2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI) | 978-1-6654-2534-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ECAI52376.2021.9515092

Finding approximate analytical solutions of differential equations using Neural Networks with self-adaptive training sets

1st Felix Hamza-Lup
Computer Science
Georgia Southern University
Statesboro, USA
fhamzalup@GeorgiaSouthern.edu

2nd Ionut E. Iacob
Mathematical Sciences
Georgia Southern University
Statesboro, USA
ieiacob@GeorgiaSouthern.edu

3rd James Orgeron
Mathematical Sciences
Georgia Southern University
Statesboro, USA
jo02219@GeorgiaSouthern.edu

***Abstract*—Artificial Neural Networks are known as powerful models capable of discovering complicated patterns and are de facto standard models in deep learning. But they are also universal function approximators and, consequently, their applicability extends to finding approximate solutions for many computational problems. These applications are interesting not only for mathematicians, but also for computer scientists and engineers interested in learning new models for many classical problems (for instance, fluid dynamics modelling, dynamical systems control, etc.). We present Neural Networks based methods for solving differential equations analytically and use the underlying optimization problem's loss function to produce localized additional training data. Our method uses a reduced initial training dataset, which is gradually, non-uniformly augmented in order to reduce the model's approximation error. This method can be used to directly produce analytical solutions for differential equations or as a pre-processing method for finding optimal, non-uniform grid points for traditional grid-based methods.**

***Index Terms*—neural networks, approximate solutions for differential equations, adaptive training data**

I. Introduction

The Artificial Neural Network (ANN) model is a computational model inspired from the biological model of the human brain. It has been originally created to mimic the learning activity of the brain and subsequently produce decisions based on the information "incorporated" into the model during the learning phase. While the first ANN models were created in the 40's, ANNs gained popularity in the 90's after efficient computational algorithms (backpropagation [1]) were discovered. Since then, ANNs have been successfully used in many applications in machine learning. More recently, they are the de facto models for deep learning. The famous Cybenko's "universal approximation theorem" [2], [3] (improved by Hornik et al in [4]), which states that an ANN model with one hidden layer can possibly approximate any continuous function, with any precision, makes the ANN model very appealing for a wide range of applications. In particular, Lagaris et al [5] introduced a general framework for solving initial value problems using ANN models. This new area of applications received continuous interest [6]–[9], as the approximate solutions produced by ANN models are analytical, rather than numerical.

Typically, ANNs models are used for performing machine learning classification or regression. In a typical setup, a training data set of observations whose categories (classes) are previously known are used to optimally find the parameters of the ANN's model so that the model computes the given observations' classes with good accuracy. This is called the learning phase. Subsequently, the model is used to compute classes of new observations (of categories previously unknown). In this work we do not use the ANN model in the classical way for machine learning classification. We use an ANN model as an approximation model, for the best approximation mapping between a set of input values and a set of output values. As in previous approaches, we train our model on grid points in the domain. The novelty of our approach consists of initially using a reduced grid size, then gradually refining and increasing the number of grid points on the domain boundary in order to obtain superior accuracy.

Differential equation can be classified as an ordinary (ODE) or partial differential equation (PDE) which depends on whether only ordinary derivatives are involved or partial derivatives are involved. The differential equation can also be classified as linear or nonlinear and have a wide spectrum of applications in engineering, from heat transfer and elasticity computation to quantum mechanics and fluid dynamics applications.

The rest of the article is organized as follows. In Section II we give a short background on the ANN computational model and how the model is being used to find analytical solutions for differential equations. Section III presents the self adaptive training set framework for solving differential equations. We show some examples and experimental results in Section IV and conclude in Section V.

978-1-6654-2534-6/21/$31.00 ©2021 IEEE

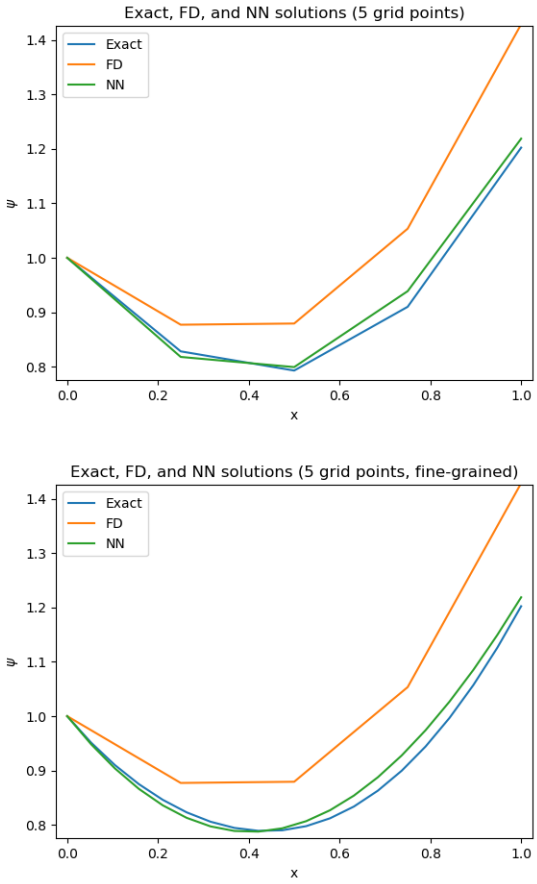


Fig. 1. Numerical, exact, and ANN model solutions.

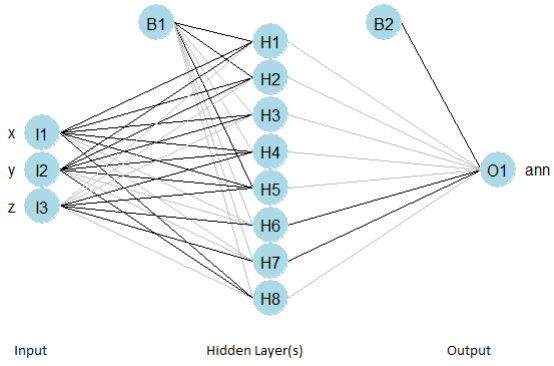


Fig. 2. Artificial Neural Network (ANN) model.

II. Background

Finding numerical solutions of differential equation for which no analytical solution is known has been of continuous interest for mathematicians and engineers (a recent survey of methods can be found in [10]). Many such numerical methods are currently successfully used to solve complex differential equations with applications in engineering [11], physics [12], chemistry [13], biology [14], economics [15], etc. These numerical methods typically rely on domain discretization (domain grid points) of the differential equation and solving large systems of linear or non-linear algebraic equations. The solution is then presented as a numerical array of values of the approximate solution at each domain grid point. Intermediate values can be found by interpolation, as needed. The ANN model solution [5] produces an approximate analytical solution, which is differentiable and integrable. Moreover, such an analytical solution is very compact, compared with the numerical counterpart, making it suitable for web applications [16] and otherwise easier to use for computing the numerical value of the solution at any domain point, without the need of interpolation.

For instance, for the ODE:

$$\frac{d\psi}{dx} + \left(x + \frac{1+3x^2}{1+x+x^3}\right)\psi = x^3 + 2x + x^2\frac{1+3x^2}{1+x+x^3}$$

on the domain

$$\Omega = [0, 1]$$

$$\psi(0) = 1$$

with the exact analytic solution:

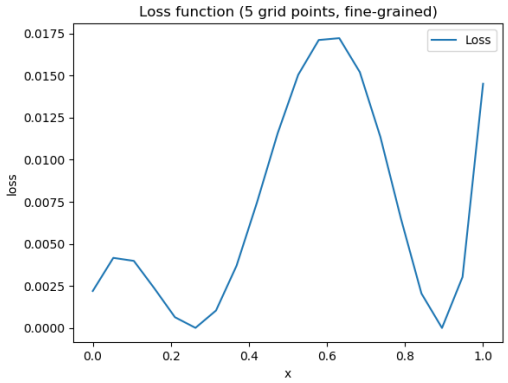
$$\psi_a(x) = \frac{e^{-x^2/2}}{1+x+x^3}$$

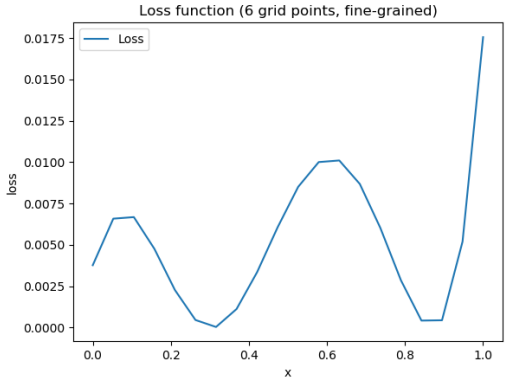
the numerical (finite differences, FD), exact, and ANN model solutions are presented in Fig. 1. The figure on the left hand side shows only the numerical results of the analytical and ANN model solutions only at the 5 grid points being considered, whereas the very same solutions are represented with finer granularity (hence, smoother) on the right hand side.

A. Artificial Neural Networks

Fig. 2 shows an example of an ANN, which is essentially a network flow model where the information flows from the "input" layer (with three input nodes) through one (or possibly more) "hidden" layer (with eight hidden nodes) to the "output" layer (with one output node), from the left to the right in the figure. The model has a single input layer (with one or more input nodes, as appropriate), a single output layer (with one or more output nodes, as appropriate) and one or mode hidden layers (with variable number of nodes in each layer).

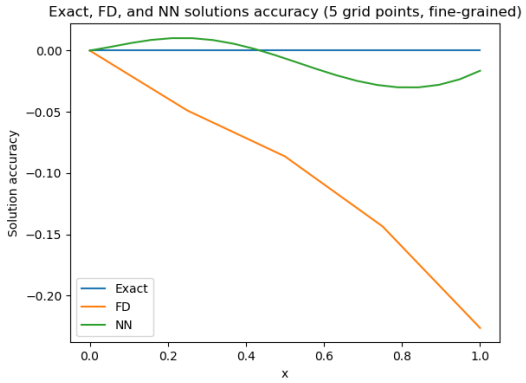
Computationally, ANN is a parametric model where to each edge of the network corresponds a weight parameter and the model performs a non-linear transformation from the input





MSE = 0.006960136028878594 MSE = 0.00524926392301045

Fig. 3. The loss distribution function for the ANN model solution for (1).



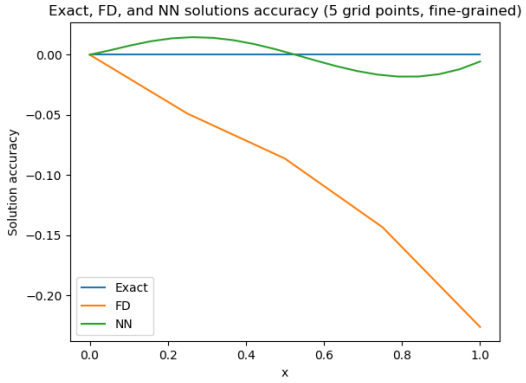


Fig. 4. Solutions accuracy for equation (1).

space ((x, y, z) in the figure) to the output space ($O1$ in the figure). More formally, for the model in Fig. 2:

$$\begin{aligned} O1 &: \mathbb{R}^3 \to \mathbb{R} \\ O1(x,y,z) &= \sum_{i=1}^{L} Z_i H_i + B_2 \\ \text{where} & \\ H_i &= \sigma(W_{1i}x + W_{2i}y + W_{3i}z + B_{1i}) \\ \sigma(x) &= \frac{1}{1+e^{-x}} \end{aligned} \tag{1}$$

In the ANN model (1), W_{ij}, Z_i, B_{1i}, and B_2 represent the weight parameters associated to edges from input to hidden layer, hidden layer to output, node B_1 to hidden, and node B_2 to output, respectively. The model parameters are determined as solutions of an optimization problem as follows. Given a training data set $\{(x_i, y_i) \mid x_i \in \mathbb{R}^n, y_i \in \mathbb{R}^m, i = 1...L\}$ (where n is the input space dimension, m is the output space dimension), the ANN parameters are the solution of the optimization problem:

$$\min_{W,B} \sum_{i=1}^{L} ||O1(x_i; W, B) - y_i||$$

where $O1$ is the ANN model function as in (1), for the appropriate network input and output space dimensions, and W, B are generic notations for all parameters in the respective network.

B. The ANN model for solving differential equations

For a general differential equation (one- or multi-dimensional) $F(\boldsymbol{x}, \psi(\boldsymbol{x}), \nabla\psi(\boldsymbol{x}), \triangle\psi(\boldsymbol{x})) = 0$ on a domain Ω, with appropriate boundary conditions, the finite difference methods (FDM) rely on discretizing the domain Ω into a set of grid-points $\{x_1, x_2, ..., x_L\}$ then use various discrete approximations for all derivatives in the equation to convert the given equation into L algebraic equations, which are subsequently solved using specific methods.

Lagaris et al [5] define a NN-based model solution

$$\psi_m(\boldsymbol{x}) = A(\boldsymbol{x}) + B(\boldsymbol{x})O1(\boldsymbol{x}; \boldsymbol{w}) \tag{2}$$

(where $A(\boldsymbol{x})$ and $B(\boldsymbol{x})$ are defined as appropriate [5] for the given boundary conditions) of which parameters are the solutions of the optimization problem:

$$\min_{\boldsymbol{w}} \sum_{i}^{L} \left[F\left(\boldsymbol{x_i}, \psi_m(\boldsymbol{x_i}), \nabla\psi_m(\boldsymbol{x_i}), \triangle\psi_m(\boldsymbol{x_i})\right)\right]^2 \tag{3}$$

where the training set $\{x_1, x_2, ..., x_L\}$ consists of the domain grid points, as in FD methods.

A major issue for FDMs and the analytical solution provided through (2) and (3) is the selection of the domain grid

points. Intuitively, uniform grid points are suitable for non-stiff differential equations (yet, the appropriate step size may still be an issue) but not good enough for stiff equations. Our method builds on the solution given by (2) and (3) and tries to overcome the grid points selection such that the overall solution error decreases.

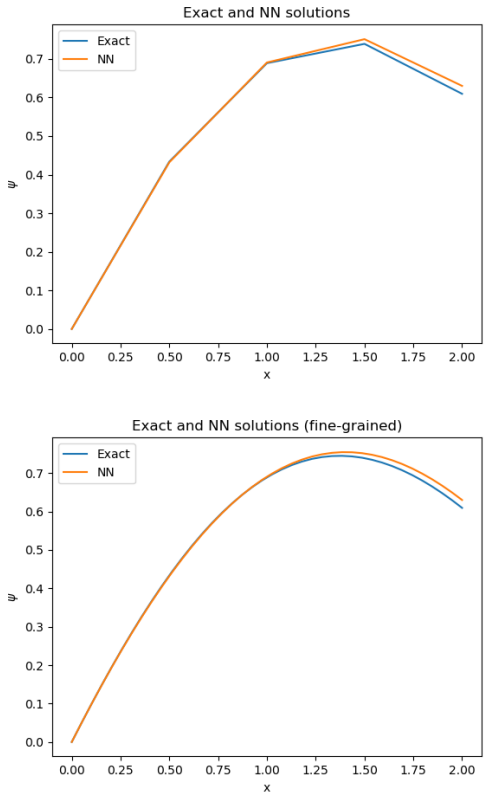


Fig. 5. The ANN and analytical solutions.

III. A SELF-ADAPTIVE TRAINING SET ANN MODEL FOR FINDING EXACT SOLUTION FOR DIFFERENTIAL EQUATIONS

The basic idea of the method is to start with a coarse domain grid as the ANN's initial training set, then gradually and non-uniformly add grid points so that the overall error decreases. For this purpose, we define the *loss* function distribution (for each grid point) based on (3) as:

$$loss(\boldsymbol{x_i}) = \left[F\left(\boldsymbol{x_i}, \psi_m(\boldsymbol{x_i}), \nabla\psi_m(\boldsymbol{x_i}), \triangle\psi_m(\boldsymbol{x_i})\right)\right]^2 \quad (4)$$

Correspondingly, the mean squared error (MSE) value is computed as:

$$MSE = \left(\sum_{i=1}^{L}\left[F\left(\boldsymbol{x_i}, \psi_m(\boldsymbol{x_i}), \nabla\psi_m(\boldsymbol{x_i}), \triangle\psi_m(\boldsymbol{x_i})\right)\right]^2\right)/L$$

The key difference between (3) and (4) is that the loss function in (4) is being computed after ANN's training (that is, after the equation's analytical solution is being computed).

The self-adaptive training set ANN model proceeds as follows:

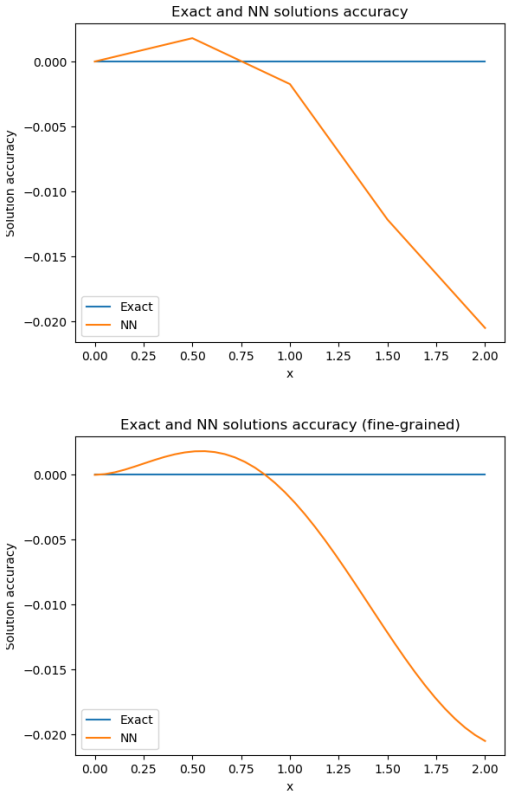
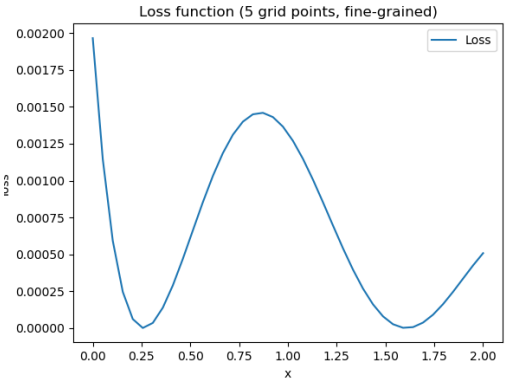


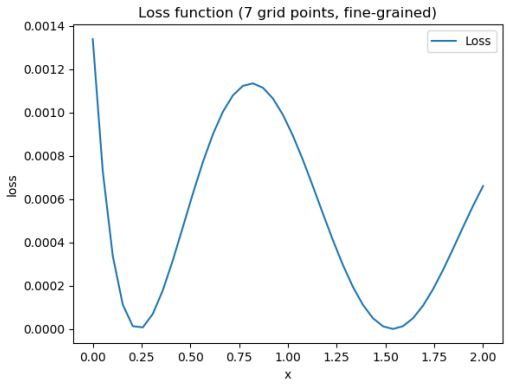
Fig. 6. Solutions accuracy.

1) Input parameters:
   - differential equation: $F\left(\boldsymbol{x}, \psi(\boldsymbol{x}), \nabla\psi(\boldsymbol{x}), \triangle\psi(\boldsymbol{x})\right) = 0$ on a domain Ω
   - number of initial grid points: L
   - percentage of loss reduction at each iteration: $lossP$
   - desired total loss error, maximum iterations: $lossE$, $maxN$
2) Initialize uniform grid: $\{x_1, x_2, ..., x_L\}$
3) Compute the analytical solution $\psi_m(\boldsymbol{x}) = A(\boldsymbol{x}) + B(\boldsymbol{x})O1(\boldsymbol{x}; \boldsymbol{w})$
4) Compute the corresponding total loss value $totalloss = \sum_{i}^{L}\left[F\left(\boldsymbol{x_i}, \psi_m(\boldsymbol{x_i}), \nabla\psi_m(\boldsymbol{x_i}), \triangle\psi_m(\boldsymbol{x_i})\right)\right]^2$
5) If desired total loss or maximum iterations achieved, then stop
6) Generate additional grid points $\{x_{L+1}, ..., x_{L+p}\}$ in the subdomains of Ω where $loss(\boldsymbol{x_i}) > loss(\boldsymbol{x_i}) * lossP$
7) Increment the number of grid points: $L = L + p$, $\{x_1, x_2, ..., x_L\} \cup \{x_{L+1}, ..., x_{L+p}\}$
8) Repeat from Step (3)

For instance, for the example given by equation (1) we added a sixth grid point (at $x = 0.6$). Fig. 3 illustrates side-by-side the loss function for 5- and 6-point, respectively. The 6th point is selected based on the loss function for the 5-point grid. The figure also shows how the corresponding MSE value decreases from 5-point to 6-point grid. Subsequently, Fig. 4



MSE = 0.0006336612324773529



MSE = 0.0004999357018473655

Fig. 7. The loss distribution functions.

illustrates side-by-side the solutions accuracy for the 5-point grid and 6-point grid.

IV. Experimental results

In addition to the solution for equation (1), we run experiments for one more ODE and one PDE, as in [5]. The results are presented in the following subsections.

A. Second order ODE, mixed boundary conditions

We consider the second order ODE with mixed boundary conditions:

$$\frac{d^2\psi}{dx^2} + \frac{1}{5}\frac{d\psi}{dx} + \psi = -\frac{1}{5}e^{-x/5}\cos x$$
$$\Omega = [0, 2]$$
$$\psi(0) = 0, \quad \frac{d\psi}{dx}(0) = 1$$

and the analytical solution:

$$\psi_a(x) = e^{-x/5}\sin x$$

Figures 5, 6, and 7 (the right hand side corresponds to two extra grid points $x = 0.05$ and $x = 0.8$) show the corresponding analytical solutions, accuracies, and loss distribution functions for the experiment. The corresponding MSE values in Fig. 7 show the error decrease when two extra grid points are being used.

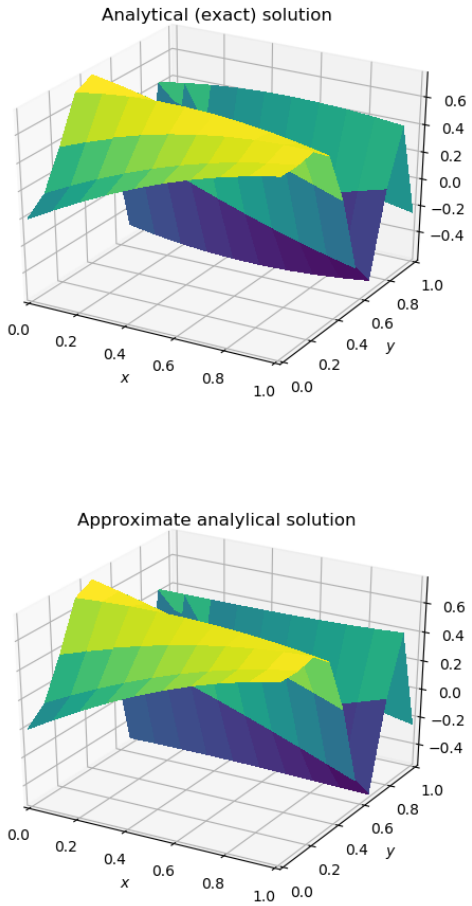


Fig. 8. The analytical (exact) and ANN (approximate) solutions.

B. PDE, Dirichlet boundary conditions

Next, we considered the following PDE:

$$\Delta\psi = f(x, y), \qquad \text{where:}$$
$$f(x,y) = e^{-\frac{3x+y}{5}}\left(\left(-\frac{108}{5}x + \frac{88}{5}\right)\cos(9x^2 + y) + \left(-\frac{3}{5} - 324x^2\right)\sin(9x^2 + y)\right),$$
$$\text{on } \Omega = [0, 1]^2$$

The corresponding analytical solution is given in [5]. The corresponding solutions (exact and ANN model) are shown side by side in Fig. 8 and error and loss function in Fig. 9.

V. Conclusion

We present a practical method for improving Artificial Neural Networks models solutions for differential equations based on smart refinement of the domain grid points. The method produces iterative solutions, starting with coarse grid

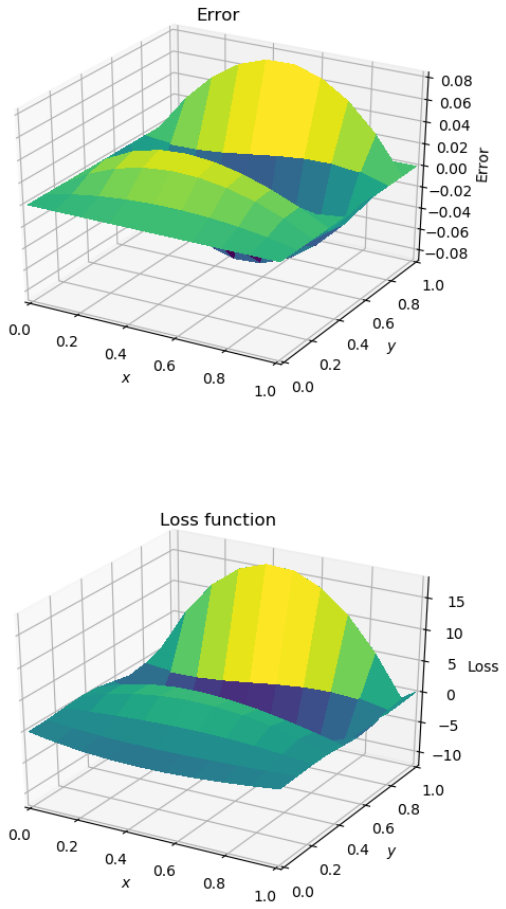


Fig. 9. Solution accuracy and the loss function.

points in the initial training set, then, based on the model's approximation error additional grid points are chosen for the subsets of domain where the approximation error is largest.

The method presented is suitable for finding approximate analytical solutions of ordinary or partially differential equations, or as a preprocessing step for finding optimal, non-uniform grid points for traditional numerical solutions.

Acknowledgement. We would like to thank the anonymous reviewers for the corrections and suggestions that helped us make a better version of the paper.

REFERENCES

[1] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, p. 533–536, 1986.

[2] G. Cybenko, "Approximations by superpositions of sigmoidal functions," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[3] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[4] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[5] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," in *IEEE Transactions on Neural Networks*, 1998, vol. 9, no. 5, pp. 987–1000.

[6] J. Han, A. Jentzen *et al.*, "Overcoming the curse of dimensionality: Solving high-dimensional partial differential equations using deep learning," *arXiv preprint, arXiv:1707.02568*, 2017.

[7] M. L. Piscopo, M. Spannowsky, and P. Waite, "Solving differential equations with neural networks: Applications to the calculation of cosmological phase transitions," *Physical Review D*, vol. 100, no. 1, Jul 2019. [Online]. Available: http://dx.doi.org/10.1103/PhysRevD.100.016002

[8] T. Dockhorn, "A discussion on solving partial differential equations using neural networks," 2019.

[9] L. S. Tan, Z. Zainuddin, and P. Ong, "Solving ordinary differential equations using neural networks," *AIP Conference Proceedings*, vol. 1974, no. 1, p. 020070, 2018.

[10] B. Denis, "An overview of numerical and analytical methods for solving ordinary differential equations," 2020.

[11] K. Chau, *Applications of Differential Equations in Engineering and Mechanics*, 01 2019.

[12] T. Li and T. Qin, *Physics and Partial Differential Equations, Volume 1*. USA: Society for Industrial and Applied Mathematics, 2012.

[13] G. Scholz and F. Scholz, "First-order differential equations in chemistry," *ChemTexts*, vol. 1, 03 2014.

[14] I. Mihai, M. Turnea, and M. Rotariu, "Ordinary differential equations with applications in molecular biology," *Revista medico-chirurgicala a Socitatii de Medicina si Naturalisti din Iasi*, vol. 116, pp. 347–52, 10 2012.

[15] M. Burger, L. Caffarelli, and P. A. Markowich, "Partial differential equation models in the socio-economic sciences," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 372, no. 2028, November 2014.

[16] F. G. Hamza-Lup, I. E. Iacob, and S. Khan, "Web-enabled intelligent system for continuous sensor data processing and visualization," in *The 24th International Conference on 3D Web Technology, Web3D, Los Angeles, California, USA, July 26-28, 2019*, 2019, pp. 1–7. [Online]. Available: https://doi.org/10.1145/3329714.3338127