Solving differential equations using neural networks

M. M. Chiaramonte and M. Kiener

1 INTRODUCTION

The numerical solution of ordinary and partial differential equations (DE's) is essential to many engineering fields. Traditional methods, such as finite elements, finite volume, and finite differences, rely on discretizing the domain and weakly solving the DE's over this discretization. While these methods are generally adequate and effective in many engineering applications, one limitation is that the obtained solutions are discrete or have limited differentiability. In order to avoid this issue when numerically solving DE's (i.e., obtain a differentiable solution that can be evaluated continuously on the domain), one can implement a different method which relies on neural networks (NN). The purpose of this study is to outline this method, implement it for some examples, and analyze some of its error properties.

2 FORMULATION

The study is restricted to second-order equations of the form

$$G(x, \Psi(x), \nabla\Psi(x), \nabla^2\Psi(x)) = 0, \quad \forall x \in D, \tag{1}$$

where $x \in \mathbf{R}^n$ is the independent variable over the domain $D \subset \mathbf{R}^n$, and $\Psi(x)$ is the unknown (scalar-valued) solution. The boundary of the domain is decomposed as $\partial D = \overline{\partial_d D \cup \partial_\tau D}, \emptyset = \partial_d D \cup \partial_\tau D$, where $\partial_d D$ is the portion of ∂D where essential boundary conditions (BC's) are specified. This study is restricted to problems with only essential BC's: for a given function $\hat{\Psi}(x)$, $\Psi(x) = \hat{\Psi}(x)$, $\forall x \in \partial_d D$. To approximately solve the above using an NN, a trial form of the solution is assumed as

$$\Psi_t(x, p) = \hat{\Psi}(x) + F(x)N(x, p), \tag{2}$$

where $N(x, p)$ is a feedforward NN with parameters p. The scalar-valued function $F(x)$ is chosen so as not to contribute to the BC's: $F(x) = 0, \forall x \in \partial_d D$. This allows the overall function $\Psi_t(x, p)$ to automatically satisfy the BC's. A subtle point is that (the single function) $\hat{\Psi}(x)$ must often be constructed from piecewise BC's (see Section 3). Furthermore, for a given problem there are multiple ways to construct $\hat{\Psi}(x)$ and $F(x)$, though often there will be an "obvious" choice.

The task is then to learn the parameters p such that Eqn. 1 is approximately solved by the form in Eqn. 2. To do this, the original equation is relaxed to a discretized version and approximately solved. More specifically, for a discretization of the domain $\hat{D} = \left\{x^{(i)} \in D; i = 1, \ldots, m\right\}$, Eqn. 1 is relaxed to hold only at these points:

$$G(x^{(i)}, \Psi(x^{(i)}), \nabla\Psi(x^{(i)}), \nabla^2\Psi(x^{(i)})) = 0, \quad \forall i = 1, \ldots, m. \tag{3}$$

Note this relaxation is general and independent of the form in Eqn. 2. Because with a given NN it may not be possible to (*exactly*) satisfy Eqn. 3 at each discrete point, the problem is further relaxed to find a trial solution that "nearly satisfies" Eqn. 3 by minimizing a related error index. Specifically, for the error index

$$J(p) = \sum_{i=1}^{m} G(x^{(i)}, \Psi_t(x^{(i)}, p), \nabla_x\Psi_t(x^{(i)}, p), \nabla_x^2\Psi_t(x^{(i)}, p))^2, \tag{4}$$

the optimal trial solution is $\Psi_t(x, p^\star)$, where $p^\star = \arg\min_p J(p)$. The optimal parameters can be obtained numerically by a number of different optimization methods [1], such as back propagation or the quasi-Newton BFGS algorithm. Regardless of the method, once the parameters $p^\star$ have been attained, the trial solution $\Psi_t(x, p^\star)$ is a smooth approximation to the true solution that can be evaluated continuously on the domain.

A schematic of the NN used in this study is shown in Fig. 1.

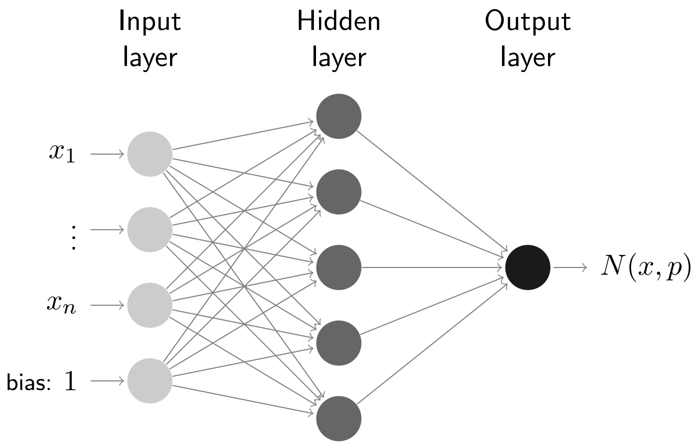


Figure 1: Schematic of NN with $n+1$ input nodes, H hidden nodes, and 1 output node.

There are $n+1$ input nodes (including a bias node) and a single hidden layer of H nodes with sigmoid activation functions. The single scalar output is thus given by

$$N(x, v, \bar{W}) = v^T g(\bar{W}\bar{x}), \tag{5}$$

where $v \in \mathbf{R}^H$ and $\bar{W} \in \mathbf{R}^{H \times n+1}$ are the specific NN parameters (replacing the general parameter representation p). The input variable is $\bar{x} = [x^T, 1]^T$, where the "bar" indicates the appended "1" used to account for the bias at each of the hidden units. The function $g : \mathbf{R}^H \to \mathbf{R}^H$ is a component-wise sigmoid that acts on the hidden layer.

Given the above, the overall task is to choose the discretization $\hat{D}$ and the number of hidden nodes H, and then minimize Eqn. 4 to obtain the approximation $\Psi_t(x, p^\star)$. Assuming a given numerical method that reliably obtains the solution $p^\star$, this leaves the discretization and the hidden layer as basic design choices. Intuitively, it is expected that the solution accuracy will increase with a finer discretization and a larger hidden layer (i.e. NN complexity), but at the expense of computation and possible over fitting. These trends will be explored in the examples. Ultimately, one would like to obtain an approximation of sufficient accuracy by using a minimum of computation effort and NN complexity.

3 EXAMPLES

The method is now showcased for the solution of two sample partial differential equations (PDE). In both examples, $n = 2$ and the domain was taken to be the square $D = [0,1] \times [0,1]$ with a uniform grid discretization $\hat{D} = \{(i/K, j/K)\,; i = 0, \ldots, K\,, j = 0, \ldots, K\}$, where $m = (K+1)^2$. Both backpropagation and the BFGS algorithm were initially implemented to train the parameters. It

[1]These methods may reach a local optimum in Eqn. 4 as opposed to the global optimum.

was discovered that BFGS converged more quickly, so this was ultimately implemented for these final examples. Furthermore, through trial and error it was discovered that including a regularization term in Eqn. 4 provided benefits in obtaining parameters of relatively small magnitudes. Without this term, the parameters occasionally would become very large in magnitude. This regularization term also seemed to provide some marginal benefits in reducing error and convergence time compared to the unregularized implementations.

3.1 Laplace's Equation

The first example is the elliptic Laplace's equation:

$$\nabla^2\Psi(x) = 0, \quad \forall x \in D. \tag{6}$$

The BC's were chosen as

$$\begin{aligned} &\Psi(x) = 0, \quad \forall x \in \{(x_1,x_2) \in \partial D \mid x_1 = 0,\, x_1 = 1, \text{ or } x_2 = 0\} \\ &\Psi(x) = \sin \pi x_1, \quad \forall x \in \{(x_1,x_2) \in \partial D \mid x_2 = 1\}. \end{aligned} \tag{7}$$

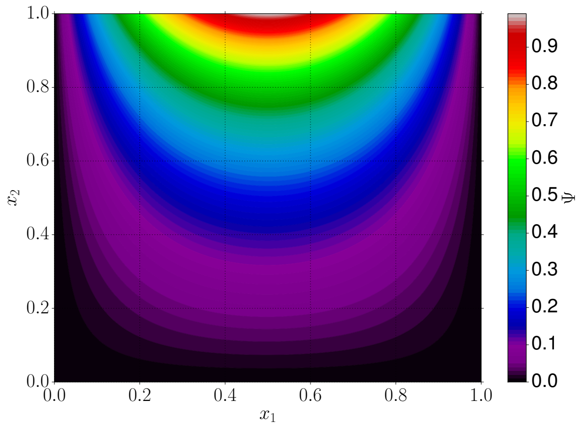
The analytical solution is

$$\Psi(x) = \frac{1}{e^{\pi} - e^{-\pi}} \sin \pi x_1 \left(e^{\pi x_2} - e^{-\pi x_2}\right). \tag{8}$$

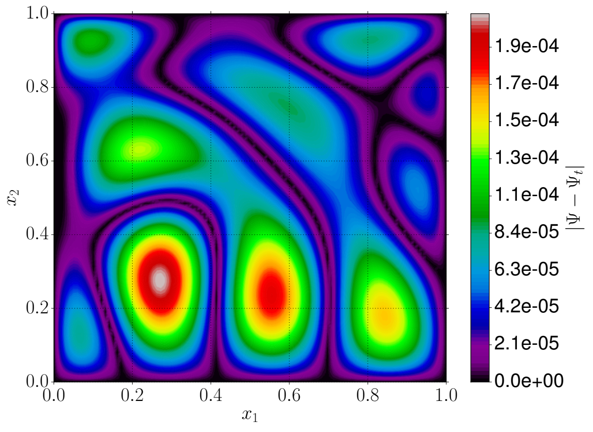
Using the BC's, the trial solution was constructed as

$$\Psi_t(x, v, \bar{W}) = x_2 \sin \pi x_1 + x_1(x_1 - 1)x_2(x_2 - 1)N(x, v, \bar{W}). \tag{9}$$

For the case of $K = 16$ and $H = 6$, the numerical solution and the corresponding error from the analytical solution are shown in Fig. 2. The numerical solution is in good agreement with the analytical solution, obtaining a maximum error of about $2 \cdot 10^{-4}$.



(a) The computed solution $\Psi_t(x, v, \bar{W})$.



(b) The error of the computed solution from the analytical solution: $|\Psi(x) - \Psi_t(x, v, \bar{W})|$.

Figure 2: Solution to Laplace's equation (Eqn. 6) for BC's in Eqn. 7.

3.2 Conservation law

The next example is the hyperbolic conservation law PDE

$$x_1\partial_{x_1}\Psi(x) + \partial_{x_2}\Psi(x) = x_1\,x_2, \quad \forall x \in D, \tag{10}$$

where the BC's were chosen as

$$\Psi(x) = x_1^2 + \exp(-x_1^2), \quad \forall x_1 \in [0,1], x_2 = 0. \tag{11}$$

The analytical solution is

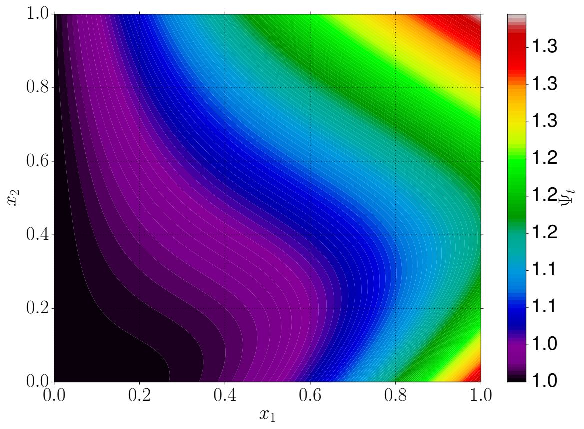
$$\Psi(x) = x_1(x_2 - 1) + x_1^2 e^{-2x_2} + e^{-x_1^2 e^{-2x_2}} + x_1 e^{-x_2}. \tag{12}$$

Using the BC's, the trial solution was constructed as

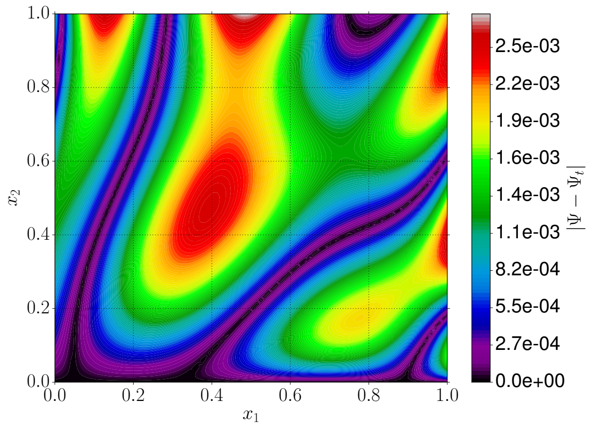
$$\Psi_t(x, v, \bar{W}) = x_1^2 + \exp(-x_1^2) + x_2 N(x, v, \bar{W}). \tag{13}$$

The network parameters were again obtained for $K = 16$ and $H = 6$, and the solution and error is shown in Fig. 3. The numerical and analytical solutions are in good agreement, with a maximum error of about $2.5 \cdot 10^{-3}$.

Although the errors in both examples are small, the error for Laplace's equation is about an order of magnitude smaller than that of the hyperbolic equation. While this may be due in part to the different nature of the solutions, the different BC's may also have an effect. In Laplace's equation the BC's constrain the solution around the entire square domain (since it is second-order in both variables), while in the hyperbolic equation the BC's only constrain the solution along the bottom edge (since it is first order in both variables). Because the solution will automatically hold at the BC's due to the construction of $F(x)$, the BC's along the entire boundary in Laplace's equation most likely contributes to overall smaller error throughout the domain.



(a) The computed solution $\Psi_t(x, v, \bar{W})$.

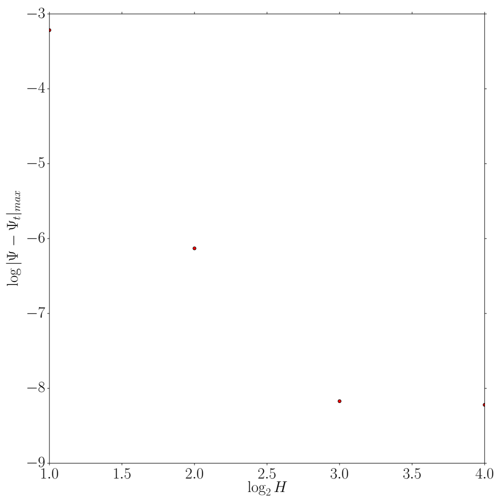


(b) The error of the computed solution from the analytical solution: $|\Psi(x) - \Psi_t(x, v, \bar{W})|$.

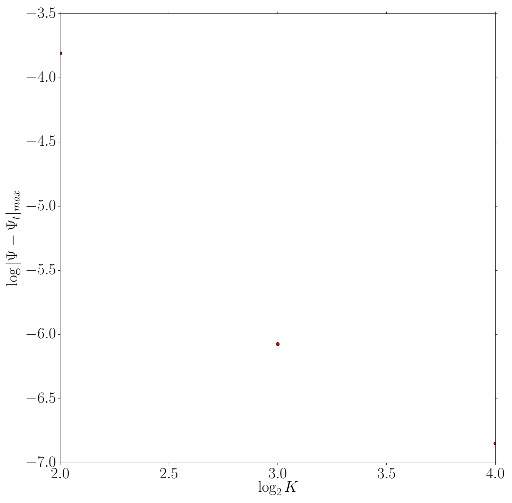
Figure 3: Solution to the hyperbolic conservation law (Eqn. 10) for the BC's in Eqn. 11.

4 ERROR PROPERTIES

As discussed previously, it is intuitively expected that refining the discretization and increasing the size of the hidden layer will increase the accuracy of the solution. To study this, Laplace's equation was solved for a number of choices in K and H, and the maximum error over the domain $|\Psi(x) - \Psi_t(x, v, \bar{W})|_{max}$ for each solution was recorded. To assess the dependence on H, solutions were obtained for $H = 2, 4, 8, \text{and } 16$ for a fixed $K = 8$. To assess the dependence on K, solutions were obtained for $K = 4, 8, \text{and } 16$ for a fixed $H = 4$. The results are shown in Fig. 4. From the first figure, the error steadily decreases for $H = 2, 4, \text{and } 8$ but plateaus for $H = 16$. This suggests that for the given discretization, a network complexity greater than $H = 8$ yields diminishing returns in reducing error. From the second figure, the error steadily decreases with increasing mesh refinement. It is unclear how this trend continues for even finer discretizations of $K > 16$.



(a) Plot of maximum error versus hidden layer sizes $H = 2, 4, 8, \text{and } 16$ for fixed mesh size $K = 8$.



(b) Plot of maximum error versus discretization sizes $K = 4, 8, \text{and } 16$ for fixed hidden layer size $H = 4$.

Figure 4: Error trends for Laplace's equation.

5 CONCLUSIONS AND FUTURE WORK

In this study, a framework for the numerical solution of DE's using NN's has been showcased for several examples. The benefit of this method is that the trial solution (via the trained NN) represents a smooth approximation that can be evaluated and differentiated continuously on the domain. This is in contrast with the discrete or non-smooth solutions obtained by traditional schemes. Although the method has been implemented successfully, there are several areas of possible improvement. Because there is a considerable tradeoff between the discretization training set size (and solution accuracy) and the cost of training the NN, it could be useful to devise adaptive training set generation to balance this tradeoff. Also, this study used a uniform rectangular discretization of the domain, so future studies could explore nonuniform discretizations. This could be especially useful in examples with irregular boundaries, where more sample points might be needed in some regions of the domain compared to others.