the loss function, N_{inside} is the number of training/test points inside the domain, $\mathcal{N}$ is the operator representing the PDE, N_{boundary} is the number of points for boundary conditions, $\mathcal{B}$ is the operator representing the boundary conditions, $N_{\text{collocation}}$ is the number of collocation points, $\hat{\psi}(x_i)$ is the prediction of the PINN at point x_i and $\psi(x_i)$ is the desired value of the prediction for collocation point x_i. Collocation points, are the points chosen from the interior of the problem domain, highlighted with different value of weight than for other training points.

In addition, L_2 relative error, M, is calculated, according to the following definition:

$$M = \sum_{i=1}^{N_{\text{test}}} \frac{\left|\hat{\psi}(x_i) - \psi(x_i)\right|^2}{\left|\psi(x_i)\right|^2}, \tag{6.2}$$

where $\psi(x_i)$ is the value of the true solution at test point x_i.

For convenience, the atomic unit system has been used, what leads to $\hbar = 1$, the mass of electron $m_e = 1$ and the elementary charge $e = 1$.

The default approach in DeepXDE is used as the distribution of points used in model training and evaluation – training points on the boundary are chosen randomly while training points inside the domain and test points are chosen uniformly. To ensure determinism of the experiments, the random seed value was set to 1234.

The NN for prediction of partial charges on atoms of a given system was defined using Schnetpack version 2.0.3. Three NNs were trained on different datasets for particular systems: neat EC, neat F1EC, neat F2EC. These datasets were taken from the results of the paper [16]. NNs were trained for 200 epochs using a batch of size 8. The number of data points for each dataset is

Table 6.1: Number of training and validation points for used datasets

System	Training points	Validation points
EC	40 500	4 501
F1EC	40 500	4 501
F2EC	40 500	4 501

listed in Table 6.1. The NN was the SchNet with 3 interaction layers, a cutoff equal to 7.5 Å (the distance above which the atoms are treated as non-interacting) with pairwise distances expanded on 20 Gaussians and 128 atomwise features and convolution filters. The optimizer was AdamW with a learning rate of 10^{-4}, as suggested in the Schnetpack documentation. Mean absolute error was used as a loss metric during training. Training was performed with GPU acceleration.

Molecular dynamics simulations were performed using Schnetpack utils (thermostats, Newton's equations integrators) with determination of atomic partial charges done by previously trained NN. From these charges energies and forces acting on atomic nuclei were calculated using Python API of DFTB+ package in version 23.1. For integration of Newton equation, a Velocity Verlet algorithm was used with a timestep of 0.5 fs and the simulations were 70000 steps long for each of the systems considered. The temperature was set as 298 K with initialization of atomic velocities from the Maxwell-Boltzmann distribution. The cutoff was chosen as equal to 5.0 Å. To ensure stabilization of the temperature, a Langevin thermostat was used with bath temperature equal to 298 K and time constant (friction coefficient γ_α from Equation 4.93) equal to 100 fs. The energies and forces were calculated using the 3ob-3-1 [66, 67, 68] parameter set (as in the reference publication [16]) using the standard SCC-DFTB method in the third order approximation. For technical reasons – DFTB+ needs to perform at least one SCC cycle per calculation even with preset partial charges – the SCC tolerance parameter was set to 10^{-2}. Dispersion interactions were calculated with the use of the Universal Force Field [84].

After simulations from the last 30 ps of them (i.e. the last 60000 steps) dipole moments were determined and theoretical IR spectra were calculated using Equation 4.98. For the calculation of the theoretical IR spectra, the Fourier tool [85] from the CPMD [86] contributed tools was used with the application of the harmonic thermal correction.

This part of experiments was performed on the Ares cluster in the ACC Cyfronet.

6.2 Model problems

In this section, results obtained for model quantum chemical problems are presented, starting with a particle in a box, through harmonic oscillator and rigid rotor, ending with the hydrogen atom.

In general, four approaches have been applied to solve these problems (if possible):

- prediction of the solution from the NN trained on the exact solution for fixed quantum numbers (used as a reference),
- prediction of the solution from the PINN trained on the PDE for fixed quantum numbers,

- prediction of the solution from PINN for quantum numbers within a given range, where PINN is trained on PDE within the domain, using quantum number as an additional dimension, having continuous values,
- prediction similar to the preceding one, but using a custom domain admitting only integer values of quantum numbers.

Due to technical issues, for some model problems, the last two of these approaches were not possible to use. This was the case for solutions described by special functions, for example the Airy function for the particle in a triangular box. For them, Scipy implementation of these functions did not support automatic differentiation needed for training the NN within the Tensorflow backend.

In all considered PDEs, the explicit values of the system energy E were used. Results of the L_2 relative error dependence on different parameters of PINNs are plotted with dashed/continuous lines for easier tracking of the relations.

6.2.1 One-dimensional infinite box

For this case, a box with length L equal to 2 (L is defined in Section 4.2) was chosen for the quantum number n between 1 and 5. This means, that the domain of the problem is the range [-1, 1]. The mass of the particle was assumed to be unit ($m = 1$). All four approaches to the use of NNs, mentioned in the introduction to this section, were used.

The Schrödinger equation for this problem has the following form:

$$-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2\psi_n(x)}{\mathrm{d}x^2} = E_n\psi_n(x). \tag{6.3}$$

After substitution of $m = 1$, $L = 2$, $\hbar = 1$ and the value of energy E_n into this equation and a simple manipulation, the PDE for this case takes the following form:

$$\frac{1}{2}\frac{\mathrm{d}^2\psi_n(x)}{\mathrm{d}x^2} + \frac{n^2\pi^2}{8}\psi_n(x) = 0. \tag{6.4}$$

As this equation is second order, at least two points are necessary as boundary values. For the experiments, Dirichlet boundary conditions were used in the form:

$$\psi_n(\pm\frac{L}{2}) = \psi_n(\pm 1) = 0. \tag{6.5}$$

Choosing for this purpose these boundary conditions may lead to trivial solutions, because the function identically equal 0 also satisfies this equation. In addition to boundaries, it was needed to use collocation points.

Differentiating Equation 4.20 and equating it to zero, we obtain the following minima or maxima:

$$x = \frac{mL}{n} + \frac{L}{2n} - \frac{L}{2}, \quad m = 0, 1, 2, 3, \dots . \tag{6.6}$$

First n of them are inside the domain of the problem. Thus, instead of using random positions of collocation points it is also reasonable to use positions of extremal points of the solution.

PINN trained on PDE with fixed n

In order to choose optimal hyperparameters of learning, some initial experiments were performed, separately for all mentioned approaches, starting with PINN trained on PDE for fixed n. For all experiments, 32 training points, n collocation points, 2 boundary points and 100 test points were used.

First, different weights of collocation points and their choice (minima/maxima from equation 6.6 vs random choice) were studied. For these experiments, the number of layers was set to 5 and the number of neurons per layer was chosen to 20 (these values were derived from tutorials for usage of DeepXDE). The weight of collocation points, $\lambda_{\text{collocation}}$, relative to weight of satisfying the PDE, λ_{PDE}, (as defined in Equation 6.1) was set to 1, 10 and 100. The dependence of the L_2 relative error on weights and choice of these points for n equal to 1, 3 or 5 is shown in Figure 6.2. The same weight was used for boundary points ($x = \pm 1$), where the solution vanishes.

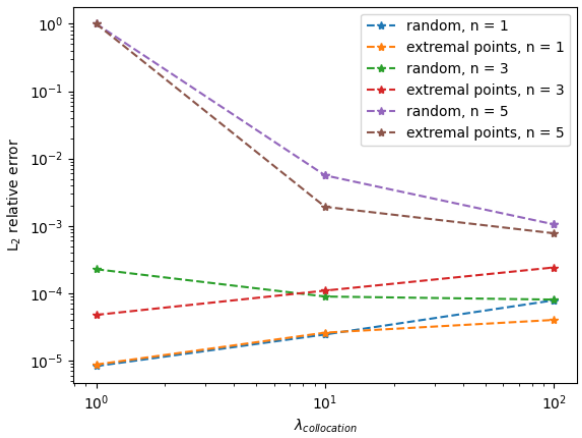


Figure 6.2: Dependence of the L_2 relative error on different weights $\lambda_{\text{collocation}}$ of collocation points and strategy of their choice. PINN was trained on PDE with fixed n value

From this picture, it can be observed, that the L_2 relative error values do not differ significantly between approaches of choice of collocation points. For low values of n (1 or 3) there is a small increase in value of the error with increasing weight, however for n equal to 5 the error decreases by nearly 3 orders of magnitude between $\lambda_{\text{collocation}}$ equal to 1 and 10, and then exhibits only a small decrease when it is changed to 100. Thus, for further experiments for potential well, randomly chosen collocation points with $\lambda_{\text{collocation}}$ equal to 10 were chosen, as the best compromise between reproduction of real conditions of experiments with PDEs when extremal points are not known and a reasonable contribution of λ_{PDE} in the loss function. However, in the next experiments for $n = 5$ a $\lambda_{\text{collocation}}$ value of 100 was used, due to incorrect predictions in some cases for this example with lower $\lambda_{\text{collocation}}$ values (trivial solution identically equal 0 for all values of x).

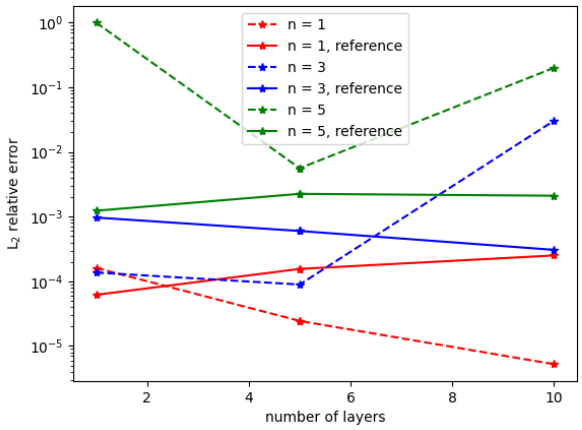


Figure 6.3: Dependence of the L_2 relative error on different number of layers in the PINN, 20 neurons each. PINN was trained on PDE with fixed quantum number n

Next, for chosen weights and strategy of choosing collocation points, a different number of layers in the PINN were examined (1, 5 or 10). The results are shown in Figure 6.3. Errors for simple function approximation by the NN are shown as the reference values. For each case, the lowest error is obtained for 5 layers of neurons, for 10 layers the error is larger for $n = 3$ or $n = 5$. This result could be explained with the fact, that 1 layer is insufficient for learning all the necessary features of the problem, while 10 layers overcomplicate the model and lead to worse predictions. Thus, 5 layers were chosen as the best value.

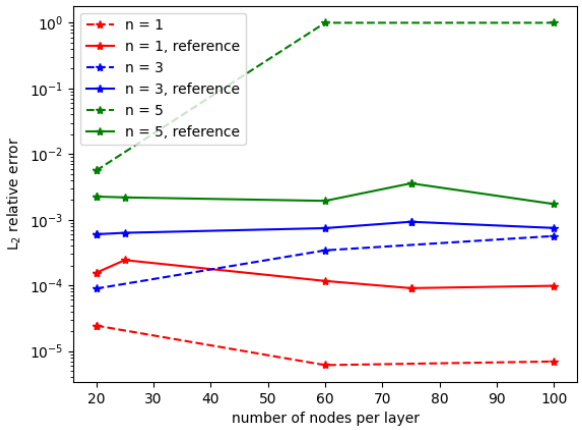


Figure 6.4: Dependence of the L_2 relative error on different number of nodes per layer in the PINN with 5 layers. PINN was trained on PDE with fixed quantum number n

A similar study was performed for the number of nodes per layer in the PINN equal to 20, 60 or 100. The results are shown in Figure 6.4. Except for the $n = 1$ case, the error increases with the number of nodes. As before, it is probably caused by the overcomplication of the model. Therefore, 20 nodes per layer were chosen as the best value.

PINN trained on PDE with n as a continuous variable

For the approach where the PINN is trained within the two-dimensional domain with quantum number n used as the continuous dimension, it was expected that the larger number of training points would be needed due to the larger size of the problem. Thus, 1024 of them were used, which is the square of the value used in the one-dimensional domain learning. Analogously, the number of test points was equal to 10000. Similarly, as the starting point, the larger number of nodes per layer equal to 75 was chosen with 5 layers in the PINN. This PINN was trained for n values between 1 and 5.

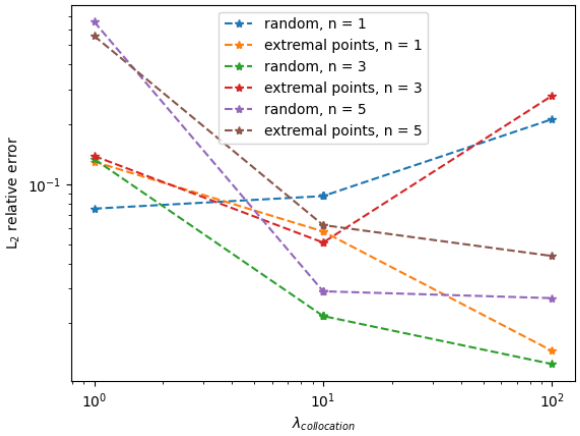


Figure 6.5: Dependence of the L_2 relative error on different weights $\lambda_{\text{collocation}}$ of collocation points and strategy of their choice. PINN was trained on PDE with continuous quantum number n values

Results of looking for the optimal value of $\lambda_{\text{collocation}}$ and the choice of collocation points are shown in Figure 6.5. Similarly as for the previous case, value of 10 was chosen as the best compromise.

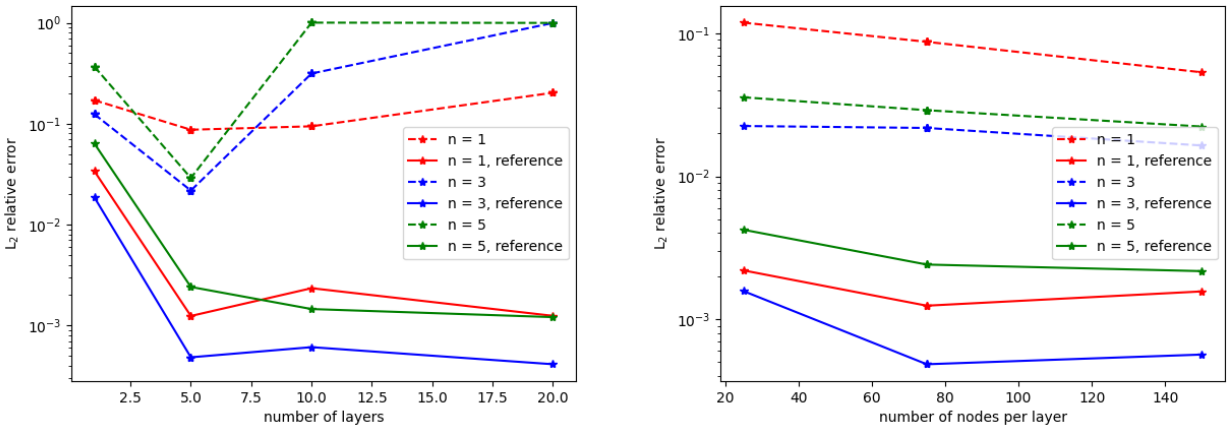


Figure 6.6: Dependence of the L_2 relative error on different number of layers 75 nodes each (left) and on different number of nodes for 5 layers (right). PINN was trained on PDE with continuous quantum number n values. Errors for fitting a function of continuous x and n are shown as a reference

Next experiments were performed for different number of layers with 75 nodes each (Figure 6.6, left panel). After choosing 5 layers as optimal, the errors for different numbers of nodes per layer were measured (Figure 6.6, right panel). From these experiments, 5 layers, each with 150 nodes, were found as the best structure of the PINN in this case. Obtained errors were 1-2 orders of magnitude higher than for the reference case (approximation of function by NN learned on function values for continuous x and n), however for chosen parameters were still acceptably low.

Table 6.2: Optimal hyperparameters of training models for one-dimensional infinite box

approach	N_{layers}	N_{nodes}	N_{train}	N_{test}	N_{boundary}	$\lambda_{\text{collocation}}$	epochs
PDE, fixed n	5	20	32	100	2	10	10000
PDE, continous n	5	150	1024	10000	10	10	10000
PDE, discrete n	5	20	160	500	10	10	10000

PINN trained on PDE with n as a discrete variable

The last examined case for one-dimensional infinite box was the approach using both quantum number n and x as input values in the PINN, but using only discrete values of n. In order to provide a reasonable reference to previous data, these PINNs were trained up to n_{max} equal to 5, with 160 training points ($32 \cdot n_{max}$) and 500 test points ($100 \cdot n_{max}$).

An analogous study of choice of $\lambda_{\text{collocation}}$ was done and its results are shown in Figure 6.7. The PINN consisted of 5 layers, 20 nodes each. Similarly as in previous cases, random collocation points with $\lambda_{\text{collocation}}$ equal to 10 were chosen for further experiments.

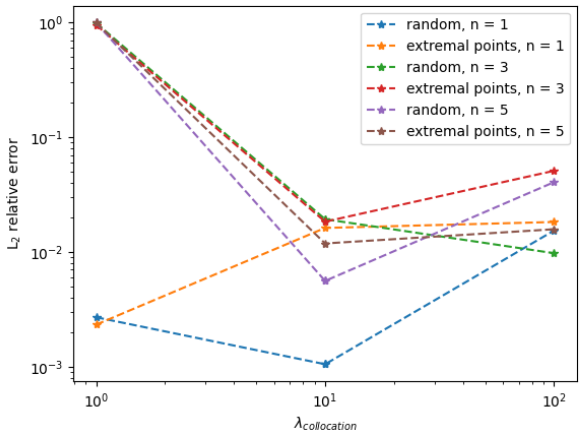


Figure 6.7: Dependence of the L_2 relative error on different weights $\lambda_{\text{collocation}}$ of collocation points and strategy of their choice. PINN was trained on PDE with discrete quantum number n values

Next, an optimal value of the number of layers in PINN with 20 nodes each was found to be equal to 5 (Figure 6.8, left panel). Then the optimal number of nodes per layer was found to be equal to 20 (Figure 6.8, right panel).

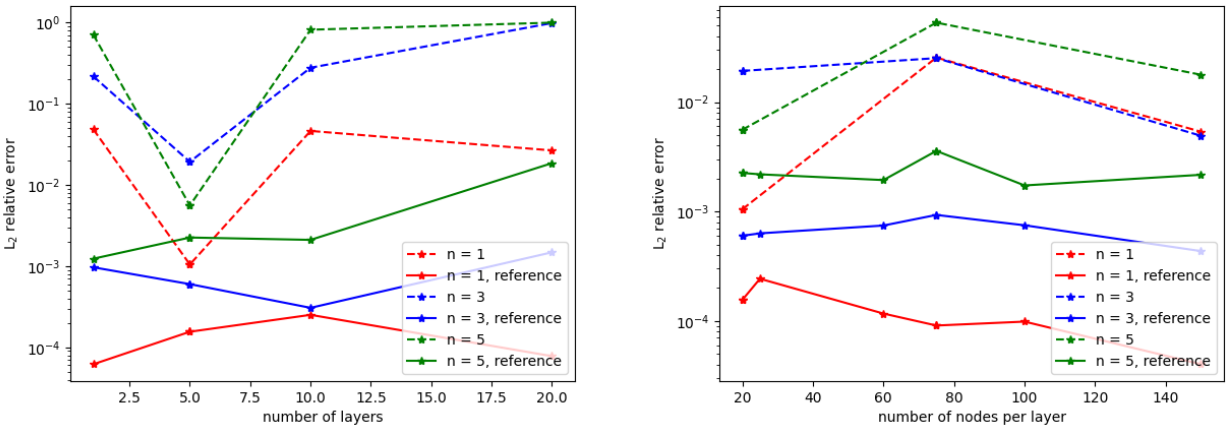


Figure 6.8: Dependence of the L_2 relative error on different number of layers 20 nodes each (left) and on different number of nodes for 5 layers (right). PINN was trained on PDE with discrete quantum number n values. Errors for fitting a function of continuous x and n are shown as a reference

Summary

Optimal hyperparameters for each approach are presented in Table 6.2. For PINN trained with fixed n values, the error was lower than for the reference method (approximation of the solution by NN trained on the exact function values). For the other approaches the errors were higher than in the reference case, but it should be noted that they were still in an acceptable range (below 10^{-1}). This example was difficult for modeling, despite having non-complicated functions as solutions (trigonometric functions), it needed a proper choice of collocation points. This was due to the fact that this problem has a trivial solution – function identically equal 0. Thus, setting only Dirichlet boundary conditions was insufficient to obtain correct predictions.

Examples of predictions made by PINNs are shown in Figure 6.9. It presents one advantage of the last two approaches using quantum number n as an additional input parameter: they are able to predict solutions outside the training domain (here for $n = 6$), however, the quality of the results is not perfect.

To compare approaches, the L_2 relative error dependence on quantum number n value is plotted for all of them in Figure 6.10. The results for reference model are not shown because the PINN hyperparameters varied between the approaches. For all studied n values, the PINN trained with fixed n resulted in the lowest error, however for $n = 5$ it is approximately equal to error obtained from other approaches. For low values of n, lower error is obtained in the approach with discrete n compared to one with continuous n. As n increases, these errors tend to become similar. Thus, because of the smaller size of PINNs and the smaller number of training points, in generalised approaches (using n as a variable), an approach with discrete n seems to be better because of less time-consuming process of training (compared to the model with continuous n).

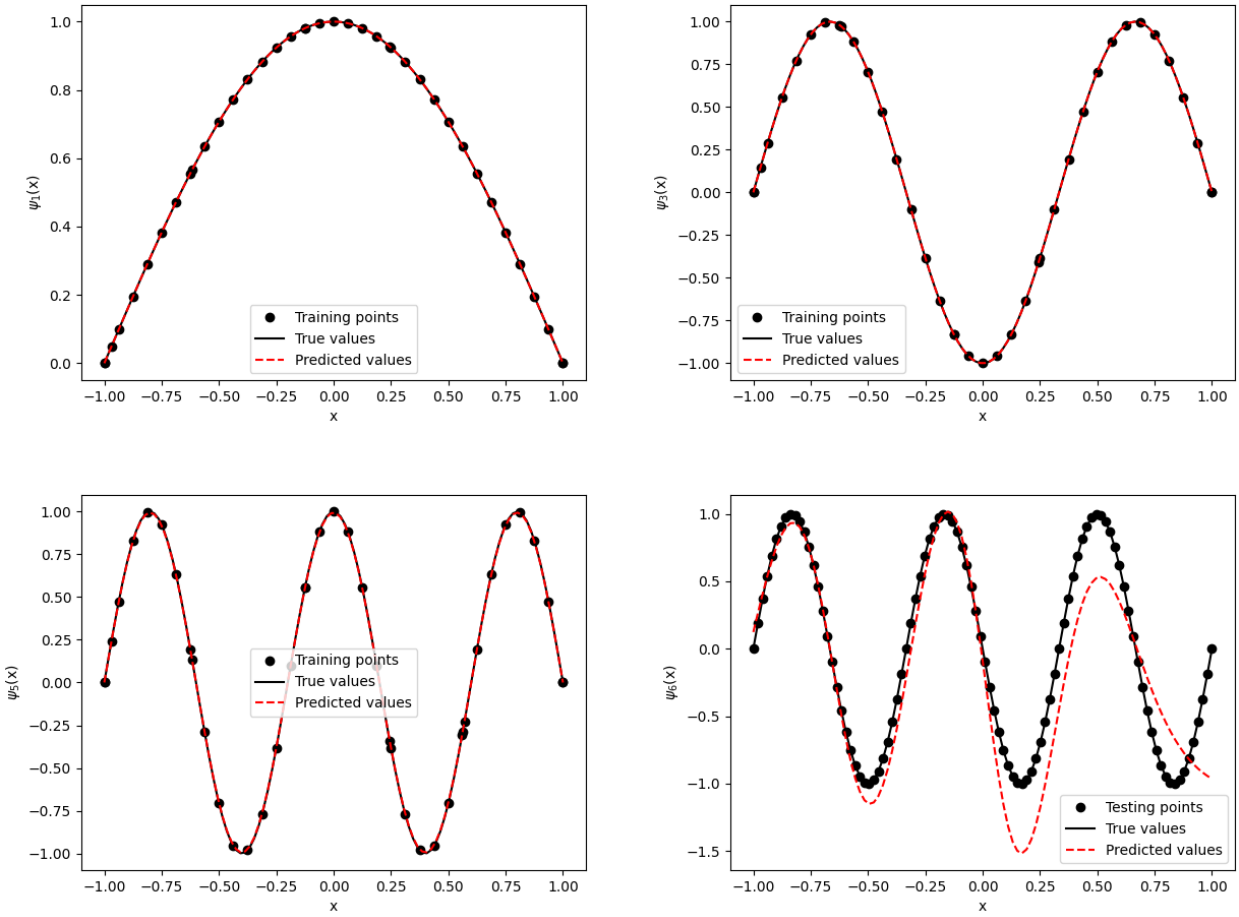


Figure 6.9: Examples of predictions made for one-dimensional infinite rectangular box: $n = 1$ (top left), $n = 3$ (top right), $n = 5$ (bottom left) and value from PINN with n as discrete value laying outside the training domain - $n = 6$ (bottom right).

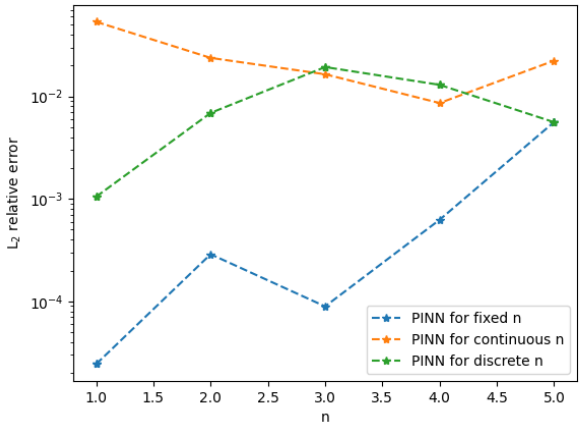


Figure 6.10: Dependence of the L_2 relative error on value of n for different approaches to training PINN

6.2.2 One-dimensional triangular box

The solution for this case involves the Airy function from the Scipy package for which automatic differentation has not been implemented. Thus, only training NNs for fixed quantum numbers n was possible. The considered n was in the range between 1 and 5.

The Schrödinger equation for this problem has the following form:

$$-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2\psi_n(x)}{\mathrm{d}x^2} + q\varepsilon x\psi_n(x) = E_n\psi_n(x). \tag{6.7}$$

For convenience, unit mass of the particle ($m = 1$), unit positive charge of the particle ($q = 1$) and unit electric field ($\varepsilon = 1$) were chosen for experiments in atomic units (where $\hbar = 1$). The maximum considered distance x_{max} was set to 2, so the problem domain was $x \in [0, 2]$. Equation 4.29 for the system energy E_n takes the form:

$$E_n = -\frac{\alpha_n}{\sqrt[3]{2}}, \tag{6.8}$$

where α_n is the n-th zero of the Airy's function of the first kind, Ai. The PDE to be solved takes the following form:

$$\frac{\mathrm{d}^2\psi_n(x)}{\mathrm{d}x^2} - 2\left(\frac{\alpha_n}{\sqrt[3]{2}} + x\right)\psi_n(x) = 0. \tag{6.9}$$

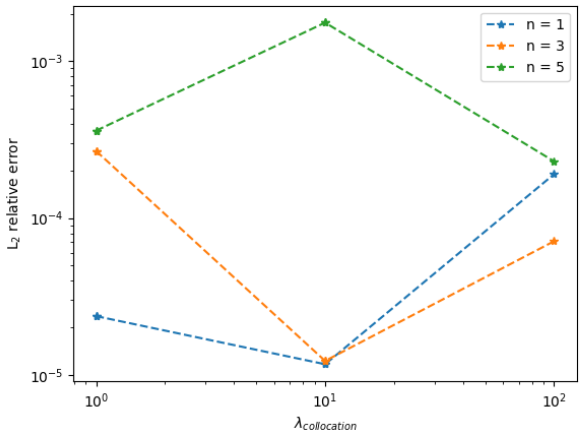


Figure 6.11: Dependence of the L_2 relative error on different weights $\lambda_{\text{collocation}}$ of collocation points for PINN solving Schrödinger equation for the particle in triangular potential box

Since the solution here does not vanish for positive finite values of x, the Dirichlet boundary condition is applied only to the beginning of the domain:

$$\psi_n(0) = 0. \tag{6.10}$$

Choosing a large value of x_{max} and using it as another Dirichlet boundary condition with vanishing $\psi_n(x)$ was not appropriate for this case, because it led to the prediction of solution identically equal 0. Instead of x_{max}, collocation points were used, chosen uniformly between 0 and x_{max} in a number 3 times greater than n. The same weight $\lambda_{\text{collocation}}$ was used again for collocation

points and Dirichlet boundary conditions.

First, different values of weight $\lambda_{\text{collocation}}$ were checked with the fixed number of layers (5) and nodes per layer (20). Results are presented in Figure 6.11. Value of 10 was chosen as the best compromise to minimize the error. However it is worth noting that even for $\lambda_{\text{collocation}}$ equal to 1 the errors were relatively low – below 10^{-3}.

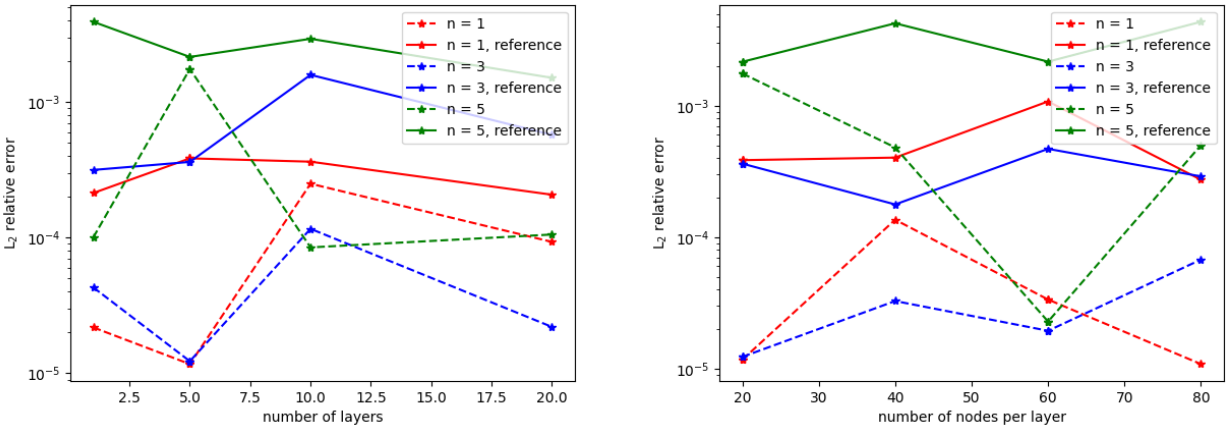


Figure 6.12: Dependence of the L_2 relative error on different number of layers 20 nodes each (left) and on different number of nodes for 5 layers (right). Values for PINN for the particle in the triangular box. Errors for fitting a function by a simple NN are shown as a reference

For chosen $\lambda_{\text{collocation}} = 10$, next studies to search for optimal values of the number of layers and the number of nodes per layer were performed. The results are shown in Figure 6.12. From the left panel the optimal number of layers was identified as 5, and then for this number of layers, the optimal number of nodes per layer was identified as 60 (right panel). For all shown examples, the errors of PINN prediction are lower than the reference errors.

Examples of predictions of the PINN for some chosen values of n are presented in Figure 6.13. The PINN was able to correctly predict the shape of the function, and the differences between the predicted and exact solution are not visible. Figure 6.14 shows dependence of the L_2 relative error on n for PINNs and the reference case. For the triangular potential box the PINN resulted in lower errors than a simple approximation of the function by NN.

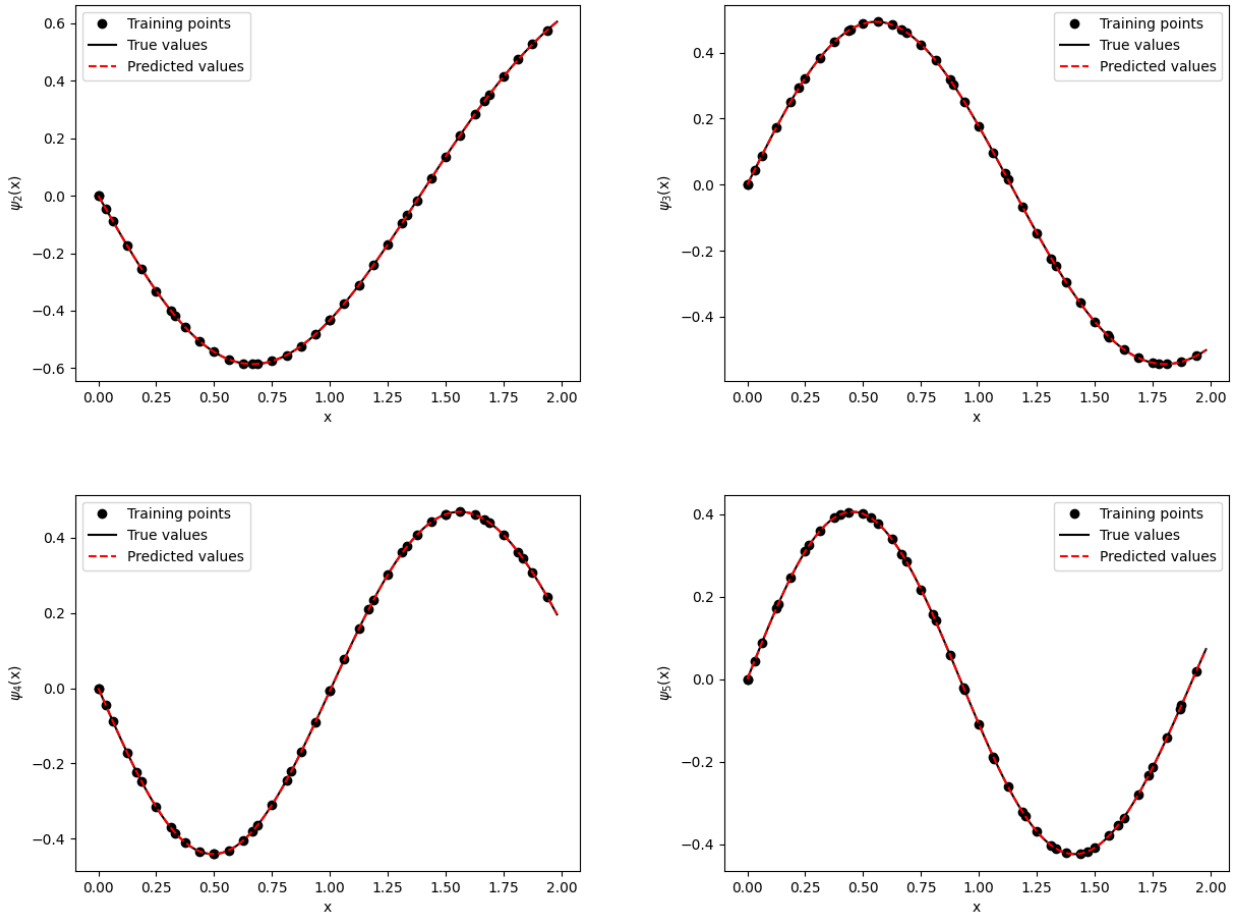


Figure 6.13: Examples of predictions made for one-dimensional triangular box: $n = 2$ (top left), $n = 3$ (top right), $n = 4$ (bottom left) and $n = 5$ (bottom right).

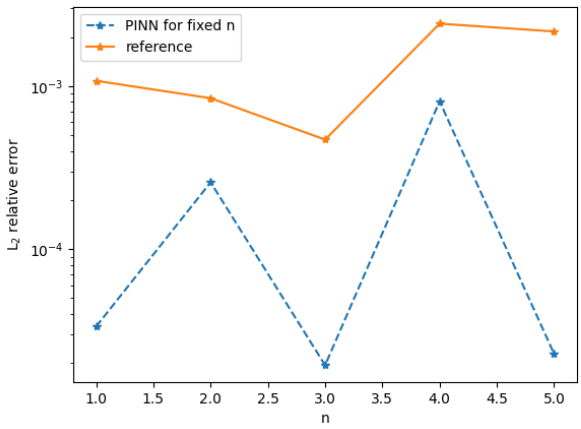


Figure 6.14: Dependence of the L_2 relative error on value of n for different approaches to training PINN

6.2.3 Two-dimensional rectangular box

In the case of a rectangular infinite box, the two-dimensional problem:

$$-\frac{\hbar^2}{2m}\Delta\Psi_{n_x,n_y}(x,y) = E_{n_x,n_y}\Psi_{n_x,n_y}(x,y), \tag{6.11}$$

is separable into two one-dimensional problems. The solution becomes a product of two one-dimensional solutions, as described in Equation 4.36. Thus, it can be predicted by two PINNs designed for one-dimensional problems (separately for n_x and n_y) solving Equation 6.4.

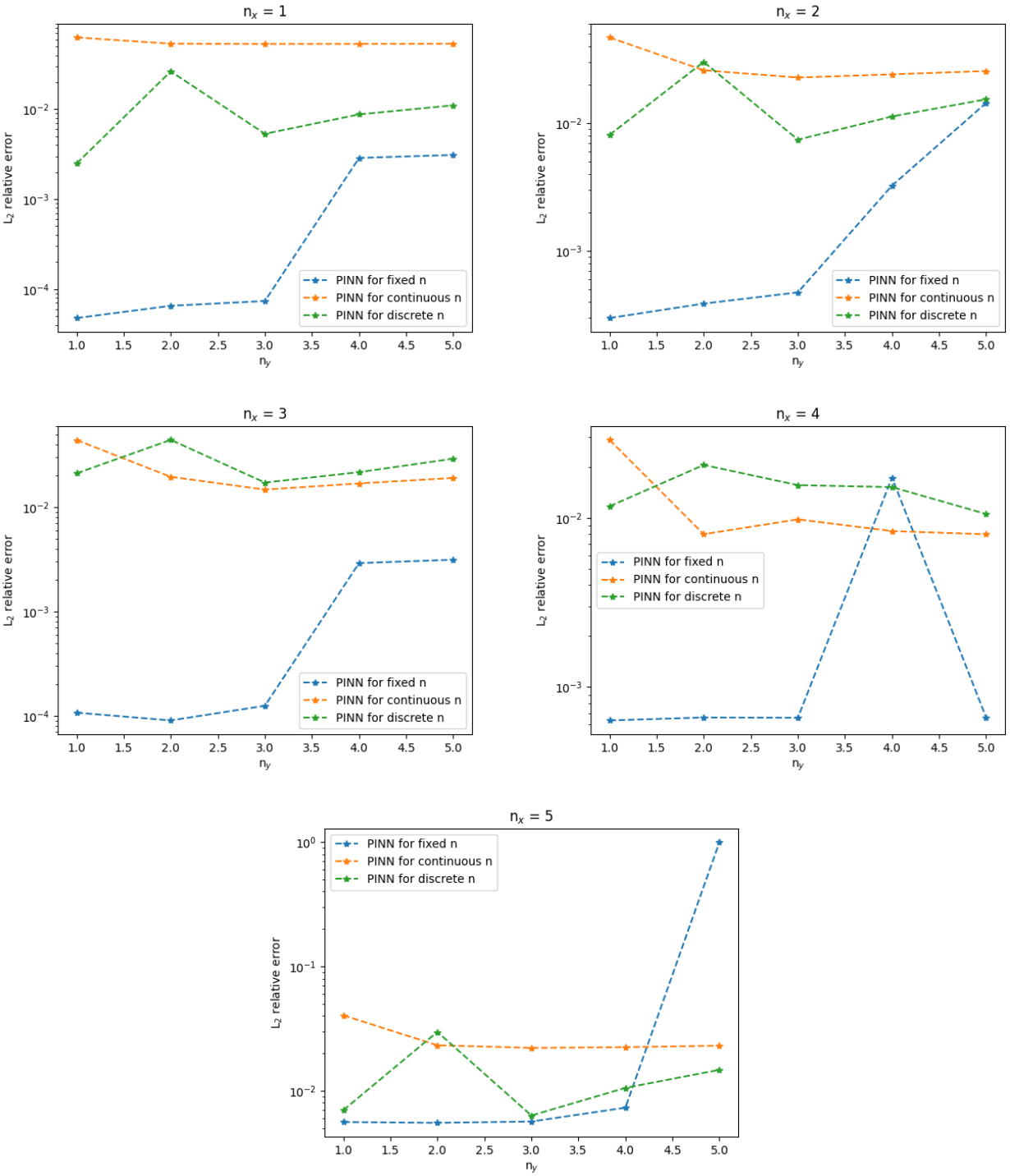


Figure 6.15: L_2 relative errors dependence on n_y for all studied approaches for fixed values of n_x

In this case, the mass of the particle is assumed to be unit ($m = 1$). The both lengths of the box edges are equal and $L_x = L_y = 2$. Thus, the domain of the problem is $[-1, 1] \times [-1, 1]$. All approaches using PINNs were used: training PINN for fixed n, for continuous n and for discrete n. The hyperparameters of training were the same as identified for one-dimensional case (listed in Table 6.2).

Figure 6.15 presents obtained errors for n_x and n_y values in the range from 1 to 5. Similarly to previous results, in most cases training PINN for fixed values of quantum numbers resulted in the lowest errors. However, with increasing values of n_y errors in all approaches become close to each other. In all cases the errors are acceptable and below 10^{-1} (usually below 10^{-2}). Only in one case (fixed $n_x = n_y = 5$) the error was close to 1. This may be the result of very low weight $\lambda_{\text{collocation}}$ for large quantum numbers (which is the standard difficulty for the case of vanishing potential).

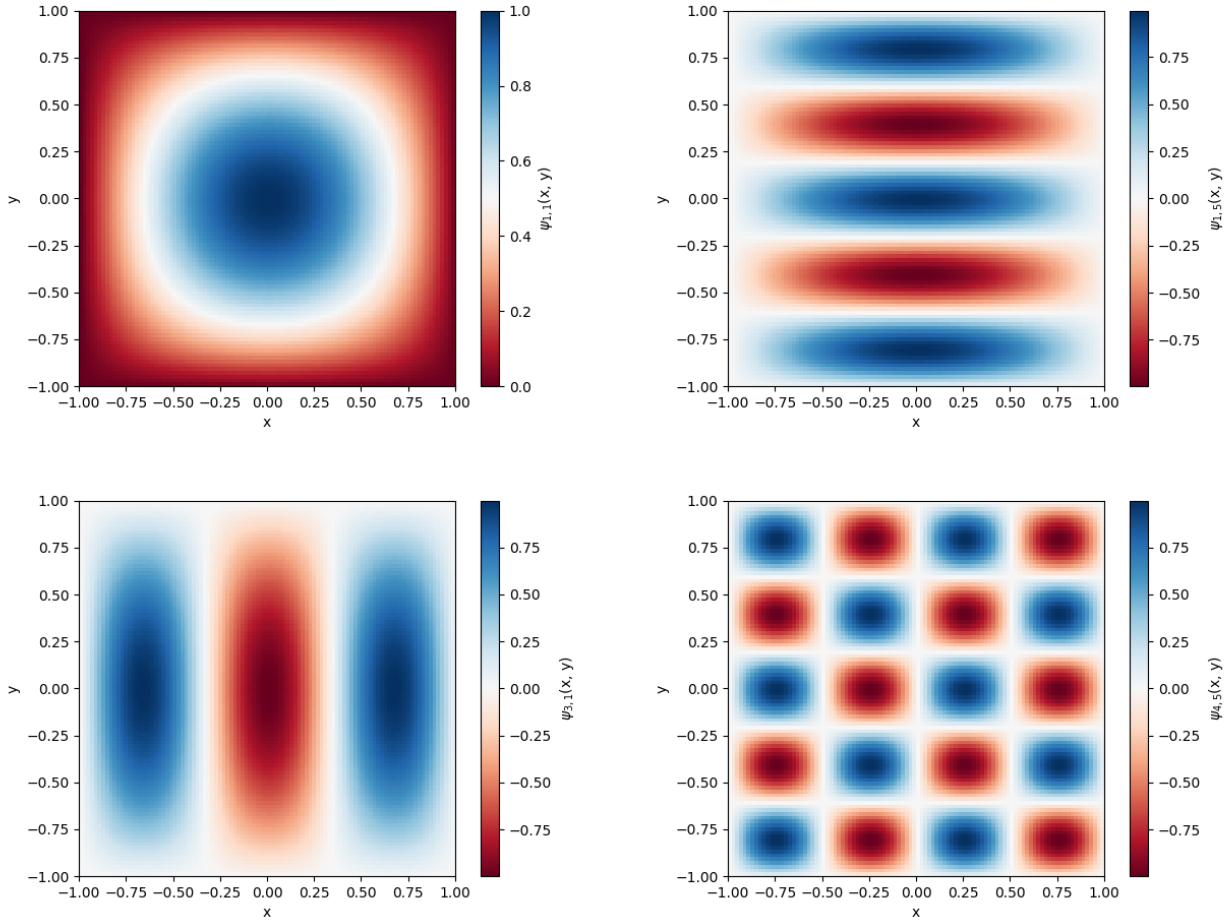


Figure 6.16: Examples of predictions made for two-dimensional rectangular box: $n_x = n_y = 1$ (top left), $n_x = 1, n_y = 5$ (top right), $n_x = 3, n_y = 1$ (bottom left) and $n_x = 4, n_y = 5$ (bottom right)

Figure 6.16 presents examples of predicted functions for chosen values of n_x and n_y.

6.2.4 Two-dimensional circular box

For this case only the radial part of the wavefunction was modeled (Equation 4.46) because the angular part of the solution (Equation 4.47) has complex values. Such functions are not implemented in DeepXDE.

The radial part of the Schrödinger equation has the following form:

$$\frac{\mathrm{d}^2 R_{nl}(r)}{\mathrm{d}r^2} + \frac{1}{r}\frac{\mathrm{d}R_{nl}(r)}{\mathrm{d}r} + \left(k^2 - \frac{l^2}{r^2}\right) R_{nl}(r) = 0. \tag{6.12}$$

Similarly to the case of the triangular one-dimensional box, only the approach with PINN trained for fixed values of quantum numbers n, l was possible to use. This was because for the Bessel functions from the Scipy package, automatic differentiation was not implemented.

Atomic units were used again with assumption that the particle has unit mass ($m = 1$) and the problem domain is a circle with radius equal to 1 ($a = 1$, according to Equation 4.40). After substitution of these values, and a simple manipulation of Equation 4.46 done to avoid division by small numbers, the PDE to be solved has the form:

$$r^2 \frac{\mathrm{d}^2 R_{nl}(r)}{\mathrm{d}r^2} + r\frac{\mathrm{d}R_{nl}(r)}{\mathrm{d}r} + \left(r^2\alpha_{nl}^2 - l^2\right) R_{nl}(r) = 0. \tag{6.13}$$

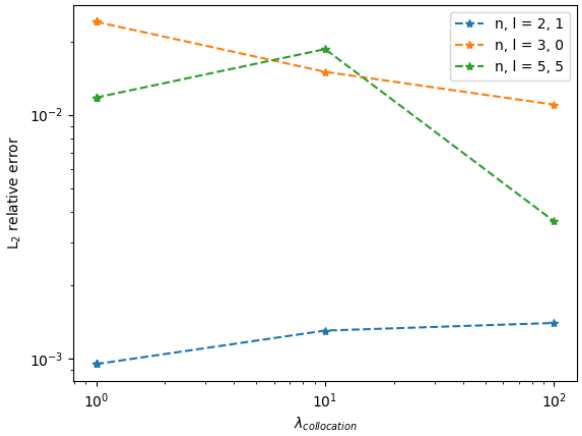


Figure 6.17: Dependence of the L_2 relative error on different weights $\lambda_{\text{collocation}}$ of collocation points for PINN with 5 layers, 60 neurons each, solving Schrödinger equation for the particle in circular potential box

To obtain the solution, Dirichlet boundary condition was applied to the edge of the domain:

$$R_{nl}(a) = 0, \tag{6.14}$$

along with Neumann boundary condition applied to the edge:

$$\frac{\mathrm{d}R_{nl}(r)}{\mathrm{d}r}\bigg|_{r=a} = \frac{\sqrt{2}\alpha_{nl}}{|J_{l+1}(\alpha_{nl})|} J_l'(\alpha_{nl}), \tag{6.15}$$

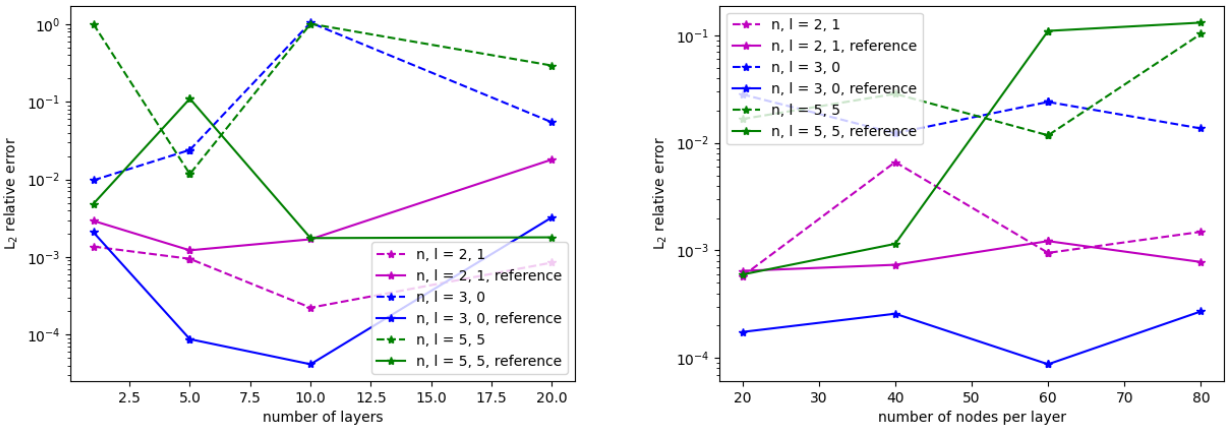


Figure 6.18: Dependence of the L_2 relative error on different number of layers, 60 nodes each (left) and on different number of nodes for 5 layers (right). Values for PINN for the particle in the circular box. Errors for fitting a function by a simple NN are shown as a reference

where $J_l^{'}$ is the first derivative of the Bessel function J_l. In addition, collocation points chosen uniformly from problem domain in the number of $3 \cdot (n + l)$ were used. All of these additional conditions had the same weight, denoted as $\lambda_{\text{collocation}}$. Training was done using 64 training points.

Errors for different weights $\lambda_{\text{collocation}}$ were studied for some chosen values of n and l and for PINNs with 5 layers, 60 neurons each. The results are presented in Figure 6.17. For all cases shown, the errors were rather low even for $\lambda_{\text{collocation}}$ equal to 1, so this value was chosen for further experiments as more natural, without highlighting any conditions as more important than the others.

For chosen value of $\lambda_{\text{collocation}}$, a study into the dependence of the error on the number of layers and the number of nodes per layer was done. Results are shown in Figure 6.18. Five layers of 60 neurons each were used as error-minimising parameters for all cases presented. In this case, errors lower than in the reference method were not obtained in all cases. It seems that increasing values of n complicate the problem of solving the PDE. Summary of hyperparameters choice for triangular and circular potential boxes is presented in Table 6.3. It should also be highlighted that satisfactory results for this case were obtained when the number of training points was increased from 32 to 64.

Table 6.3: Optimal hyperparameters of models for triangular and circular infinite box

case	N_{layers}	N_{nodes}	N_{train}	N_{test}	N_{boundary}	$\lambda_{\text{collocation}}$	epochs
triangular box	5	60	32	100	1	10	10000
circular box	5	60	64	100	1	1	10000

The results shown in Figure 6.19 confirm the above expectations. For $n = 1$ PINN gives a lower error than the reference case, for $n = 2$ this relation depends on the value of l, while for higher values of n the reference error is lower than in the method using PINN. However, as for the previous cases, the errors are relatively low and acceptable.

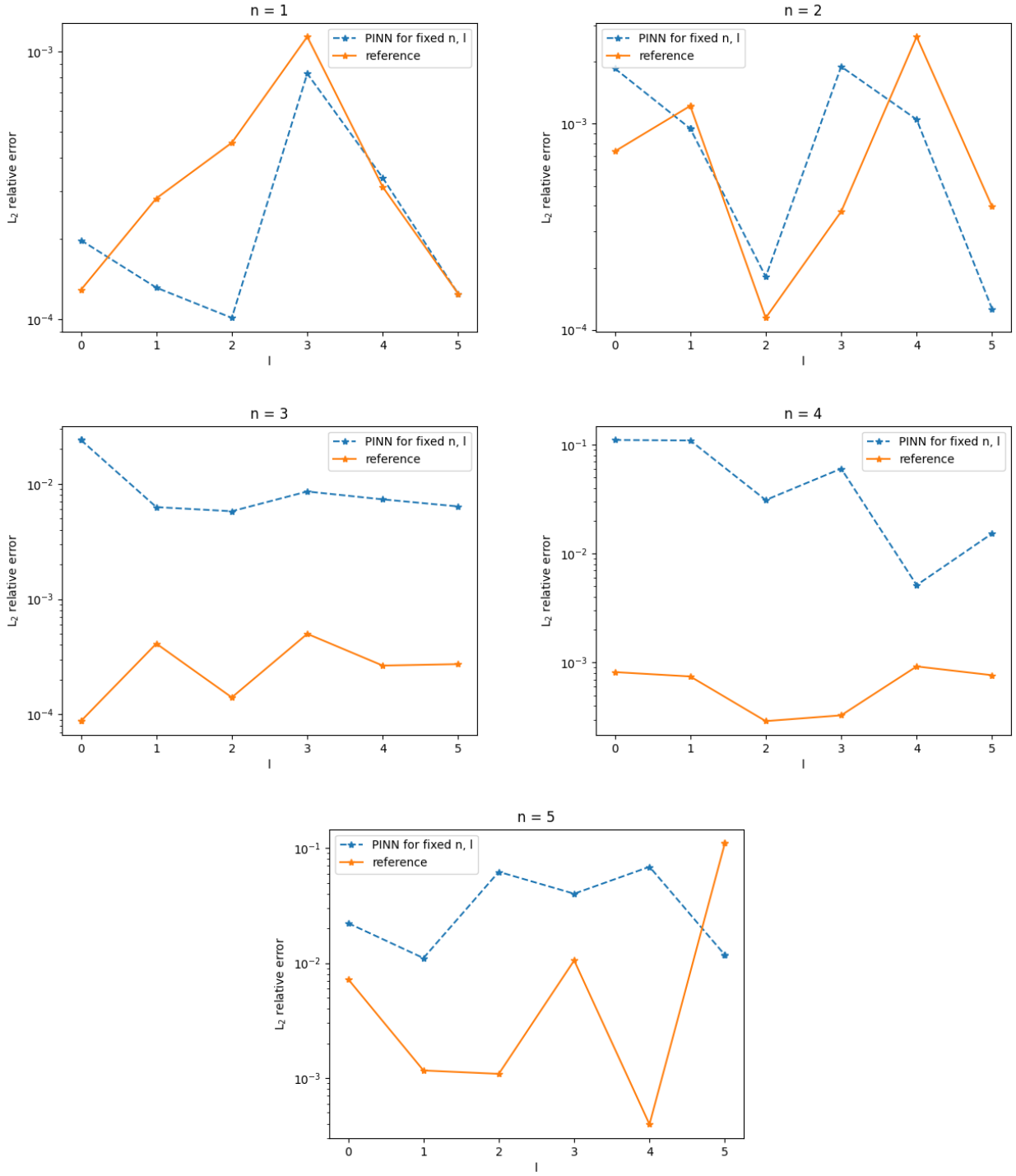


Figure 6.19: L_2 relative error dependence on l for all studied approaches for fixed values of n

Examples of functions predicted for chosen values of n and l are presented in Figure 6.20.

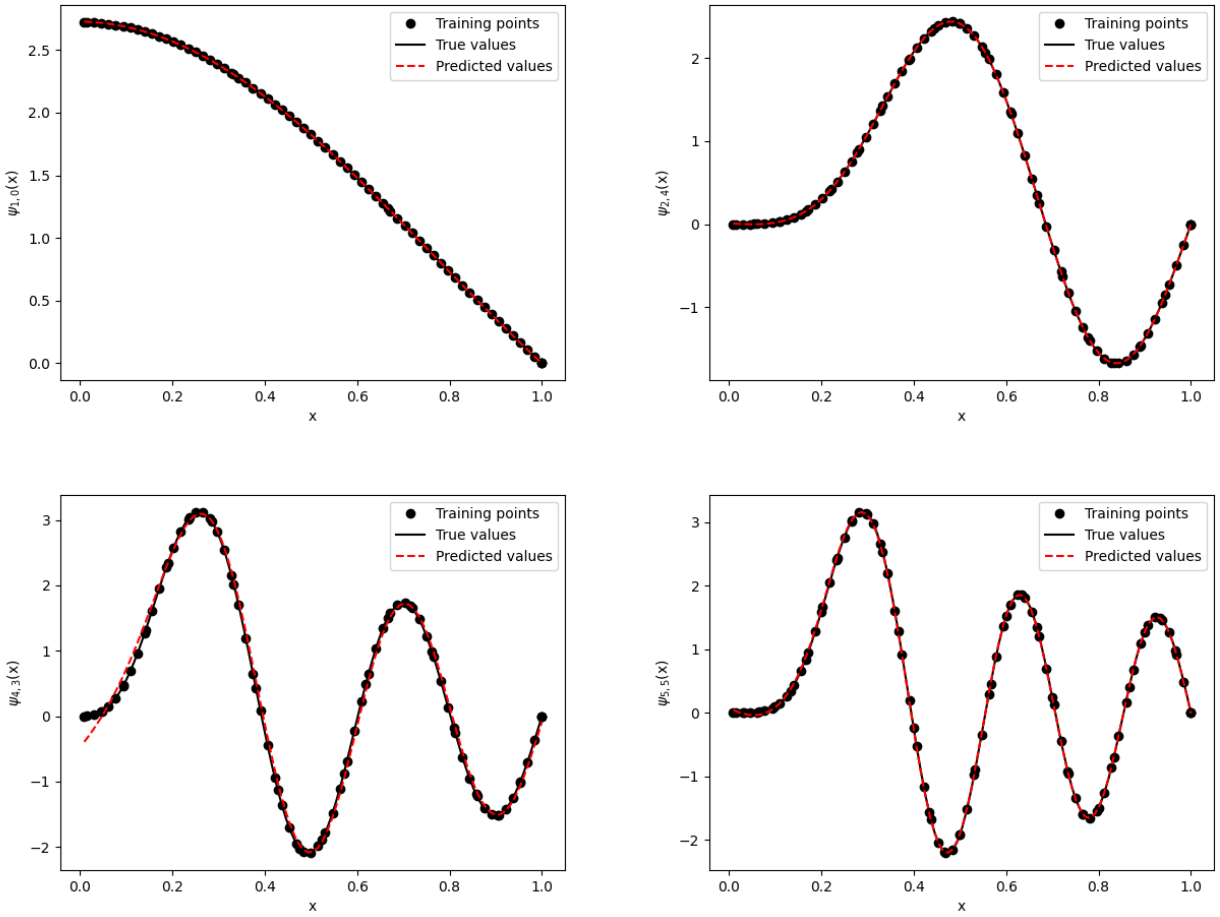


Figure 6.20: Examples of predictions made for radial part of a solution for two-dimensional circular box: $n = 1, l = 0$ (top left), $n = 2, l = 4$ (top right), $n = 4, l = 3$ (bottom left) and $n = 5, l = 5$ (bottom right)

6.2.5 One-dimensional harmonic oscillator

The Schrödinger equation for one-dimensional harmonic oscillator has the following form:

$$-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2\psi_n(x)}{\mathrm{d}x^2}+\frac{1}{2}m\omega^2x^2\psi_n(x)=E_n\psi_n(x). \tag{6.16}$$

In the case of the quantum harmonic oscillator, atomic units were used with the unit mass of the oscillator ($m=1$) and its frequency ω equal to 0.5 (symbols are the same as in Equation 4.51). After substitution of these values into Equation 4.51, the potential has the following form:

$$V(x)=\frac{1}{8}x^2. \tag{6.17}$$

Substituting this potential and the energy of the oscillator (Equation 4.55) into the Schrödinger Equation 6.16 leads to the form of the PDE:

$$-\frac{\mathrm{d}^2\psi_n(x)}{\mathrm{d}x^2}+\left(\frac{1}{4}x^2-n-\frac{1}{2}\right)\psi_n(x)=0. \tag{6.18}$$

For the quantum harmonic oscillator the wavefunction vanishes in the limit of infinite values of x and the problem domain for computation had to be limited. The maximum considered absolute value of x (denoted as L) was chosen to be equal to 5.

To solve this equation, boundary conditions at $x=0$ were applied. The Dirichlet condition forces that:

$$\psi_n(0)=\frac{1}{\sqrt{2^n n!}}\left(\frac{1}{2\pi}\right)^{\frac{1}{4}}H_n(0). \tag{6.19}$$

For odd values of n, the properties of Hermite's polynomials cause that $\psi_n(0)=0$.

After differentiation of Equation 4.53, and using the property of Hermite's polynomials:

$$\frac{\mathrm{d}H_n(x)}{\mathrm{d}x}=2nH_{n-1}(x), \tag{6.20}$$

the Neumann's boundary condition at $x=0$ forces that:

$$\frac{\mathrm{d}\psi_n(x)}{\mathrm{d}x}\Big|_{x=0}=\frac{2n}{\sqrt{2^n n!}}\frac{1}{\sqrt[4]{8\pi}}H_{n-1}(0). \tag{6.21}$$

For even values of n this derivative is equal to 0.

For this model problem all of the approaches using PINNs were possible to be applied. In all of them the range of considered values of n was between 0 and 5. Furthermore, the weights for all parts of the loss function were the same and equal to 1. The initial hyperparameters for PINNs were chosen to be the same as for the one-dimensional infinite potential well (Table 6.2) without weights. The modeled functions are more complicated than simple trigonometric functions. Thus, as an initial guess, the number of training points was increased to 64 (similarly to the case with the circular box radial function).

PINN trained on PDE with fixed n

In this approach, a modification of boundary conditions needed to be done. For odd values of n function $\psi_n(0)$ is equal to 0, and in this case it led to the prediction of a function close to the function identically equal zero, $\psi(x) = 0$. To avoid this behaviour, for odd n, instead of point $x = 0$, points $x = \pm\frac{L}{2}$ were used for Dirichlet boundary conditions.

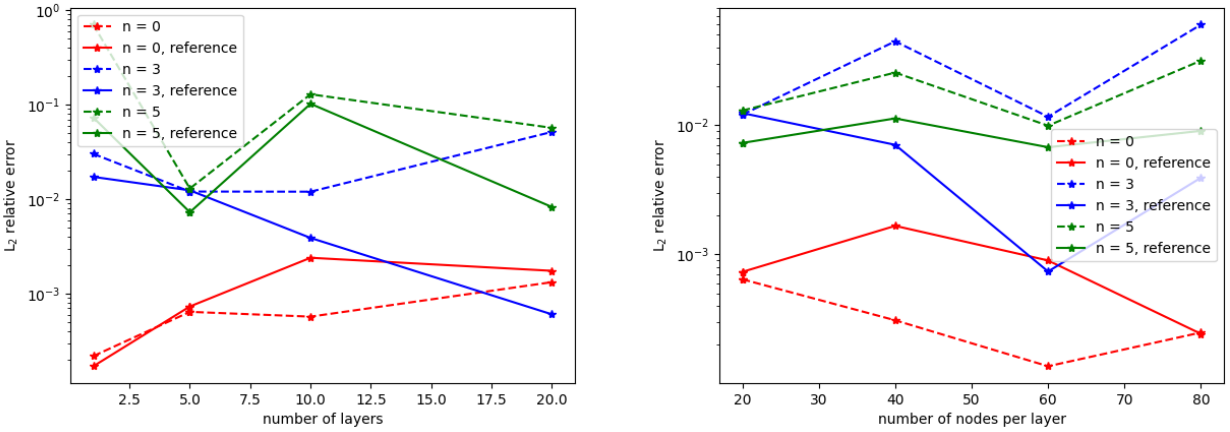


Figure 6.21: Dependence of the L_2 relative error on different number of layers, 20 nodes each (left) and on different number of nodes for 5 layers (right). Values for PINN for the one-dimensional harmonic oscillator. Errors for fitting a function by a simple NN are shown as a reference

As cases for research on the optimal hyperparameters of PINN, values of n equal to 0, 3 and 5 were chosen, as the training task was more difficult for the odd numbers. For the fixed number of nodes per layer (20), the error dependence on the number of layers is shown in the left panel of Figure 6.21. As the value minimizing the error for all plotted cases, and resulting in lower or similar error as the reference method, 5 layers were chosen. For this number of layers on the right panel of Figure 6.21 error dependence on number of nodes per layers is shown. From this relation, 60 nodes per layer were chosen as the optimal value. The errors obtained for the reference method were lower for odd values of n, and higher for $n = 0$ compared to those obtained by PINN. However, for PINN they are acceptable and lower than 10^{-2}.

PINN trained on PDE with n as a continuous variable

For the case of treating n as the second input with continuous values, some modifications needed to be done in order to implement this approach. Hermite polynomials H_n are defined only for integer values of n. Thus, here for any n the order of Hermite polynomial is assumed to be the floor of n:

$$H_n(x) \approx H_{\lfloor n \rfloor}, \quad n \in [0, 5]\,. \tag{6.22}$$

Furthermore, the solution (Equation 4.53) involves using factorial of n, which is defined only for natural numbers. To avoid this limitation, the Euler's gamma function $\Gamma(x)$ was used instead,

with the argument increased by 1, due to its property:

$$\Gamma(n+1) = n!, \quad \text{for } n \in \mathbb{N}. \tag{6.23}$$

The reference solution in training PINN in this approach has the following form:

$$\psi(x,n) = \frac{1}{\sqrt{2^n \Gamma(n+1)}} \left(\frac{1}{2\pi}\right)^{\frac{1}{4}} e^{-\frac{x^2}{4}} H_{\lfloor n \rfloor}\left(\frac{x}{\sqrt{2}}\right). \tag{6.24}$$

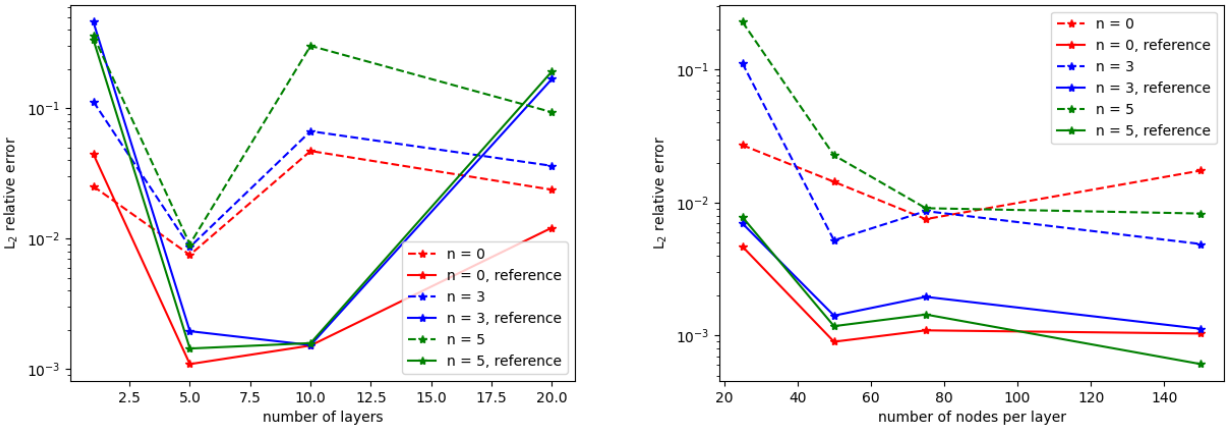


Figure 6.22: Dependence of the L_2 relative error on different number of layers, 75 nodes each (left) and on different number of nodes for 5 layers (right). Values for PINN for the one-dimensional harmonic oscillator using n as a continuous variable. Errors for fitting a function by a simple NN are shown as a reference

Boundary conditions are the same as for the case with fixed n, and they are defined for points (x, n) where $x = 0$ and $n \in \mathbb{N}$.

The results of the tests done for choice of optimal hyperparameters of learning the PINN are shown in Figure 6.22. From them, 5 layers were chosen with 75 nodes each as the values minimizing the error. Here, the errors were larger than for the reference method, however they were in the acceptable order of magnitude.

PINN trained on PDE with n as a discrete variable

In this case there was no need to modify the form of the reference solution for the PINN. The boundary conditions were the same as for the case with fixed n, and were applied to points with $x = 0$.

Analysis of the error dependence on the number of layers and number of nodes per layer is shown in Figure 6.23. Five layers of 60 nodes each were chosen as the values minimizing errors for studied cases. Here, for $n = 5$ or $n = 0$, the errors are below or close to the errors in reference method, while for $n = 3$ are larger. For all of these cases their values were below 10^{-2}.

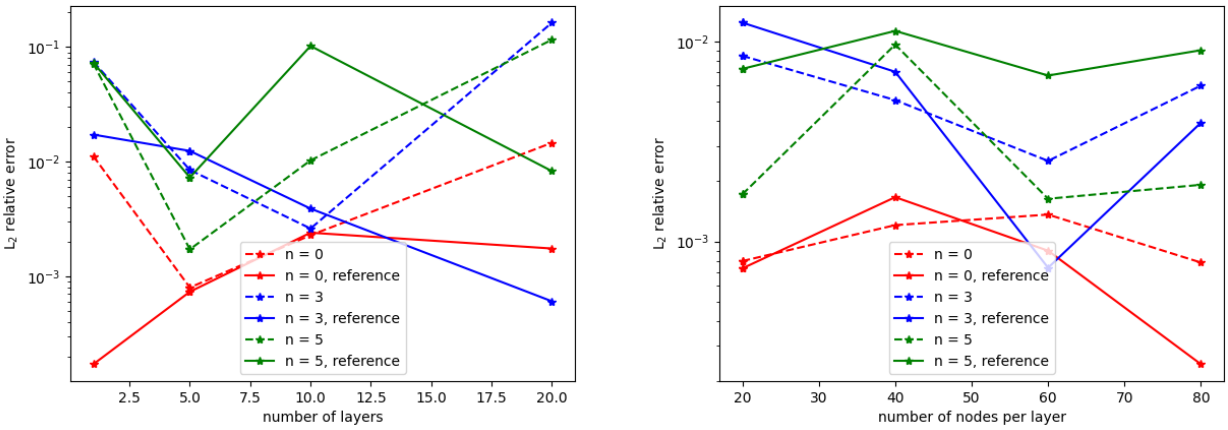


Figure 6.23: Dependence of the L_2 relative error on different number of layers, 20 nodes each (left) and on different number of nodes for 5 layers (right). Values for PINN for the one-dimensional harmonic oscillator using n as a discrete variable. Errors for fitting a function by a simple NN are shown as a reference

Summary

The hyperparameters found for different approaches are presented in Table 6.4. Compared to the values for one-dimensional potential well (Table 6.2) for the case of a harmonic oscillator, all of them have the same number of layers. For fixed or discrete n approaches, the number of nodes per layer is larger than for the potential well, while for continuous n it is smaller. However, for the continuous n, the number of training points was 4 times larger, so probably this allowed a lower number of nodes. These results suggest that a more complicated form of the predicted solution needs a larger number of training points and nodes to catch all of the necessary features.

Table 6.4: Optimal hyperparameters of training models for one-dimensional harmonic oscillator

approach	N_{layers}	N_{nodes}	N_{train}	N_{test}	N_{boundary}	epochs
PDE, fixed n	5	60	64	100	2	10000
PDE, continuous n	5	75	4096	10000	12	10000
PDE, discrete n	5	60	384	600	12	10000

The dependence of the L_2 relative error on n for different approaches is presented in Figure 6.24. For fixed n the error grows with increasing n, while for approaches treating n as a variable the error remains approximately constant, with a single outlier for continuous n for $n = 1$. For discrete n, the changes are much smaller. In all cases (except the outlier for continuous n) the errors are of the acceptable order of magnitude and have values below 10^{-2}. The changes for fixed n approach may be caused by the previously mentioned problems with fitting the solution for odd values of n.

Sample functions predicted by PINNs, with one prediction made for $n = 6$ outside the learning domain for the continuous n approach are presented in Figure 6.25. It can be observed that this prediction is in a good qualitative agreement with the true solution, with incorrect values at the boundaries of the problem domain.

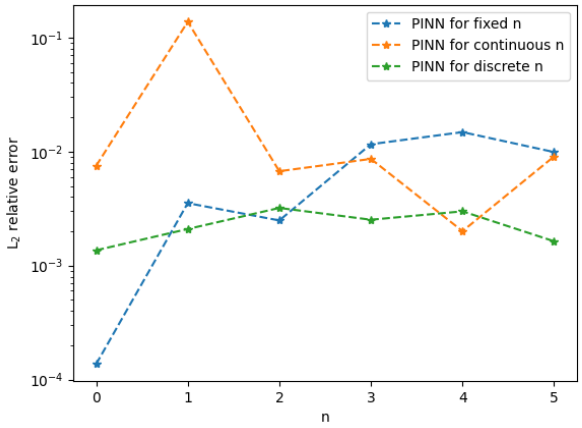


Figure 6.24: Dependence of the L_2 relative error on value of n for different approaches of training PINN

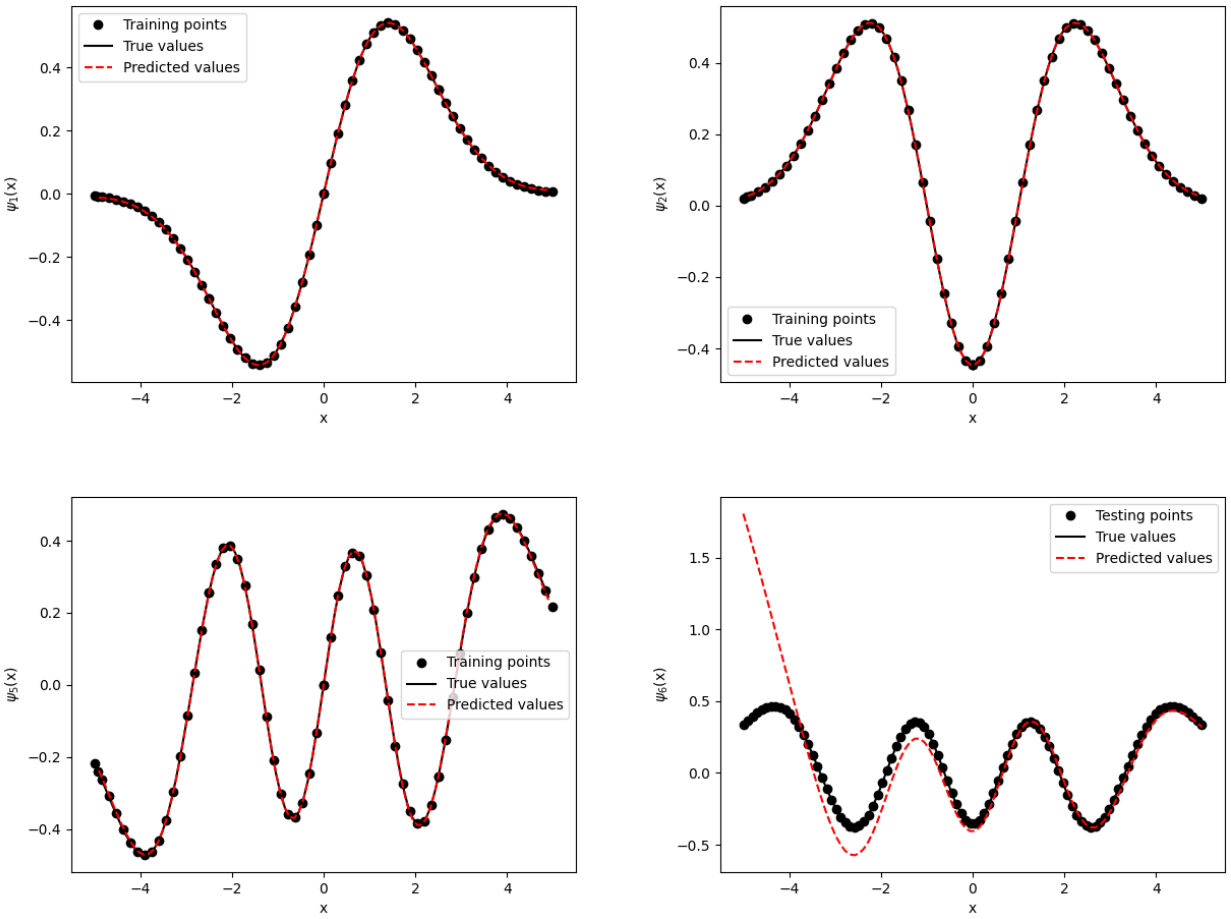


Figure 6.25: Examples of predictions made for one-dimensional harmonic oscillator: $n = 1$ (top left), $n = 2$ (top right), $n = 5$ (bottom left) and $n = 6$ (bottom right)

6.2.6 Two-dimensional harmonic oscillator

The problem of two-dimensional harmonic oscillator has the following form of the potential energy function $V(x, y)$:

$$V(x, y) = \frac{1}{2}m\omega_x^2 x^2 + \frac{1}{2}m\omega_y^2 y^2. \tag{6.25}$$

The Schrödinger equation for this problem has the following form:

$$-\frac{\hbar^2}{2m}\Delta\Psi_{n_x,n_y}(x, y) + \frac{1}{2}\left(m\omega_x^2 x^2 + m\omega_y^2 y^2\right)\Psi_{n_x,n_y}(x, y) = E_{n_x,n_y}\Psi_{n_x,n_y}(x, y). \tag{6.26}$$

In atomic units for a particle of unit mass ($m = 1$) and frequencies equal $\omega_x = \omega_y = 0.5$:

$$\left(-\frac{1}{2}\Delta + \frac{1}{8}x^2 + \frac{1}{8}y^2\right)\Psi_{n_x,n_y}(x, y) = E_{n_x,n_y}\Psi_{n_x,n_y}(x, y), \tag{6.27}$$

the equation is separable into two independent equations of one-dimensional harmonic oscillator after substitution:

$$\Psi_{n_x,n_y}(x, y) = \psi_{n_x}(x)\psi_{n_y}(y). \tag{6.28}$$

For factors of function Ψ_{n_x,n_y} the PDEs to be solved have the form of Equation 6.18 after substitution $\omega_x = \omega_y = 0.5$. Thus, this problem was solved similarly as for the case of two-dimensional rectangular potential well, i.e., by training two PINNs for one-dimensional problems for predicting of the x and y part of the solution respectively. All of the approaches described for the one-dimensional case were used with the hyperparameters found to be optimal in the preceding subsection (Table 6.4). The range of n_x and n_y was from 0 to 5.

Figure 6.26 shows L_2 relative errors dependence on n_y for fixed n_y values. In all cases, the highest errors were obtained for the approach with n as the discrete variable. For $n_x = 0$ the lowest errors were obtained for fixed n approach, while for other values of n_x the lowest errors were obtained from the approach with n as a continuous variable. However, for all of these cases errors for these two approaches were similar. For approaches treating n as a variable, the errors remain approximately constant for the whole range of examined values of n_y. These effects are probably caused by cumulations of errors after multiplication of predicted solutions from PINNs, and also from difficulties in modeling odd and even n cases for fixed values of n_x and n_y. Examples of predicted functions are presented in Figure 6.27.

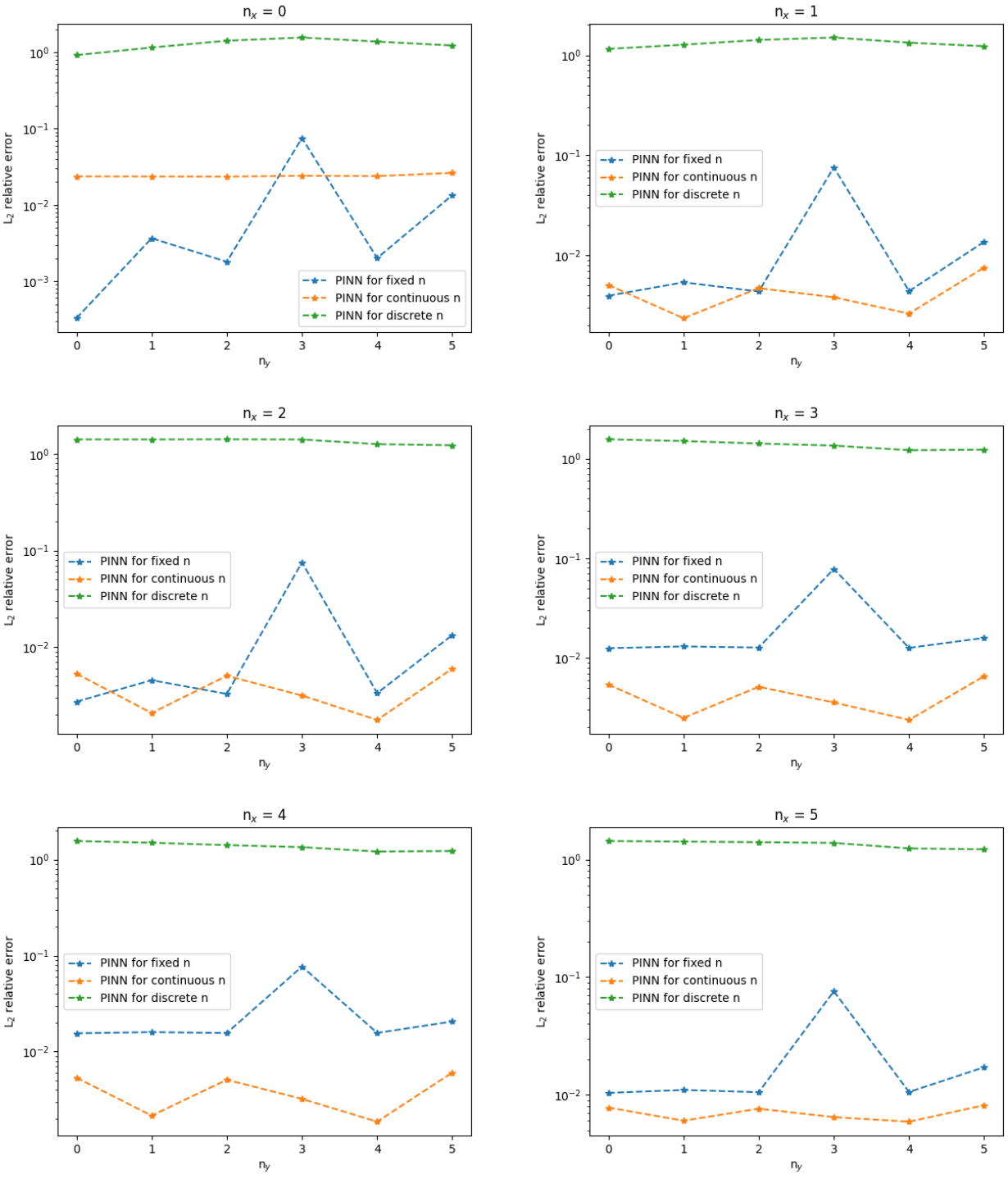


Figure 6.26: L_2 relative errors dependence on n_y for all studied approaches for fixed values of n_x

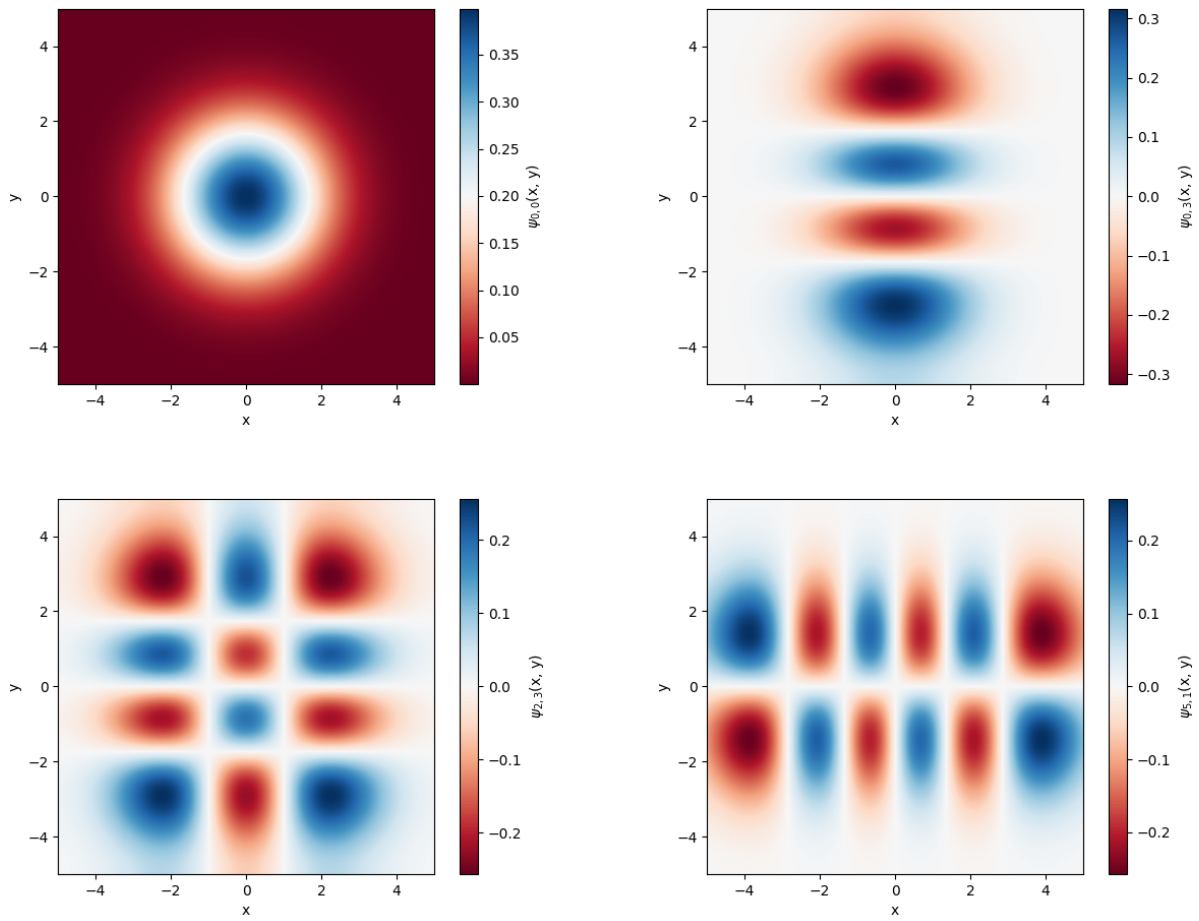


Figure 6.27: Examples of predictions made for two-dimensional harmonic oscillator: $n_x = n_y = 0$ (top left), $n_x = 0, n_y = 3$ (top right), $n_x = 2, n_y = 3$ (bottom left) and $n_x = 5, n_y = 1$ (bottom right)

6.2.7 Rigid rotor

For rigid rotor, the horizontal part of the Schrödinger equation has the following form:

$$\frac{\sin\vartheta}{\Theta(\vartheta)}\frac{\partial}{\partial\vartheta}\left(\sin\vartheta\frac{\partial\Theta(\vartheta)}{\partial\vartheta}\right)+\frac{2IE}{\hbar^2}\sin^2\vartheta=m^2. \tag{6.29}$$

This case involves associated Legendre polynomials in the horizontal part of the solution (Equation 4.61). Implementation of Legende polynomials was taken from the Scipy package and automatic differentiation was not implemented for them. Because of that, only the approach using fixed quantum numbers (l and m) was possible to apply. Furthermore, the azimuthal part of the solution (Equation 4.60) has complex values, and such functions are not implemented in DeepXDE. For this reason, only the horizontal part of the solution was modeled by PINNs.

The domain of the problem was the whole range of possible values of horizontal angle ϑ in the spherical coordinate system, i.e., $\vartheta\in[0,\pi]$. Considered values of quantum numbers were from 1 to 5 for l, and for every l values of m were from 0 to l.

The PDE for the horizontal part (Equation 4.59) was manipulated to avoid division by small numbers, and was used in form:

$$\sin^2\vartheta\frac{\partial^2\Theta(\vartheta)}{\partial\vartheta^2}+\sin\vartheta\cos\vartheta\frac{\partial\Theta(\vartheta)}{\partial\vartheta}+\left(l(l+1)\sin^2\vartheta-m^2\right)\Theta(\vartheta)=0. \tag{6.30}$$

As boundary conditions, uniformly distributed collocation points were used in the number of $3\cdot(l+m)$ with weight $\lambda_{\text{boundary}}=1$. After initial tests, it was decided that 64 training points were insufficient to properly model solutions for possible values of m for $l=5$, thus the number of training points was increased to 128.

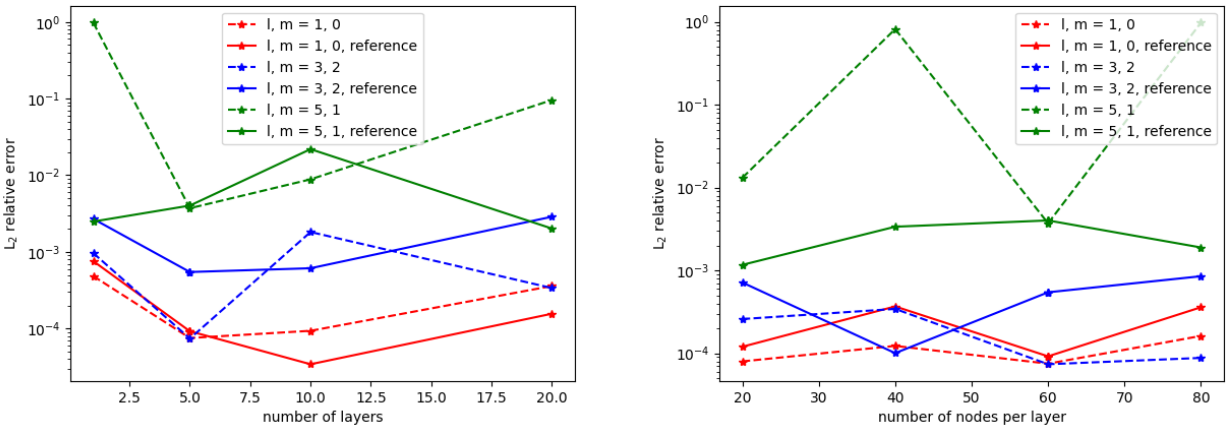


Figure 6.28: Dependence of the L_2 relative error on different number of layers, 60 nodes each (left) and on different number of nodes for 5 layers (right). Values for PINN for the rigid rotor. Errors for fitting a function by a simple NN are shown as a reference

The error dependence on the number of layers in the PINN was studied, and results are shown in the left panel of Figure 6.28. The minimal values of errors were observed for 5 layers. For this chosen number of layers, the dependence of the error on the number of nodes per layer was

examined (right panel of Figure 6.28) and the optimal value was chosen as 60. These parameters resulted in lower errors than for the reference method.

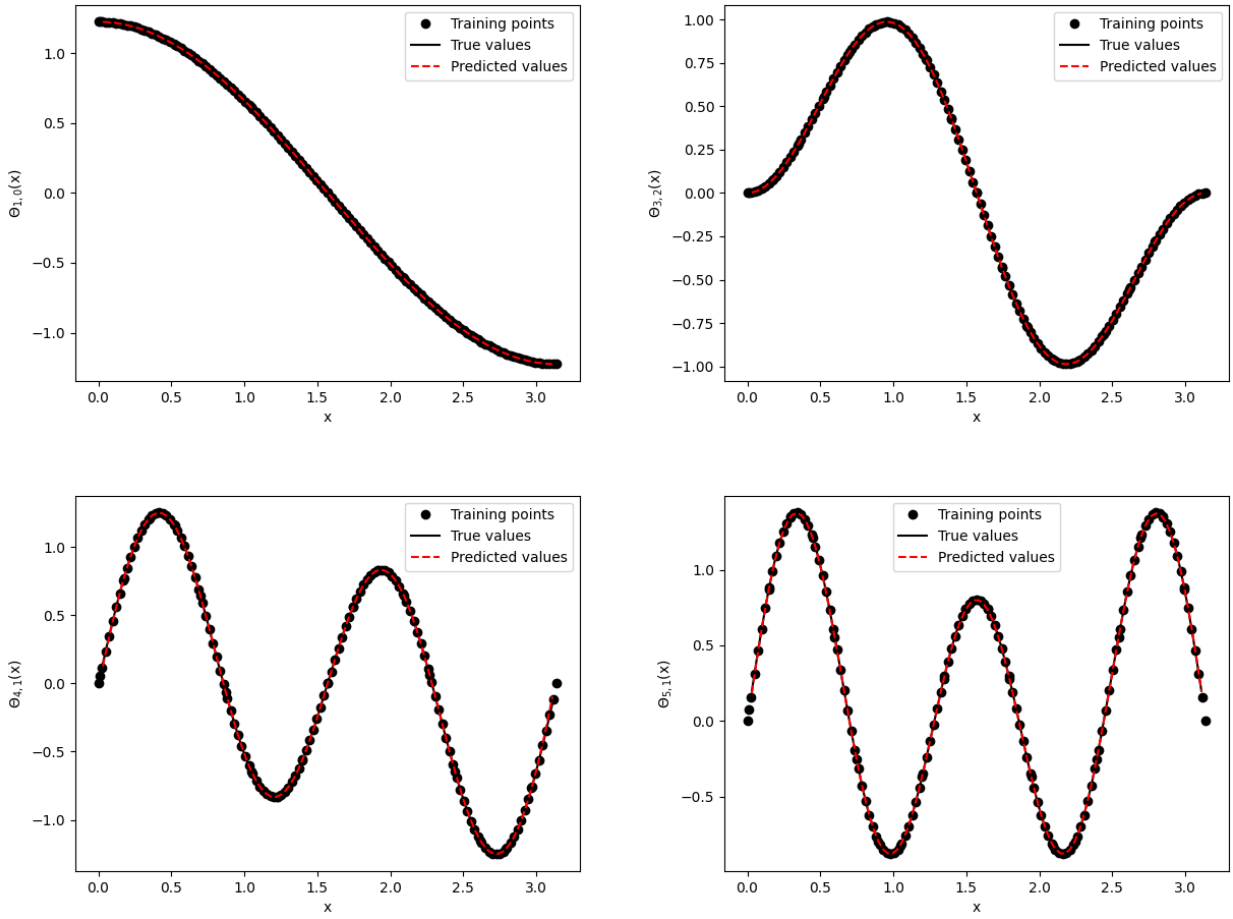


Figure 6.29: Examples of predictions made for rigid rotor: $l = 1, m = 0$ (top left), $l = 3, m = 2$ (top right), $l = 4, m = 1$ (bottom left) and $l = 5, m = 1$ (bottom right)

The error dependence on m for studied values of l is presented in Figure 6.30. For maximal values of m, the error of PINN prediction is lower than that of the reference method, for lower values of m these two errors are close, and for the lowest values of m the error of PINN is higher. This is probably the result of larger oscillations of the resulting solution for low values of m. Examples of predictions made by PINN are presented in Figure 6.29.

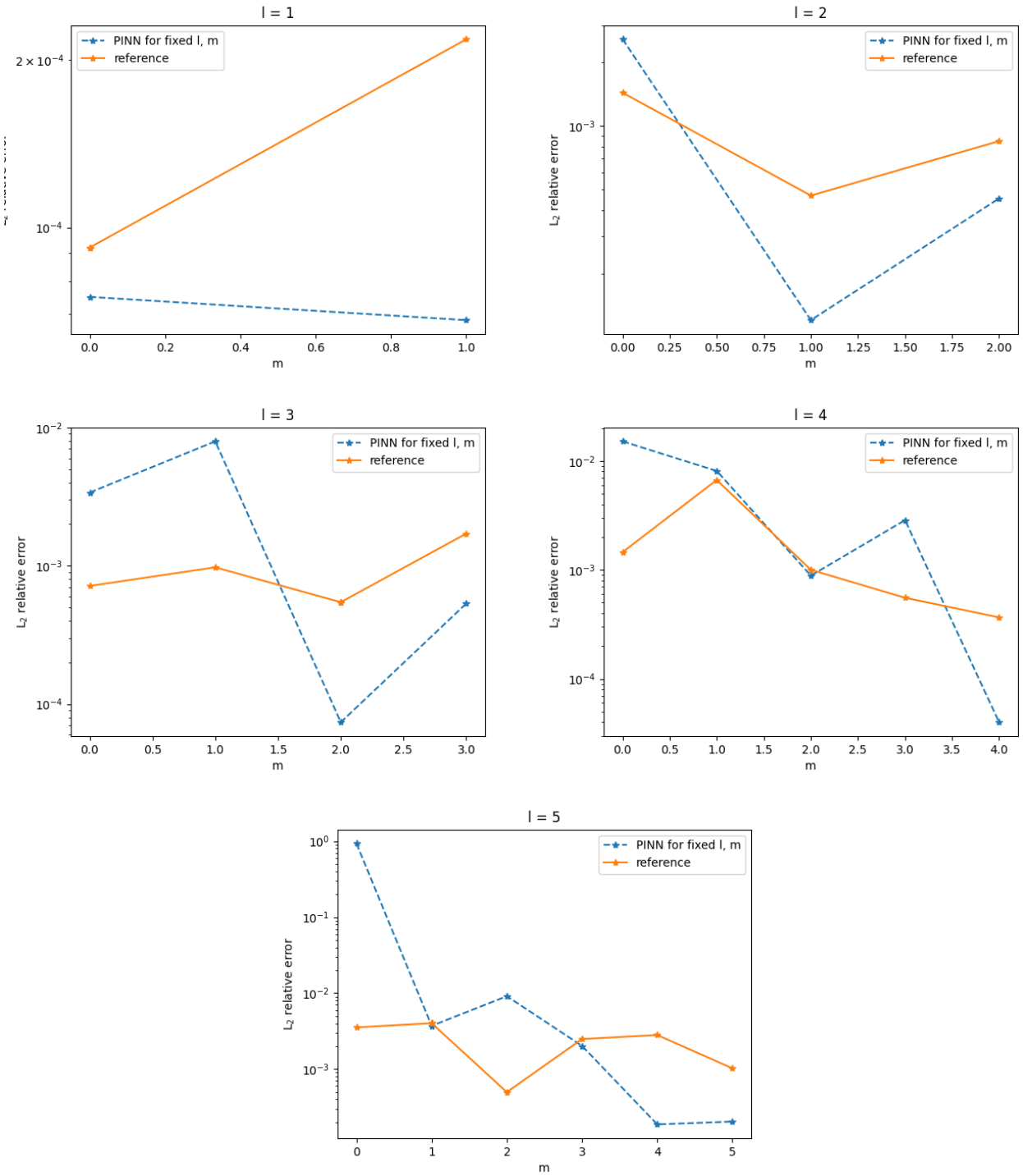


Figure 6.30: L_2 relative errors dependence on m for rigid rotor for fixed values of l

6.2.8 Hydrogen atom

For hydrogen atom, the radial part of the Schrödinger equation has the following form:

$$\left[-\frac{1}{r^2}\frac{\mathrm{d}}{\mathrm{d}r}\left(r^2\frac{\mathrm{d}}{\mathrm{d}r}\right)+2\mu\left(V(r)-E\right)+\frac{l(l+1)}{r^2}\right]R_{nl}(r)=0. \tag{6.31}$$

In the solution of the Schrödinger equation for a hydrogen-like atom, the radial part (previous equation) involves generalized Laguerre polynomials from the Scipy package. For them, automatic differentiation is not implemented, so only the approach with fixed quantum numbers was possible to use in this case. In addition, the angular part (Equation 4.77) is the function for the rigid rotor described in the previous subsection. Thus, here only the radial part of the problem is solved.

After taking the charge of nucleus $Z = 1$ and the unit reduced mass of the system $\mu = 1$, in atomic units, and a simple manipulation to avoid division by small numbers, the PDE of Equation 4.70 takes the following form:

$$-\frac{r^2}{2}\frac{\mathrm{d}^2R_{nl}(r)}{\mathrm{d}r^2}-r\frac{\mathrm{d}R_{nl}(r)}{\mathrm{d}r}+\left(\frac{l(l+1)}{2}-r+\frac{r^2}{2n^2}\right)R_{nl}(r)=0. \tag{6.32}$$

The range of used quantum numbers was from 1 to 5 for n, and for each n the range was from 0 to $n-1$ for l value. The domain of r values in general is infinite, for practical reasons it was restricted to the range $r \in [0, r_{max}]$ with $r_{max} = 50$. The boundary conditions for this problem are of two types: collocation points uniformly distributed in the number of $4 \cdot (n+l)$ and the Dirichlet boundary condition of vanishing function over large distances:

$$R_{nl}(r_{\mathrm{max}}) = 0. \tag{6.33}$$

The weights for additional conditions $\lambda_{\mathrm{boundary}}$ were equal to 1. Number of training points was increased to 256.

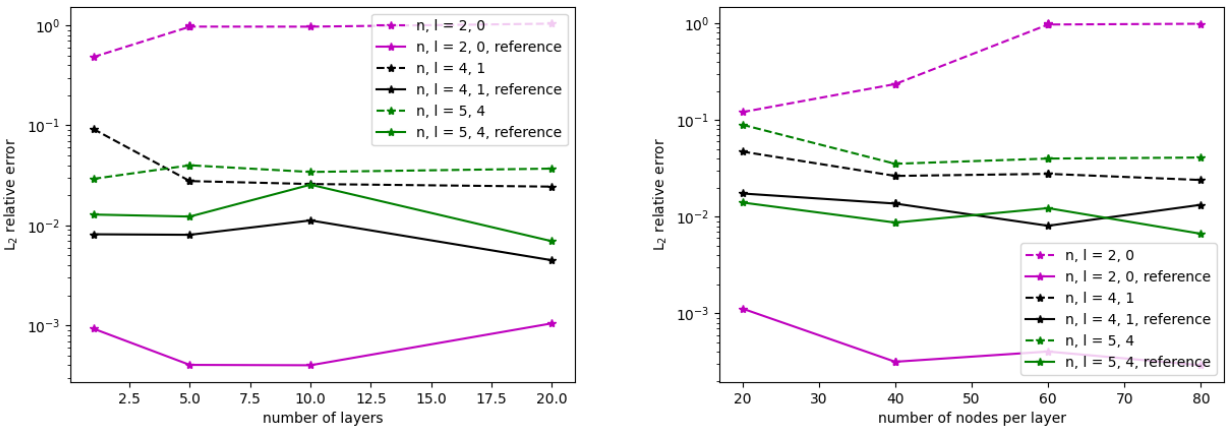


Figure 6.31: Dependence of the L_2 relative error on different number of layers, 60 nodes each (left) and on different number of nodes for 5 layers (right). Values for PINN for the hydrogen atom. Errors for fitting a function by a simple NN are shown as a reference

The errors obtained for selected examples of (n, l) combinations as a function of the number of layers of the PINN are presented in the left panel of Figure 6.31. The optimal number of layers was chosen to be equal to 5. From the right panel of Figure 6.31 it was chosen that the lowest overall errors are obtained for 40 nodes per layer. It is worth noting in this case that the error variations are smaller than in the other studied model problems. Optimal hyperparameters found for the rigid rotor and the hydrogen atom are presented in Table 6.5.

The errors obtained with increasing values of l for a fixed n are presented in Figure 6.32. The case of $n = 1$ is not shown, because it contains only one allowed value of l. For other values of n, the errors obtained by PINN are larger than in the reference method. As l increases, these errors tend to become smaller and approach the reference error. The reason is probably the same as for the rigid rotor, a better representation is obtained for functions with smaller oscillations (larger values of l). Examples of predicted functions are presented in Figure 6.33.

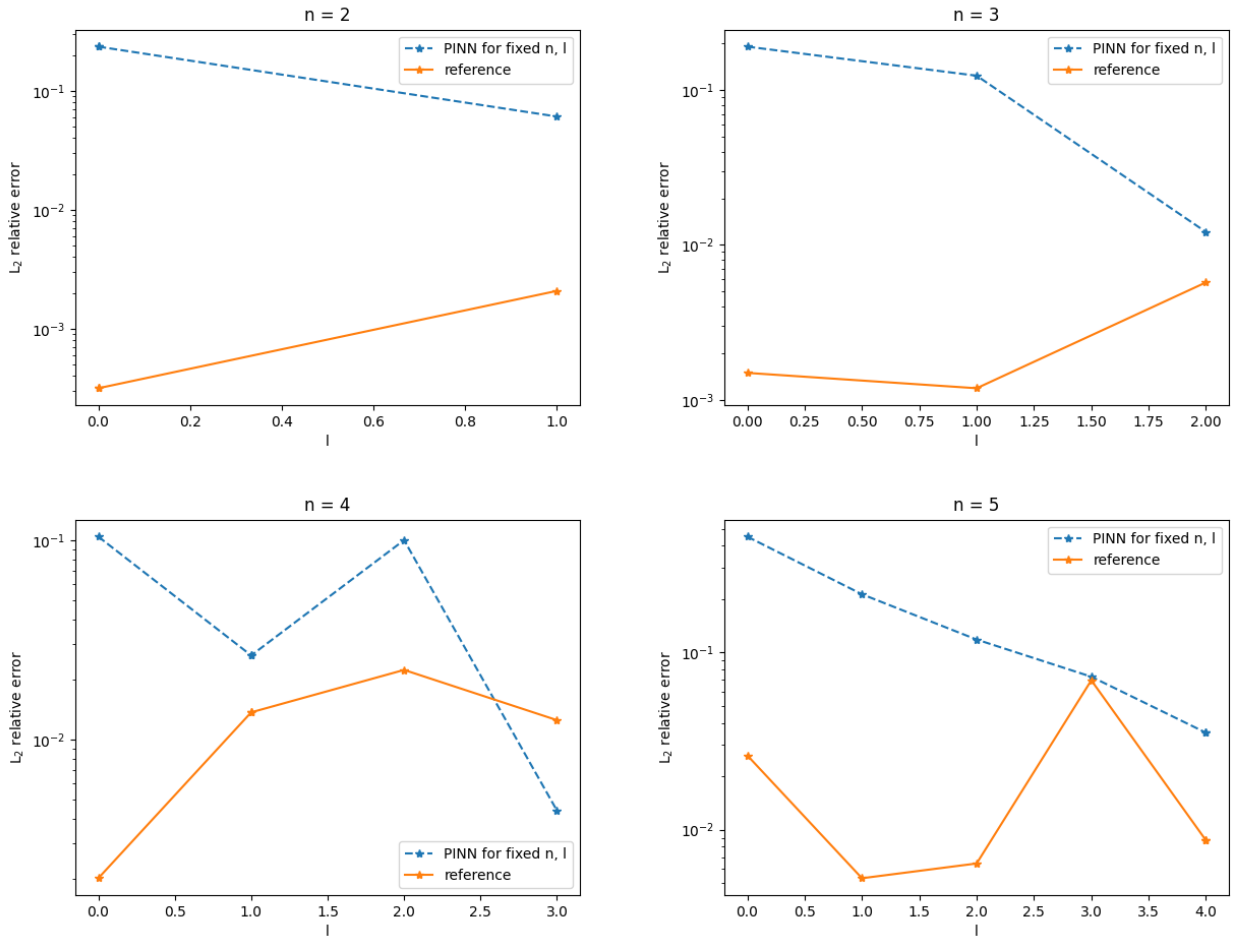


Figure 6.32: L_2 relative errors dependence on m for hydrogen atom for fixed values of n

Table 6.5: Optimal hyperparameters of models for rigid rotor and hydrogen atom

case	N_{layers}	N_{nodes}	N_{train}	N_{test}	N_{boundary}	$\lambda_{\text{boundary}}$.	epochs
rigid rotor	5	60	128	100	0	1	10000
hydrogen atom	5	40	256	100	1	1	10000

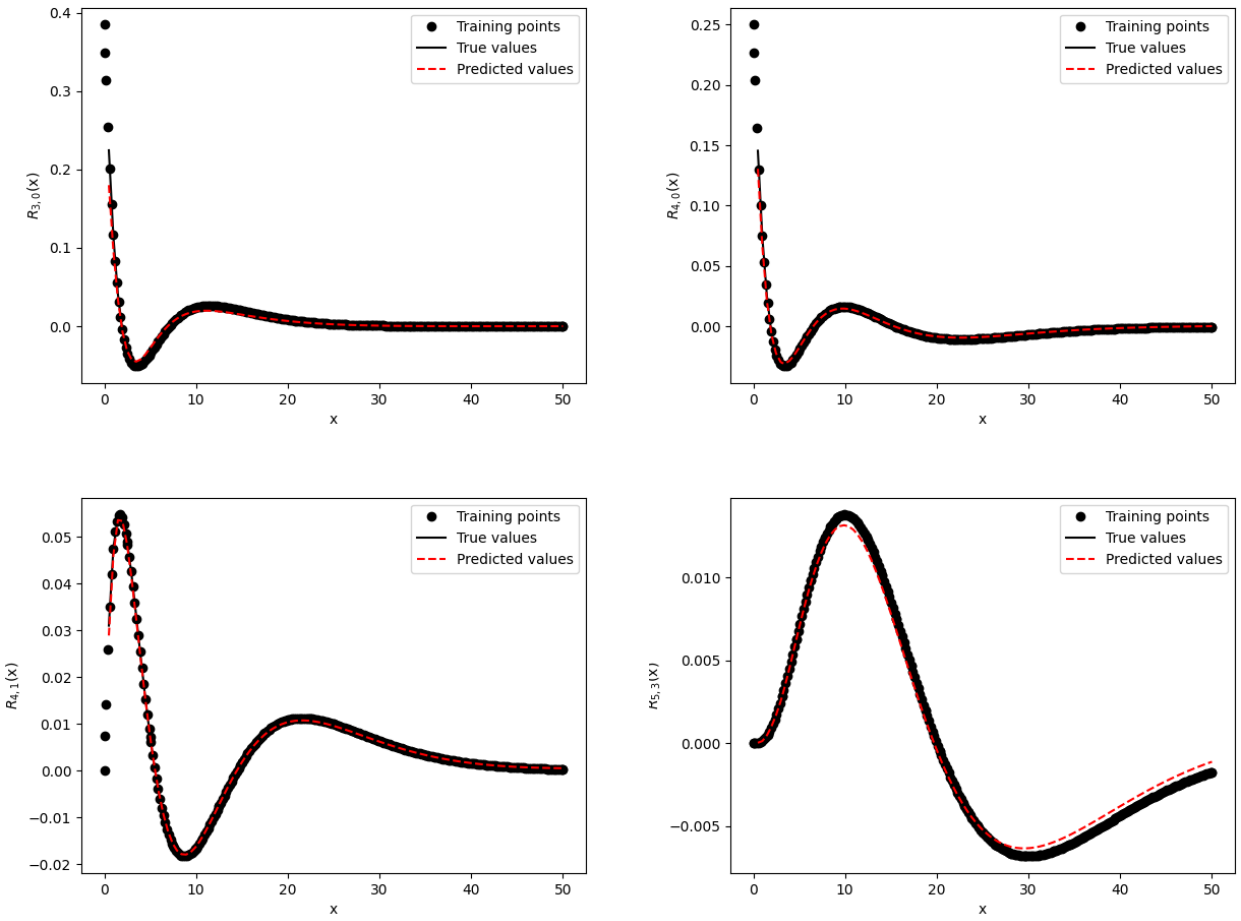


Figure 6.33: Examples of predictions made for hydrogen atom: $n = 3, l = 0$ (top left), $n = 4, l = 0$ (top right), $n = 4, l = 1$ (bottom left) and $n = 5, l = 3$ (bottom right)

6.3 Molecular dynamics

To ensure that the system state is equilibrated, the first five picoseconds of simulations were excluded from the analysis. To verify the stability of the system after the equilibrations (i.e. that the chemical bonds are not dissociating, the atoms do not collapse into each other, the velocities are not rapidly growing), at the first step the changes in potential energy and temperature during the simulation period were plotted. Sample dependencies for the EC system are shown in Figure 6.34.

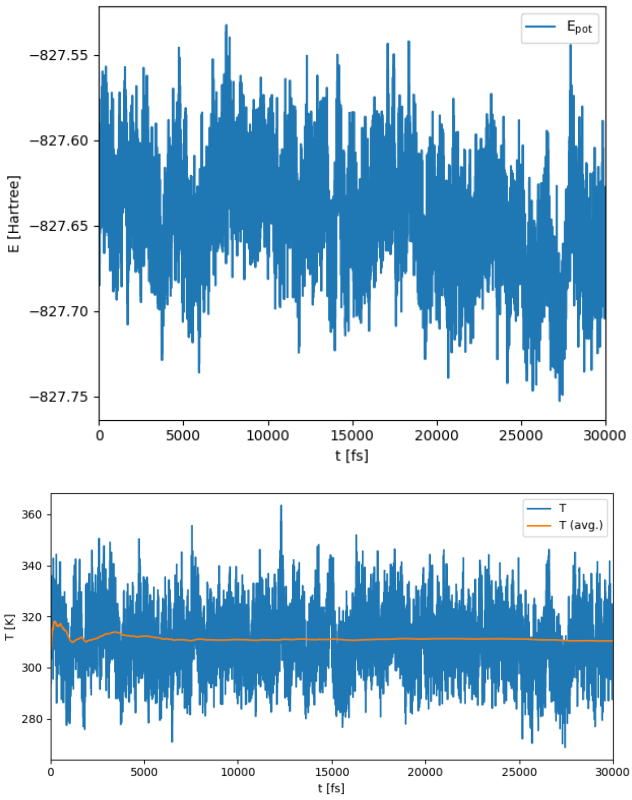


Figure 6.34: Changes of potential energy (left) and temperature (right) during the simulation of EC system. For the temperature also the running average value is plotted.

The potential energy shows fluctuations but does not exhibit rapid growths. The same holds for the temperature, where in addition the average value remains constant. This proves that the thermostating was implemented correctly. These two results indicate that the system was stable as instabilities would cause a rapid increase in potential or kinetic energy. Similar behaviour was observed in other systems.

A sample snapshot of the simulation box for the EC system is shown in Figure 6.35. The picture was prepared using VMD software [87]. The analysis confirmed that chemical bonds remained non-broken during the simulation and that the molecules did not collapse into each other.

To identify vibrations present in the system, the power spectrum was calculated. This is the spectrum calculated as a Fourier transform of the velocity autocorrelation function. The spectrum for the EC system is shown in Figure 6.36. The expected oscillations for C=O stretching mode at about 1800 cm^{-1} and for C-H stretching mode are located at about 3000 cm^{-1}

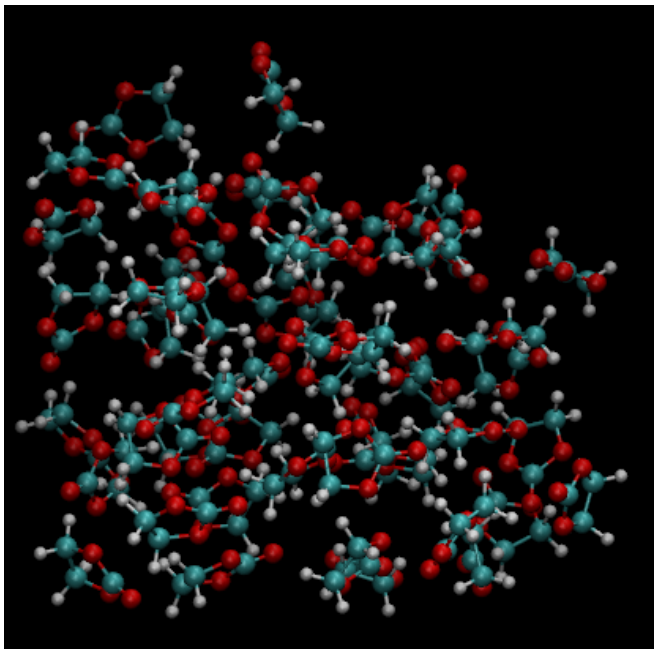


Figure 6.35: Snapshot of the last step of simulation for EC system. Oxygen atoms are marked with red color, carbon atoms with turquoise color and hydrogen atoms with white color

are present. This indicates that in the used approach the oscillations were correctly predicted in terms of their frequency. However, the power spectrum shows all vibrations present in the system, even these which are not active in the IR spectrum. Nonetheless, the prediction of the oscillations present in the IR spectrum may still be inaccurate. As a result, the dipole moments for the systems were calculated, for further verification.

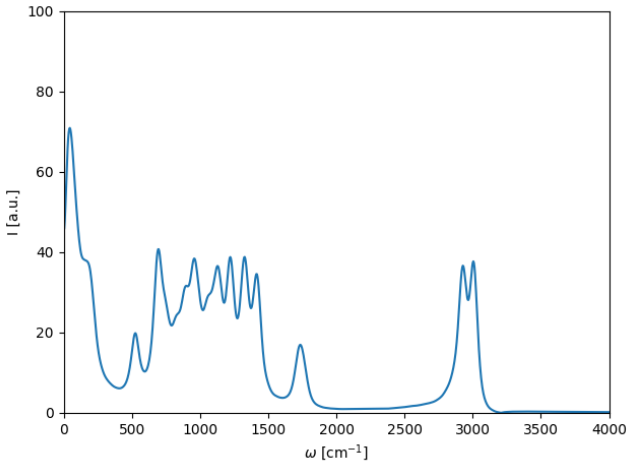


Figure 6.36: Power spectrum of the EC system

For all systems, theoretical IR spectra were calculated using Equation 4.98 with application of the harmonic correction (Equation 4.99) to include the temperature effect.

A comparison of spectra obtained from the full DFTB calculation (from paper [16]) and from DFTB combined with SchNet prediction of partial charges in the range of frequencies studied in the reference publication is presented in Figure 6.37. Both spectra have similar features: clear peaks between 1600 and 1800 cm^{-1}, between 1300-1400 cm^{-1} and at about 700 cm^{-1}. In the

case of SchNet supported simulation, the structure of the band between 1000-1200 cm^{-1} is less clear than in the full DFTB case. Thus, with the method proposed in this thesis the general shape of the spectrum was correctly reproduced, with some errors done for the low frequency range of the spectrum. However, activity in the IR spectrum of the vibration of the C=O group (at about 1700 cm^{-1}) was correctly predicted, but the position of the peak is shifted towards higher frequencies. Similar behaviour was observed for other systems (F1EC, F2EC). Thus, the systematic shift of this peak towards higher frequencies with respect to the full DFTB simulations could be the result of the systematic error done by trained NN, and probably could be lowered through expanding the training dataset or modifying the structure of the NN itself. This study was not done due to the long time consumed by the training process.

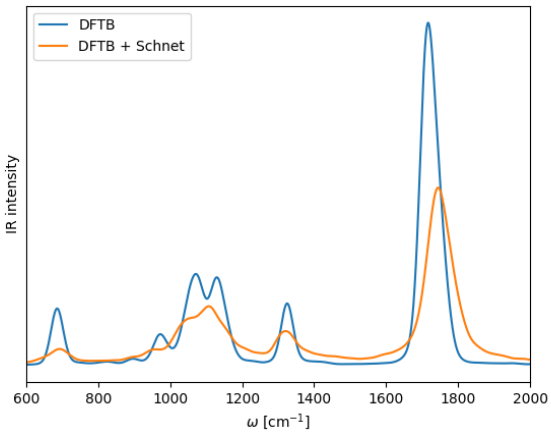


Figure 6.37: IR spectra of EC obtained from DFTB calculations [16] and from DFTB calculations supported by SchNet

Spectra for all carbonates (EC, F1EC, F2EC) for both full DFTB simulation and DFTB supported by Schnetpack are shown in Figure 6.38. The main effect studied in [16], the shift of the C=O band (around 1700 cm^{-1}) to higher frequencies with increasing fluorination of the solvent molecule, was correctly reproduced (in qualitative terms).

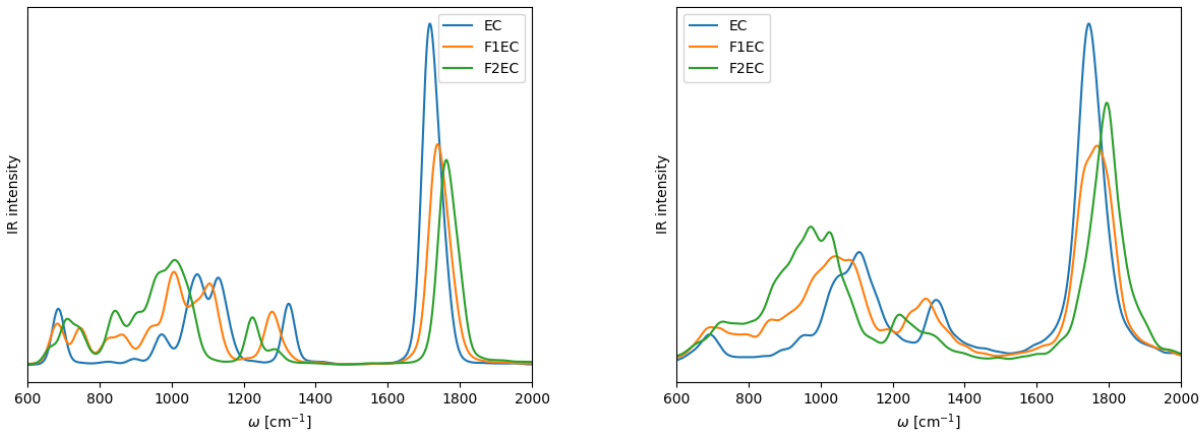


Figure 6.38: IR spectra of carbonates (EC, F1EC, F2EC) obtained from DFTB calculations [16] (left) and from DFTB calculations supported by Schnetpack (right)

To obtain a quantitative result, positions of C=O peaks were found for both approaches, and shifts relative to the EC position were calculated. Results are presented in Table 6.6. Based on these data, it can be observed that although the positions of the C=O bands in relation to the reference method were shifted, the relative shifts between carbonates were successfully reproduced. The shift for F1EC using the proposed method was equal to 22 cm^{-1} (21 cm^{-1} in the reference method) and the shift for F2EC was equal to 49 cm^{-1} (45 cm^{-1} in the reference method). This shows that the proposed method was able to correctly predict effects visible in IR spectra compared to the reference method.

Table 6.6: Positions of C=O bands on the spectra obtained from different methods for different carbonates

System	full DFTB [cm^{-1}]	DFTB + Schnetpack [cm^{-1}]	Shift full DFTB [cm^{-1}]	Shift DFTB + Schnetpack [cm^{-1}]
EC	1717	1745	0	0
F1EC	1738	1767	21	22
F2EC	1762	1794	45	49

Results presented in this chapter prove that DFTB method could be successfully supported by NNs. The results are not perfect – the low frequency region of the spectrum for carbonates could not be accurately replicated. However, the main effect studied in [16] was observed both qualitatively and quantitatively. This method allowed for faster performance of MD simulation — simulations were completed in 72 h on 24 CPU nodes, while the reference computations needed about two weeks on the same number of CPUs. The proposed method is promising and could give better results by incorporating larger datasets or modifying the NN structure.

Chapter 7

Summary

This work involved research on the abilities of neural networks to solve different variants of the Schrödinger equation. The PINN approach has been applied to certain model problems of quantum chemistry for which the exact analytical solutions are known.

The section describing model problems started with different variants of a quantum particle in an infinite potential well with different geometries. This problem proved to be difficult, because of the simple form of the differential equation describing this problem, which was satisfied by the trivial function identically equal 0. Manipulation of the loss function weights and the use of collocation points were needed to force the correct form of the predicted solution. The triangular and circular well involved special functions (Airy and Bessel functions) and these cases were also possible to be solved with PINNs. The harmonic oscillator, the rigid rotor and the hydrogen atom were further cases of solutions with special functions modeled with neural networks. However, the increasing complexity of the solution form and larger oscillations in the solution required a larger number of the training points used.

Where technically possible, various approaches using PINNs has been applied. Performing training for fixed values of quantum numbers usually led to the lowest errors, but this approach was limited. Another approach using quantum numbers as additional variables allowed for generalised predictions (e.g. predicting a function with a quantum number outside the learning set), but the quality of extrapolated data was not satisfactory. Additionally, training in these cases took much longer than for fixed quantum numbers. Furthermore, it was technically more difficult to use these methods, due to the lack of implementation of differentiation of special functions or the need to generalise functions defined for discrete arguments (e.g. Hermite polynomials of non-integer degrees).

All the model problems were predicted by PINNs with acceptable quality and some cases resulted in lower errors than for the reference methods. In each example, the values of errors were rather low and satisfactory. This demonstrates the applicability of PINNs in quantum chemistry for more complex systems.

SchNet NNs were able to correctly reproduce partial charges which are normally calculated in the SCC procedure of the DFTB method. Simulations of studied systems (EC and its derivatives) were stable, and did not lead to collapse or dissociation of the system. IR spectra obtained from the MD simulations were shifted with respect to the reference method, however, in both

qualitative and quantitative terms, the reproduction of shifts of C=O bands between different carbonates had satisfactory accuracy. Thus, the proposed approach is a promising method and should be studied more.

7.1 Further perspectives

During experiments made for this thesis some problems needed to be faced. One of the most crucial was to avoid PINNs prediction of function identically equal to 0 as the solution for the particle in the potential box. Thus, a possible extension of this work may be to develop an alternative method to achieve this goal than using collocation points with larger weights for this purpose.

Presented results show that PINNs trained with quantum numbers as additional variables of the model were successful in reproducing cases from the training dataset. They were also able to predict functions beyond those on which they were trained, but here the quality of the results is not satisfactory. Therefore, it seems that a more specialized method of using quantum numbers as variables in PINNs should be designed.

It would also be valuable to solve technical problems encountered during experiments. The implementation of automatic differentiation for Scipy special functions will allow to use generalised quantum numbers approaches for a broader class of problems than described here. Similarly, a valuable extension of DeepXDE would be to allow the use of complex arguments.

The modeling of described problems was done under the assumption that the exact energy E was known during training, which usually is not the case in real problems. A natural extension of this work would be to study similar PINNs but using E as an unknown parameter.

Due to the long time of training the Schnetpack NNs, the influence of the net structure on the accuracy of results was not examined. It may be also valuable to study the behaviour of NNs trained on more general datasets (for example, for dataset containing data for all carbonates at once, not only individual systems). In addition, the proposed approach to support DFTB calculations by NNs should be tested against more novel approaches currently used in DFTB, such as GFN2-xTB [88] or IPEA1-xTB [89].

7.2 Acknowledgements

I gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Centers: ACK Cyfronet AGH) for providing computer facilities and support within computational grants no. PLG/2022/015845 and PLG/2023/016721.

List of Figures

List of Tables

Bibliography

[1] F. Jensen. *Introduction to Computational Chemistry*. John Wiley & Sons, Ltd, 2007, pp. 315–350.

[2] A. Ponrouch, D. Monti, A. Boschin, B. Steen, P. Johansson, and M. R. Palacín. "Non-aqueous electrolytes for sodium-ion batteries". *J. Mater. Chem. A* 3 (2015), pp. 22–42.

[3] K. H. Wedepohl. "The composition of the continental crust". *Geochimica et Cosmochimica Acta* 59 (1995), pp. 1217–1232.

[4] H. Bae and Y. Kim. "Technologies of lithium recycling from waste lithium ion batteries: a review". *Mater. Adv.* 2 (2021), pp. 3234–3250.

[5] V. Palomares, P. Serras, I. Villaluenga, K. B. Hueso, J. Carretero-González, and T. Rojo. "Na-ion batteries, recent advances and present challenges to become low cost energy storage systems". *Energy Environ. Sci.* 5 (2012), pp. 5884–5901.

[6] H. D. Yoo, I. Shterenberg, Y. Gofer, G. Gershinsky, N. Pour, and D. Aurbach. "Mg rechargeable batteries: an on-going challenge". *Energy Environ. Sci.* 6 (2013), pp. 2265–2279.

[7] J. J. Xu, H. Ye, and J. Huang. "Novel zinc ion conducting polymer gel electrolytes based on ionic liquids". *Electrochem. Commun.* 7 (2005), pp. 1309–1317.

[8] J. Wahlers, K. D. Fulfer, D. P. Harding, D. G. Kuroda, R. Kumar, and R. Jorn. "Solvation Structure and Concentration in Glyme-Based Sodium Electrolytes: A Combined Spectroscopic and Computational Study". *J. Phys. Chem. C* 120 (2016), pp. 17949–17959.

[9] M. Okoshi, Y. Yamad, A. Yamad, and H. Nakai. "Theoretical analysis on de-solvation of lithium, sodium, and magnesium cations to organic electrolyte solvents". *J. Electrochem. Soc.* 160 (2013), pp. 2160–2165.

[10] A. Eilmes, P. Kubisiak, and M. Brela. "Explicit Solvent Modeling of IR and UV–Vis Spectra of 1-Ethyl-3-methylimidazolium Bis(trifluoromethylsulfonyl)imide Ionic Liquid". *J. Phys. Chem. B* 120 (2016), pp. 11026–11034.

[11] K. Xu. "Nonaqueous Liquid Electrolytes for Lithium-Based Rechargeable Batteries". *Chem. Rev.* 104 (2004), pp. 4303–4417.

[12] Y. Lee, J. Lee, H. Kim, K. Kang, and N.-S. Choi. "Highly stable linear carbonate-containing electrolytes with fluoroethylene carbonate for high-performance cathodes in sodium-ion batteries". *J. Power Sources* 320 (2016), pp. 49–58.

[13] M. Bolloli, F. Alloin, J. Kalhoff, D. Bresser, S. Passerini, P. Judenstein, and J.-C. Leprêtre. "Effect of Carbonates Fluorination on the Properties of LiTFSI-based Electrolytes for Li-ion Batteries". *Electrochim. Acta* 161 (2015), pp. 159–170.

[14] C. Xu, F. Lindgren, B. Philippe, M. Gorgoi, F. Björefors, K. Edström, and T. Gustafsson. "Improved Performance of the Silicon Anode for Li-Ion Batteries: Understanding the Surface Modification Mechanism of Fluoroethylene Carbonate as an Effective Electrolyte Additive". *Chem. Mater.* 27 (2015), pp. 2591–2599.

[15] Z. Zhang, L. Hu, H. Wu, W. Weng, M. Koh, P. C. Redfern, L. A. Curtiss, and K. Amine. "Fluorinated electrolytes for 5 V lithium-ion battery chemistry". *Energy Environ. Sci.* 6 (2013), pp. 1806–1810.

[16] P. Wróbel, P. Kubisiak, and A. Eilmes. "MeTFSI (Me = Li, Na) Solvation in Ethylene Carbonate and Fluorinated Ethylene Carbonate: A Molecular Dynamics Study". *J. Phys. Chem. B* 125 (2021), pp. 1248–1258.

[17] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations" (2017). URL: https://arxiv.org/abs/1711.10561.

[18] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics Informed Deep Learning (Part II): Data-driven Solutions of Nonlinear Partial Differential Equations" (2017). URL: https://arxiv.org/abs/1711.10566.

[19] M. Raissi. "Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations" (2018). URL: https://arxiv.org/abs/1801.06637.

[20] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006, pp. 225–291.

[21] D. Porezag, Th. Frauenheim, Th. Köhler, G. Seifert, and R. Kaschner. "Construction of tight-binding-like potentials on the basis of density-functional theory: Application to carbon". *Phys. Rev. B* 51 (1995), pp. 12947–12957.

[22] Emir Kocer, Tsz Wai Ko, and Jörg Behler. "Neural Network Potentials: A Concise Overview of Methods". *Annu. Rev. Phys. Chem.* 73 (2022), pp. 163–186.

[23] F. Leja. *Rachunek różniczkowy i całkowy*. PWN, 2012, p. 450.

[24] R. P. Feynman, R. B. Lighton, and M. Sands. *Feynmana wykłady z fizyki, t. 1.1*. PWN, 2014, pp. 89–90, 120–122.

[25] D. Marx and J. Hutter. *Ab initio molecular dymamics: basic theory and advanced methods*. Cambridge University Press, 2009, pp. 11–51.

[26] R. P. Feynman, R. B. Lighton, and M. Sands. *Feynmana wykłady z fizyki, t. 2.1*. PWN, 2014, p. 80.

[27] R. F. Nalewajski. *Perspectives in Electronic Structure Theory*. Springer, 2012, pp. 80–82, 93–113, 155–159.

[28] R. F. Nalewajski. *Podstawy i metody chemii kwantowej*. PWN, 2001, pp. 40–65, 87–95, 118–125, 128–132, 206–228.

[29] D. Kincaid and W. Cheney. *Numerical mathematics and computing*. Thomson Brooks/-Cole, 2008.

[30] C. Michoski, M. Milosavljević, T. Oliver, and D. R. Hatch. "Solving differential equations using deep neural networks". *Neurocomputing* 399 (2020), pp. 193–212.

[31] X. Chen, R. Chen, Q. Wan, R. Xu, and J. Liu. "An improved data-free surrogate model for solving partial differential equations using deep neural networks". *Sci. Rep.* 11 (2021), p. 19507.

[32] R. van der Meer, C. Oosterlee, and A. Borovykh. "Optimally weighted loss functions for solving PDEs with Neural Networks" (2021). URL: https://arxiv.org/abs/2002.06269.

[33] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators". *Nat. Mach. Intell* 3 (2021), pp. 218–229.

[34] S. Buchaniec, M. Gnatowski, and G. Brus. "An Analysis of an Integrated Mathematical Modeling - Artificial Neural Network Approach for the Problems with a Limited Learning Dataset" (2019).
URL: https://arxiv.org/abs/1911.03404.

[35] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Inferring solutions of differential equations using noisy multi-fidelity data". *J. Comput. Phys.* 335 (2017), pp. 736–746.

[36] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Machine learning of linear differential equations using Gaussian processes". *J. Comput. Phys.* 348 (2017), pp. 683–693.

[37] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Numerical Gaussian Processes for Time-dependent and Non-linear Partial Differential Equations" (2017).
URL: https://arxiv.org/abs/1703.10230.

[38] M. Raissi and G. E. Karniadakis. "Hidden physics models: Machine learning of nonlinear partial differential equations". *J. Comput. Phys.* 357 (2018), pp. 125–141.

[39] H. Owhadi. "Bayesian numerical homogenization". *Multiscale Model Simul.* 13 (2015), pp. 812–828.

[40] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz. "Data-driven discovery of partial differential equations". *Sci. Adv.* 3 (2017), e1602614.

[41] C. Yangang and J. W. L. Wan. "Deep neural network framework based on backward stochastic differential equations for pricing and hedging American options in high dimensions". *Quant. Finance* 21 (2020), pp. 45–67.

[42] V. Nourani and S. Mousavi. "Spatiotemporal groundwater level modeling using hybrid artificial intelligence-meshless method". *J. hydrol* 536 (2016), pp. 10–25.

[43] M. Pakdaman, A. Ahmadian, S. Effati, S. Salahshour, and D. Baleanu. "Solving differential equations of fractional order using an optimization technique based on training artificial neural network". *Appl. Math. Comput.* 293 (2017), pp. 81–95.

[44] C. J. Zúñiga-Aguilar, H. M. Romero-Ugalde, J. F. Gómez-Aguilar, R. F. Escobar-Jiménez, and M. Valtierra-Rodríguez. "Solving fractional differential equations of variable-order involving operators with Mittag-Leffler kernel using artificial neural networks". *Chaos Solitons Fractals* 103 (2017), pp. 382–403.

[45] S. Effati and M. Pakdaman. "Artificial neural network approach for solving fuzzy differential equations". *Inf. Sci.* 180 (2010), pp. 1434–1457.

[46] S. Effati and R. Buzhabadi. "A neural network approach for solving Fredholm integral equations of the second kind". *Neural Comput. & Applic.* 21 (2012), pp. 843–852.

[47] M. Magill, F. Qureshi, and H. W. de Haan. "Neural Networks Trained to Solve Differential Equations Learn General Representations" (2018).
URL: https://arxiv.org/abs/1807.00042.

[48] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zuybov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman. "Universal Differential Equations for Scientific Machine Learning" (2020).
URL: https://arxiv.org/abs/2001.04385.

[49] J. Berg and K. Nyström. "A unified deep artificial neural network approach to partial differential equations in complex geometries". *Neurocomputing* 317 (2018), pp. 28–41.

[50] Eduardo Abreu and Joao B. Florindo. *A Study on a Feedforward Neural Network to Solve Partial Differential Equations in Hyperbolic-Transport Problems*. Springer International Publishing, 2021, pp. 398–411.

[51] S. Markidis. "The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers?" *Front. Big Data* 4 (2021), p. 669097.

[52] C. Rackauckas, M. Innes, Y. Ma, J. Bettencourt, L. White, and V. Dixit. "DiffEqFlux.jl - A Julia Library for Neural Differential Equations" (2019).
URL: https://arxiv.org/abs/1902.02376.

[53] J. Y. Araz, J. C. Criado, and M. Spannowsky. "Elvet - a neural network-based differential equation and variational problem solver" (2021).
URL: https://arxiv.org/abs/2103.14575.

[54] F. Chen, D. Sondak, P. Protopapas, M. Mattheakis, S. Liu, D. Agarwal, and M. Di Giovanni. "NeuroDiffEq: A Python package for solving differential equations with neural networks". *JOSS* 5 (2020), p. 1931.

[55] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. "DeepXDE: A Deep Learning Library for Solving Differential Equations" (2020).
URL: https://arxiv.org/abs/1907.04502.

[56] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller. "SchNet – A deep learning architecture for molecules and materials". *J. Chem. Phys.* 148 (2018), p. 241722.

[57] K. T. Schütt, P. Kessel, M. Gastegger, K. A. Nicoli, A. Tkatchenko, and K.-R. Müller. "SchNetPack: A Deep Learning Toolbox For Atomistic Systems". *J. Chem. Theory Comput.* 15 (2019), pp. 448–455.

[58] T. M. Razakh, B. Wang, S. Jackson, R. K. Kalia, A. Nakano, K. Nomura, and P. Vshishta. "PND: Physics-informed neural-network software for molecular dynamics applications". *SoftwareX* 15 (2021), p. 100789.

[59] T. Wimpfheimer. "A Particle in a Box Laboratory Experiment Using Everyday Compounds". *J. Lab. Chem. Educ.* 3 (2015), pp. 19–21.

[60] R. Shankar. *Principles of Quantum Mechanics*. Plenum Press, 1994, p. 201.

[61] S. Tokita, T. Sugiyama, F. Noguchi, H. Fujii, and H. Kobayashi. "An Attempt to Construct an Isosurface Having Symmetry Elements". *J. comput. chem., Jpn.* 5.3 (2006), pp. 159–164.

[62] C. Møller and M. S. Plesset. "Note on an Approximation Treatment for Many-Electron Systems". *Phys. Rev.* 46 (1934), pp. 618–622.

[63] P. Hohenberg and W. Kohn. "Inhomogeneous Electron Gas". *Phys. Rev. A* 136 (1964), pp. 864–871.

[64] J. C. Slater and G. F. Koster. "Simplified LCAO Method for the Periodic Potential Problem". *Phys. Rev.* 94 (1954), pp. 1498–1524.

[65] P. Koskinen and V. Mäkinen. "Density-functional tight-binding for beginnners". *Comput. Mat. Sci.* 47 (2009), pp. 237–253.

[66] M. Gaus, A. Goez, and M. Elstner. "Parametrization and Benchmark of DFTB3 for Organic Molecules". *J. Chem. Theory Comput.* 9 (2013), pp. 338–354.

[67] M. Kubillus, T. Kubař, M. Gaus, J. Řezáč, and M. Elstner. "Parametrization of the DFTB3 Method for Br, Ca, Cl, F, I, K, and Na in Organic and Biological Systems". *J. Chem. Theory Comput.* 11 (2015), pp. 332–342.

[68] M. Gaus, X. Lu, M. Elstner, and Q. Cui. "Parametrization of DFTB3/3OB for Sulfur and Phosphorus for Chemical and Biological Applications". *J. Chem. Theory Comput.* 10 (2014), pp. 1518–1537.

[69] M. Elstner, D. Porezag, G. Jungnickel, J. Elsner, M. Haugk, Th. Frauenheim, S. Suhai, and G. Seifert. "Self-consistent-charge density-functional tight-binding method for simulations of complex materials properties". *Phys. Rev. B* 58 (1998), pp. 7260–7268.

[70] B. Hourahine, B. Aradi, V. Blum, F. Bonafé, A. Buccheri, C. Camacho, C. Cevallos, M. Y. Deshaye, T. Dumitrică, A. Dominguez, S. Ehlert, M. Elstner, T. van der Heide, J. Hermann, S. Irle, J. J. Kranz, C. Köhler, T. Kowalczyk, T. Kubař, et al. "DFTB+, a software package for efficient approximate density functional theory based atomistic simulations". *J. Chem. Phys.* 152.12 (2020), p. 124101.

[71] Thomas D. Kühne, Marcella Iannuzzi, Mauro Del Ben, Vladimir V. Rybkin, Patrick Seewald, Frederick Stein, Teodoro Laino, Rustam Z. Khaliullin, Ole Schütt, Florian Schiffmann, Dorothea Golze, Jan Wilhelm, Sergey Chulkov, Mohammad Hossein Bani-Hashemian, Valéry Weber, Urban Borštnik, Mathieu Taillefumier, Alice Shoshana Jakobovits, Alfio Lazzaro, et al. "CP2K: An electronic structure and molecular dynamics software package - Quickstep: Efficient and accurate electronic structure calculations". *J. Chem. Phys.* 152 (2020), p. 194103.

[72] William C. Swope, Hans C. Andersen, Peter H. Berens, and Kent R. Wilson. "A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters". *J. Chem. Phys.* 76 (1982), pp. 637–649.

[73] M. P. Allen and D. J. Tildesley. *Computer simulation of liquids*. Oxford University Press, 1991, pp. 130–140.

[74] C.-Y. Chen, T. Koko, T. Hosokawa, K. Matsumoto, T. Nohira, and R. Hagiwara. "Ionic liquid electrolytes with high sodium ion fraction for high-rate and long-life sodium secondary batteries". *J. Power Sources* 332 (2016), pp. 51–59.

[75] K. Fujii, S. Seki, S. Fukuda, R. Kanzaki, T. Takamuku, Y. Umebayashi, and S. Ishiguro. "Anion Conformation of Low-Viscosity Room-Temperature Ionic Liquid 1-Ethyl-3-methylimidazolium Bis(fluorosulfonyl) Imide". *J. Phys. Chem. B* 111 (2007), pp. 12829–12833.

[76] J. Sadlej. *Spektroskopia molekularna*. WNT, 2002, pp. 174–176, 217–221.

[77] Z. Kęcki. *Podstawy spektroskopii molekularnej*. PWN, 2013, pp. 11–18, 24–27.

[78] B. Guillot. "A molecular dynamics study of the far infrared spectrum of liquid water". *J. Chem. Phys.* 95 (1991), pp. 1543–1551.

[79] R. G. Gordon. "Molecular Motion in Infrared and Raman Spectra". *J. Chem. Phys.* 43 (1965), pp. 1307–1312.

[80] R. Ramirez and T. López-Ciudad. "Quantum corrections to classical time-correlation functions: Hydrogen bonding and anharmonic floppy modes". *J. Chem. Phys.* 121 (2004), pp. 3973–3983.

[81] C. P. Lawrence, A. Nakayama, N. Nakri, and J. L. Skinner. "Quantum dynamics in simple fluids". *J. Chem. Phys.* 120 (2004), pp. 6621–6624.

[82] P. Wróbel. "Studies of interactions in solutions based on Molecular Dynamics methods and simulated vibrational spectra" (2023), pp. 122–124.
URL: https://rozprawy-doktorskie.bip.uj.edu.pl/nauki-scisle-i-przyrodnicze/.

[83] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". *Nat. Methods* 17 (2020), pp. 261–272.

[84] A. K. Rappe, C. J. Casewit, K. S. Colwell, W. A. Goddard III, and W. M. Skiff. "UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations". *J. Am. Chem. Soc.* 114 (1992), pp. 10024–10035.

[85] H. Forbert and A. Kohlmeyer. *Fourier: Power Spectrum of Autocorrelation Function.*

[86] J. Hutter and M. Ianuzzi. "CPMD: Car-Parinello molecular dynamics". *Z. Kristallogr. Cryst. Mater.* 220 (2005), pp. 549–551.

[87] W. Humphrey, A. Dalke, and K. Schulten. "VMD - Visual Molecular Dynamics". *J. Molec. Graphics* 14 (1996), pp. 33–38.

[88] C. Bannwarth, S. Ehlert, and S. Grimme. "GFN2-xTB—An Accurate and Broadly Parametrized Self-Consistent Tight-Binding Quantum Chemical Method with Multipole Electrostatics and Density-Dependent Dispersion Contributions". *J. Chem. Theory Comput.* 15 (2019), pp. 1652–1671.

[89] V. Ásgeirsson, C. A. Bauer, and S. Grimme. "Quantum chemical calculation of electron ionization mass spectra for general organic and inorganic molecules". *Chem. Sci.* 8 (2017), pp. 4879–4895.