NUMERICAL SOLUTION OF FREDHOLM INTEGRAL EQUATIONS OF THE SECOND KIND USING NEURAL NETWORK MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a novel method based on a neural network with one hidden layer and the collocation method for solving linear Fredholm integral equations of the second kind. We first choose the space of polynomials as the projection space for the collocation method, then approximate the solution of a integral equation by a linear combination of polynomials in that space. The coefficients of this linear combination are served as the weights between the hidden layer and the output layer of the neural network while the mean square error between the exact solution and the approximation solution at the training set as the cost function. We train the neural network by the gradient decent method with Adam optimizer and find an optimal solution with the desired accuracy. This method provides a stable and reliable solution with higher accuracy and saves computations comparing with previous neural network approaches for solving the Fredholm integral equations of the second kind.

1 INTRODUCTION

Many problems in engineering and mechanics can be converted into an integral equations. Fredholm integral equations as an important category of integral equations raise naturally in signal and image processing. There are many approaches have been exploited so far for solving integral equation. Two popular ones among them are the collocation method and Galerkin method. In (Liu, Yuzhen, Shen, Lixin, Xu, Yuesheng Yang, Hongqi, 2016) the multiscale collocation method was proposed to solve a Fredholom integral equation of the first kind, which models an image blurring process. As the deep learning neural network getting more popular, a wave of interest in applying it in solving all kinds of mathematical equations emerged recently. Lagaris (I. E. Lagaris, A. Likas, D. I. Fotiadis, 1998) used artificial neural networks to solve ordinary differential equations and partial differential equations. Sohrab Effati and Reza Buzhabadi used neural network to solve Fredholm integral equations of the second kind (Effati, S., Buzhabadi, R, 2012). Since the solution was attained totally by learning, the computational cost of this approach will be expensive. More precisely, the weights of the neural network need to be updated each time and one has to re-compute the integral which involves the product of the kernel and the derivative of the approximated solution with respect to the weights. What's more, when using a numerical method to evaluate the integral, numerical errors will be inevitably introduced and will accumulate with increasing the number of iterations.

The main contribution of this paper is to construct an neural network for the numerical simulation of the integral equation (1) based on a traditional collocation method. Guaranteed by the Weierstrass theorem, the polynomials are dense in L^2 on a compact set. We propose to project the solution of the integral equation onto the space of polynomials first, which coincides with the classical approach, then we use an neural network to train coefficients which can be treated as the weights of the NN model such that the mean square error between the left hand side and the right hand side of the integral equation (1) at the training set is minimized. This approach overcomes the problem of traditional collocation method which requires the number of basis identical to the number of collocation points in order to have a square matrix after the discretization. Furthermore, for the integrals which involve the product of the kernel function and the basis functions, our approach only needs to evaluate them once as long as the number of the basis is fixed. We save the results of these integrals in a matrix which will act as evaluations of our activation function at the input values.

We organize this paper in four sections. In section 2, we present the formulation of the NN-Collocation method for the second kind FIEs. Numerical experiments are provided in Section 3 to demonstrate the performance of the proposed method.

2 NN-COLLOCATION FORMULATIONS

In this work, we consider an integral equation taking a form of

$$u(\mathbf{x}) + \mu \int_S k(\mathbf{x}, \mathbf{y}) u(\mathbf{y})\, d\mathbf{y} = f(\mathbf{x}), \;\; \mathbf{x}, \mathbf{y} \in S \subset \mathcal{R}^l, \tag{1}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_l), \mathbf{y} = (y_1, y_2, \ldots, y_l)$, f is known, and u is unknown. For simplicity, we denote

$$\mathcal{K}(u) = \int_s k(\mathbf{x}, \mathbf{y}) u(\mathbf{y})\, d\mathbf{y},$$

which is called the integral operator. With this notation, the above integral equation can be rewritten as:

$$(\mathcal{I} + \mu\mathcal{K})u = f. \tag{2}$$

where $\mathcal{I}$ is the identity operator. We now describe the traditional collocation method for solving integral equation (1), For $n \in N$, the collocation method seeks vectors $\mathbf{w}_n := [w_j : j \in Z_n]$ such that:

$$u_n = \sum_{j \in Z_n} w_j \psi_j \tag{3}$$

is the solution of

$$(\mathcal{I} + \mu\mathcal{K})u_n(\mathbf{x}_i) = f(\mathbf{x}_i), i \in Z_m \tag{4}$$

where $S_m = \{\mathbf{x}_i : i \in Z_m\}$ is a finite subset of S, ψ_j is the basis function of the projection space. The equivalent system of equation form can be written as:

$$(\mathbf{E}_n + \mu\mathbf{K}_n)\mathbf{w}_n = \mathbf{f}_n \tag{5}$$

where

$$\begin{gathered}\mathbf{E}_n = \{E_{ij}, i \in Z_m, j \in Z_n\}, \\ \mathbf{K}_n = \{K_{ij}, i \in Z_m, j \in Z_n\}, \\ \mathbf{f}_n = \{f_i, i \in Z_m\},\end{gathered}$$

with

$$E_{ij} = \psi_j(\mathbf{x}_i), \;\; K_{ij} = \int_s k(\mathbf{x}_i, \mathbf{y}) \psi_j(\mathbf{y})\, d\mathbf{y} \text{ and } f_i = f(\mathbf{x}_i).$$

The NN-Collocation method based on this idea, try to minimize the cost function which is formulated by the means square error with L^2 regularization:

$$\mathrm{C} = \frac{1}{m} \sum_{i \in Z_m} L_n(\hat{f}(\mathbf{x}_i, \mathbf{w}_n), f_i) + \lambda \|\mathbf{w}_n\|_2^2 \tag{6}$$

with the loss function:

$$L_n(\hat{f}(\mathbf{x}_i, \mathbf{w}_n), f_i) = (\hat{f}(\mathbf{x}_i, \mathbf{w}_n) - f_i)^2 \tag{7}$$

where $\hat{f}(\mathbf{x}_i, \mathbf{w}_n) := (\mathcal{I} + \mu\mathcal{K})u_n(\mathbf{x}_i)$, λ is the regularization parameter, $\|\mathbf{w}_n\|_2$ represents the Euclidean norm of $\mathbf{w}_n$. With the notation used in (5), the loss function (7) of our NN-Collocation model can be rewritten as:

$$L_n(\hat{f}(\mathbf{x}_i, \mathbf{w}_n), f_i) = \left(\sum_{j \in Z_n} (E_{ij} + \mu K_{ij}) w_j - f_i \right)^2 \tag{8}$$

In other words, we use $\{(\mathbf{x}_i, f_i)\}_{i=0}^m$ as our training data set to implement a learning process for a trail model such that the MSE between the predicted value of operator $\mathcal{I} + \mu\mathcal{K}$ and exact value of this operator at the input data is minimized while emphasizing on maximizing margin to avoid overfitting problem.

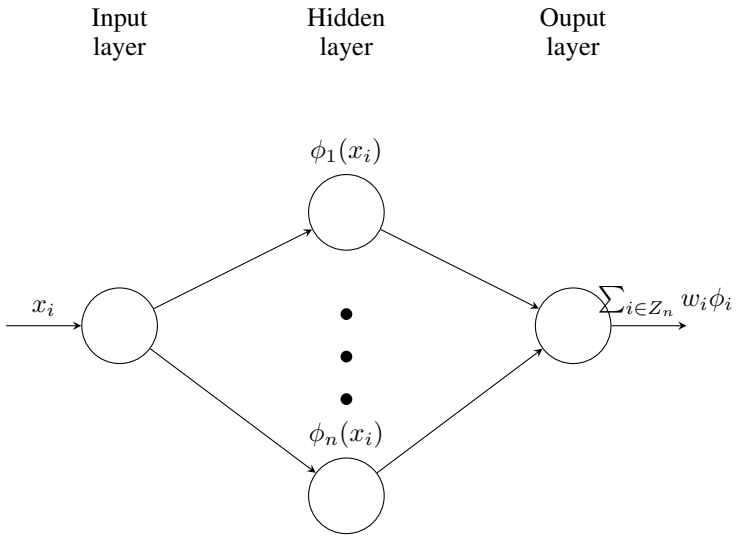


Figure 1: Diagram of NN-collocation model.

The above process can be interpreted as a feed-forward network with one hidden layer. The weights between the input layer and the hidden layer of the network are all set to 1, and the activation function on the jth neuron of the hidden layer is $\phi_j(\mathbf{x}) = (\mathcal{I} + \mu\mathcal{K})\psi_j(\mathbf{x})$, the weights between hidden layer and output layer are w_i's. The network is illustrated in Fig. 1.

We intend to find $\mathbf{w}_n$ by using the gradient descent algorithm. To this end, we need to calculate the $\frac{\partial L_n}{\partial w_j}$ from (8) as follows

$$\frac{\partial L_n}{\partial w_j} = 2(\sum_{j\in Z_n}(E_{ij} + \mu K_{ij})w_j - f_i)(E_{ij} + \mu K_{ij}). \tag{9}$$

The most computing cost is from calculating the K_{ij}, which involves evaluating integrals, but only once. We save these results in a matrix which act as the evaluations of the activation functions at the training data set. In this paper, we use the Gauss-quadrature for the evaluation of K_{ij}. The whole procedure is summarized in Algorithm 1.

Algorithm 1 Our Algorithm

1: **Input** regularization parameter λ, number of training data m, training set $S_m := \{\mathbf{x}_i : i \in Z_m\}$. Initialize the weights $\{w_j : j \in Z_n\}$ by Xavier initialization (Glorot, X. amp; Bengio, Y, 2010).
2: **Compute** $\phi_j(\mathbf{x}_i) := E_{ij} + \mu K_{ij}$, and store the result in matrix A with $A_{ij} = \phi_j(\mathbf{x}_i)$, $i \in Z_m, j \in Z_n$.
3: **Compute** $\frac{\partial C}{\partial w_j} = \frac{1}{m}\sum_{i\in Z_m}\frac{\partial L_n}{\partial w_j} + 2\lambda w_j$ according to chain rule.
4: **Update** the weights w_j with the Adam optimizer.
5: **Compute** the approximation of (3) by using the optimal weights $\{w_i : i \in Z_n\}$.

3 RESULTS

We give four numerical examples to illustrate our model's performance, the first two models are just picked randomly, the last two having polynomials as exact solution. Our projection space are polynomial space with shifted Legendre polynomials on interval $[0, 1]$ of degree up to n as basis. We picked $m = 2000$ numbers equally on interval $[0, 1]$ as our training set. Testing set are equally spaced $m = 1800$ numbers on interval $[0.01, 0.98]$.

Example 1. Consider the following linear FIE of the second kind

$$u(x) - \int_0^1 2e^{x+y}u(y)\,dy = e^x,$$

Table 1: The MSE error for Example 1.

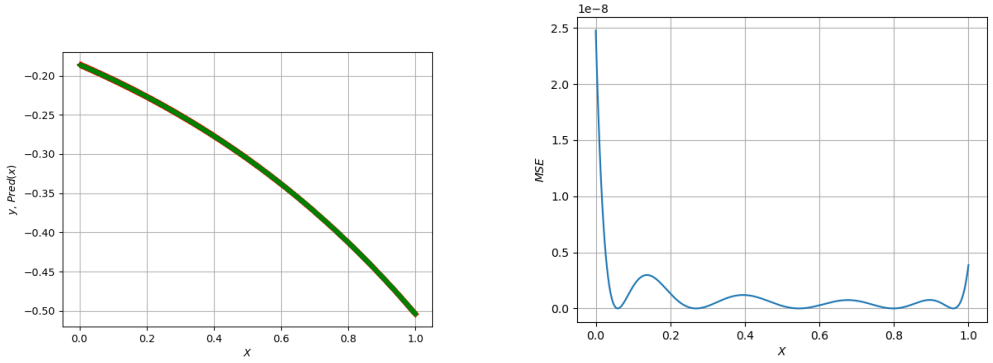
n	MSE on Training set	MSE on Testing set
3	1.09689663e-06	9.59122925e-07
4	1.27886941e-07	1.12720023e-07
5	1.8071884e-08	2.2415795e-08
6	1.0746993e-09	2.98199195e-09
7	1.72614602e-09	1.53534553e-09

with the exact solution $u(x) = \frac{e^x}{2-e^2}$.

Our solution has order of accuracy around $e-09$ for $n = 6$ after 20000 iteration with optimal weights

$$\mathbf{w}_n = [-0.23858496, -0.15568655, -0.11762842, 0.02070347, -0.0158611, 0.00271201].$$

Figure 2(a) shows the exact solution, approximated solution, Figure 2(b) shows the accuracy, which are measured by Mean Square Error (MSE). The numerical results can be found in Table 1.



(a) (b)

Figure 2: Example 1: (a) Exact solution vs NN-Collocation approximate solution with n=6; (b) MSE between the exact solution and the approximation solution.

Example 2. Consider the following linear FIE of the second kind given by (M.H. Reihani, Z.Abadi, 2007)

$$u(x) + \frac{1}{3}\int_0^1 e^{2x-\frac{5}{3}} u(y)\, dy = e^{2x+\frac{1}{3}},$$

with the exact solution $u(x) = e^{2x}$.

Our solution has order of accuracy around $e-11$ for $n = 7$ after 30000 iteration with optimal weights

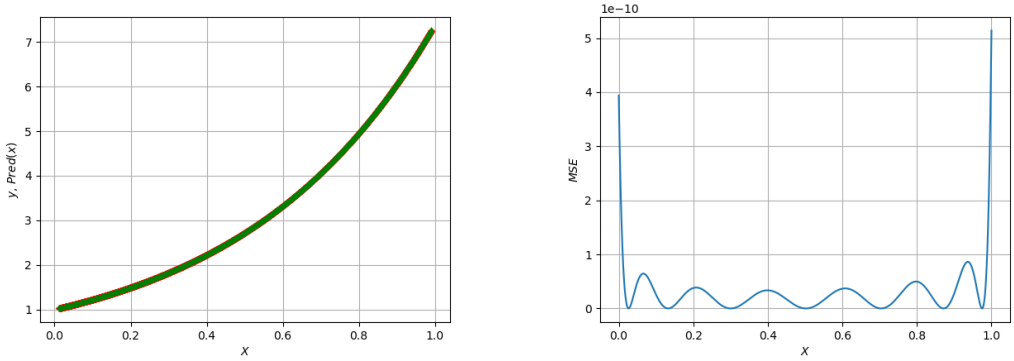
$$\begin{aligned}\mathbf{w}_n \quad = \quad & [1.87982320, 2.75563720, 1.95524998, 5.05200551e-01, \\ & 2.75259395e-01, 5.81048575e-04, 1.72820486e-02].\end{aligned}$$

Figure 3(a) shows the exact solution, approximated solution, Figure 3(b) shows the MSE. The numerical results can be found in Table 2.

Example 3. Consider the following linear FIE of the second kind

$$u(x) + \int_0^1 x(e^{xy} - 1)u(y)\, dy = e^x - x,$$

with the exact solution $u(x) = 1$.



(a) (b)

Figure 3: Example 2: (a) Exact solution vs NN-Collocation approximate solution with n=6; (b) MSE between the exact solution and the approximation solution.

Table 2: The MSE error for Example 2.

n	MSE on Training set	MSE on Testing set
3	0.00474284	0.00475947
4	7.23014255e-05	7.26806487e-05
5	8.22319408e-07	6.70043825e-07
6	8.39516517e-08	1.02247121e-07
7	2.89689923e-11	2.40565254e-11

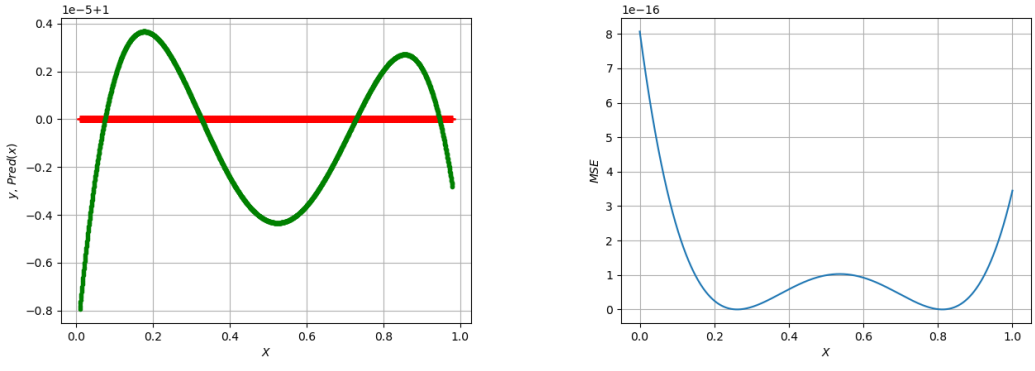
Our solution has order of accuracy around $e-15$ for $n=4$ after 30000 iteration with optimal weights

$$\mathbf{w}_n = [9.99998676e-01, 2.92296511e-06, -2.37639142e-06, 9.37088253e-07]$$

And order of accuracy around $e-16$ for $n=3$ after 50000 iteration with optimal weights

$$\mathbf{u}_n = [9.99999927e-01, 1.43643199e-07, -8.92090066e-08]$$

Figure 4(a) shows the exact solution, approximated solution, Figure 4(b) shows the accuracy measured in MSE. The numerical results can be found in Table 3.



(a) (b)

Figure 4: Example 3: (a) Exact solution vs NN-Collocation approximate solution with n=6; (b) MSE between the exact solution and the approximation solution.

Table 3: The MSE error for Example 3.

n	MSE on Training set	MSE on Testing set
3	1.09178389e-16	9.83884687e-17
4	2.72825599e-15	2.19793864e-15
5	8.72100376e-12	7.84314389e-12
6	7.75215323e-11	6.39991518e-11
7	8.23544625e-11	9.99376595e-11

Table 4: The MSE error for Example 4.

a	n	MSE on Training set	MSE on Testing set
0	3	2.00812521e-14	1.7978776e-14
0	4	2.20219156e-12	2.10087744e-12
5	3	1.25071358e-15	1.12903254e-15
5	4	3.17235264e-14	2.71073112e-14

Example 4. Consider the following linear FIE of the second kind

$$u(x) - \int_0^1 9xyu(y)\,dy = ax^2 - 4x^2,$$

with the exact solution $u(x) = (a-4)x(x-\frac{9}{8})$.

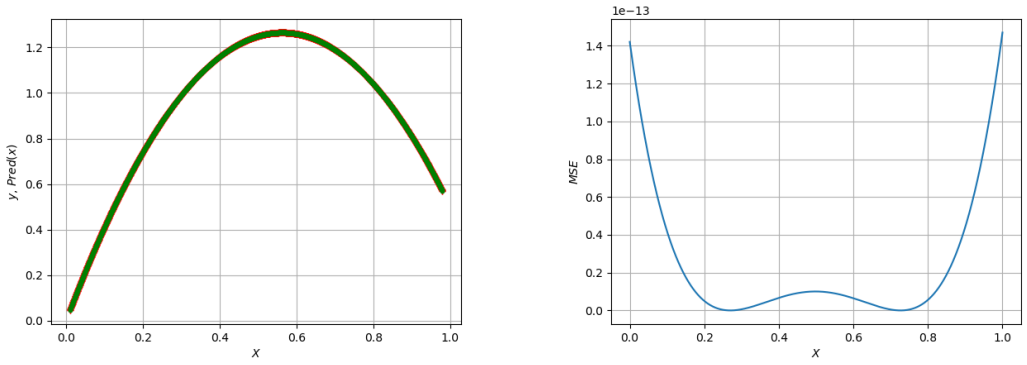
Our solution has order of accuracy around $e-14$ for $a=0, n=3$ after 30000 iteration, with optimal weights

$$\mathbf{w}_n = [-1.33333235, 4.49999805, -2.66666542]$$

And order of accuracy around $e-15$ for $a=5, n=3$ after 30000 iteration, with optimal weights

$$\mathbf{w}_n = [0.33333309, -1.12499951, 0.66666636]$$

Figure 5(a) and Figure 6(a) show the exact and approximated solution for $a=0, n=3$ and $a=5, n=3$, respectively. Figure 5(b) and Figure 6(b) show the corresponding accuracy measured in MSE. The numerical results can be found in Table 4.

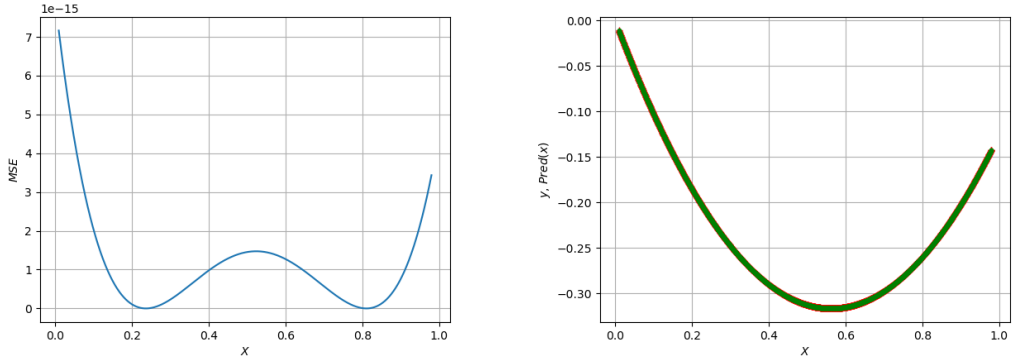


(a) (b)

Figure 5: Example 4: (a) Exact solution vs NN-Collocation approximate solution with $a=0$ and $n=3$; (b) MSE between the exact solution and the approximation solution with $a=0$ and $n=3$.

4 Conclusion

This novel approach for solving linear Fredholm integral equations of the second kind combines the traditional Collocation method and the neuron network learning model, but with different activation



(a) (b)

Figure 6: Example 4: Exact solution vs NN-Collocation approximate solution with a=5,n=3; (b) MSE between the exact solution and the approximation solution with $a = 5, n = 3$.

functions on the same layer, which differs from the classical NN model. The computational cost is low since the involved integrals need to be evaluated once. This NN model works very well on integral equations with polynomial solutions. It is worth pointing out that the accuracy of the model on testing set are better than that on the training set, which means that our model is stable and reliable. The limitation of this approach is that it only works for linear integral equation. For nonlinear integral equation, we can also try a similar way with different activation functions, but that would involve a large amount of computations.

REFERENCES

A collocation method solving integral equation models for image restoration. Journal of Integral Equations and Applications. 28. 263-307. 10.1216/JIE-2016-28-2-263

"Artificial neural networks for solving ordinary and partial differential equations," in IEEE Transactions on Neural Networks, vol. 9, no. 5, pp. 987-1000.

A neural network approach for solving Fredholm integral equations of the second kind. Neural Comput and Applic 21, 843–852.

"Deep Learning," MIT Press

Understanding the difficulty of training deep feed-forward neural networks. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, in Proceedings of Machine Learning Research 9:249-256.

Rationalized Haar functions method for solving Fredholm and Volterra integral equations, Journal of Computational and Applied Mathematics,Volume 200,Pages 12-20,ISSN 0377-0427.