OPEN ACCESS
Computing Open
Vol. 3 (2025) 2550002 (5 pages)

DOI: 10.1142/S2972370125500023

World Scientific
www.worldscientific.com

Recursive Formula for the Trial Function Boundary Function

E. L. Winter

Applied Physics Lab, Johns Hopkins University
7651 Montpelier Road, Laurel, MD 20723, USA
Eric.Winter@jhuapl.edu

R. S. Weigel*

Department of Physics and Astronomy, George Mason University
4400 University Drive, Fairfax, VA 22030, USA
rweigel@gmu.edu

Received 19 December 2024
Accepted 26 January 2025
Published 29 March 2025

The neural network trial function method of Legaris *et al.* (Artificial neural networks for solving ordinary and partial differential equations, *IEEE Trans. Neural Netw.* **9**(5) (1998) 987–1000) requires the specification of a boundary function that matches the boundary values and is finite in the solution domain. We develop a recursive formula for generating a boundary function for up to second-order partial differential equations with Dirichlet boundary conditions in a finite hyper-box domain and with an arbitrary number of dimensions.

Keywords: Differential equations; partial differential equations; neural networks; boundary value; trial function.

1. Introduction

Various neural network-based methods for solving Ordinary Differential Equations (ODEs) and Partial Differential Equations (PDEs) have been developed[1,2] as an alternative to traditional numerical methods, such as finite difference and finite element methods.

The trial function method[3] uses a trial function, ψ_t, of the following form:

$$\psi_t(\mathbf{x}, \boldsymbol{\theta}) = A(\mathbf{x}) + P(\mathbf{x})N(\mathbf{x}, \boldsymbol{\theta}), \tag{1}$$

* Corresponding author.

where $\mathbf{x}$ is a vector of inputs (e.g., t, x, y, z, ...) to the neural network and $\boldsymbol{\theta}$ is a vector of neural network parameters, $A(\mathbf{x})$ is a boundary function, $P(\mathbf{x})$ is a network coefficient function, and $N(\mathbf{x}, \boldsymbol{\theta})$ is the output of the neural network.

The trial function will satisfy the boundary conditions if, when $\mathbf{x}$ is a boundary point, (a) $A(\mathbf{x})$ matches the boundary values and (b) $P(\mathbf{x})$ is zero. $A(\mathbf{x})$ and $P(\mathbf{x})$ may have arbitrary finite values within the boundaries.

Prior works have used *ad hoc* trial functions to find the numerical solution of ODEs or PDEs with restricted dimensionality and/or differential equation order. For ODEs, the authors of Ref. 4 developed boundary functions for ODEs of up to fourth order with Cauchy boundary conditions. Reference 5 used adaptive sampling with the trial function method to solve second-order ODEs and two-dimensional PDEs. The authors of Ref. 6 used trial functions to examine sixth-order ODEs with boundary conditions specified for 0th through second-order derivatives.

For PDEs, the authors of Refs. 7 and 8 modified the trial function approach for irregularly shaped boundaries with Dirichlet boundary conditions. Reference 9 used the trial function method to solve the steady-state two-dimensional Navier–Stokes equations with constant Dirichlet boundary conditions. The authors of Ref. 10 used trial functions to examine the Poisson problem in two dimensions with fixed Dirichlet boundary conditions. The authors of Ref. 11 examined trial functions for coupled sets of one-dimensional thermal stress problems with dynamic Neumann boundary conditions.

Lagari *et al.*[12] derived a general algorithm using matching operators for the boundary function for differential equations up to second order; Dirichlet, Neumann, and Robin boundary conditions; and rectangular hyper-box boundaries. In this work, we develop an alternative algorithm for Dirichlet boundary conditions. The approach described here produces the same boundary function as Ref. 12 but uses a recursive formula rather than matching operators. For problems of high dimensionality, the recursive method may be more straightforward to implement computationally as it does not require symbolic algebra.

2. Constructing the Boundary Function $A(\mathbf{x})$

For a one-dimensional boundary value problem for $\psi(x)$ with $x \in [0, 1]$ and the constants f_0 and f_1 the given Dirichlet boundary conditions at $x = 0$ and $x = 1$, respectively, $A(x)$ can be written as

$$A^{1D}(x) = F_0(x) + F_1(x), \tag{2}$$

where $F_0(x)$ and $F_1(x)$ must be such that $A^{1D}(0) = f_0$ and $A^{1D}(1) = f_1$. This can be satisfied by choosing $F_0(x) = (1 - x)f_0$ and $F_1(x) = xf_1$ so that

$$A^{1D}(x) = (1 - x)f_0 + xf_1. \tag{3}$$

Note that $A^{1D}(x)$ given by Eq. (3) with f_0 and f_1 depending on y satisfies the $x = 0$ and $x = 1$ boundary conditions:

$$A^{1D}(x, y) = (1 - x)f_0(y) + xf_1(y). \tag{4}$$

For a two-dimensional PDE with solution $\psi(x, y)$ in the domain $x, y \in [0, 1]$ with Dirichlet boundary conditions

$$\begin{aligned} \psi(0, y) &= f_0(y), \quad \psi(1, y) = f_1(y), \\ \psi(x, 0) &= g_0(x), \quad \psi(x, 1) = g_1(x), \end{aligned} \tag{5}$$

the straightforward generalization of Eq. (4),

$$A^{2D}(x, y) = A^{1D}(x, y) + (1 - y)g_0(x) + yg_1(x),$$

does not satisfy the 2D boundary conditions given by Eq. (5).

However, by inspection, replacing $g_0(x)$ with $g_0(x) - (1 - x)g_0(0) - xg_0(1)$ and $g_1(x)$ with $g_1(x) - (1 - x)g_1(0) - xg_1(1)$, which gives

$$\begin{aligned} A^{2D}(x, y) = (1 - x)f_0(y) + xf_1(y) + (1 - y)(g_0(x) - (1 - x)g_0(0) - xg_0(1)) \\ + y(g_1(x) - (1 - x)g_1(0) - xg_1(1)), \end{aligned} \tag{6}$$

satisfies the boundary conditions of Eq. (5):

$$\begin{aligned} &x = 0 : A^{2D}(0, y) = f_0(y) + (1 - y)(g_0(0) - g_0(0)) + y(g_1(0) - g_1(0)) = f_0(y), \\ &x = 1 : A^{2D}(1, y) = f_1(y) + (1 - y)(g_0(1) - g_0(1)) + y(g_1(1) - g_1(1)) = f_1(y), \\ &y = 0 : A^{2D}(x, 0) = (1 - x)f_0(0) + xf_1(0) + g_0(x) - (1 - x)g_0(0) - xg_0(1) = g_0(x), \\ &y = 1 : A^{2D}(x, 1) = (1 - x)f_0(1) + xf_1(1) + g_1(x) - (1 - x)g_1(0) - xg_1(1) = g_1(x), \end{aligned}$$

where the continuity conditions

$$\begin{aligned} f_0(0) &= g_0(0), \quad f_1(0) = g_0(1), \\ f_1(1) &= g_1(1), \quad f_0(1) = g_1(0), \end{aligned}$$

were used for the $y = 0$ and $y = 1$ boundaries. Using these conditions and Eq. (4), Eq. (6) can be rewritten as

$$A^{2D}(x, y) = A^{1D}(x, y) + (1 - y)\big(g_0(x) - A^{1D}(x, 0)\big) + y\big(g_1(x) - A^{1D}(x, 1)\big),$$

where A^{1D} given by Eq. (4) was used.

The same procedure can be used to construct the boundary function in higher dimensions. For a three-dimensional problem, with $h_0(x, y)$ and $h_1(x, y)$ being the boundary conditions at $z = 0$ and $z = 1$, we have

$$\begin{aligned} A^{3D}(x, y, z) = A^{2D}(x, y, z) + (1 - z)\big(h_0(x, y) - A^{2D}(x, y, 0)\big) + z\big(h_1(x, y) \\ - A^{2D}(x, y, 1)\big). \end{aligned}$$

Written in general form, for an m-dimensional problem, A^j can be generated using A^{j-1}

$$\begin{aligned} A^1(\mathbf{x}) &= (1-x_1)B^1(\mathbf{x}|x_1=0) + x_1 B^1(\mathbf{x}|x_1=1), \\ A^j(\mathbf{x}) &= A^{j-1}(\mathbf{x}) + (1-x_j)\big(B^j(\mathbf{x}|x_j=0) - A^{j-1}(\mathbf{x}|x_j=0)\big) \\ &\quad + x_j\big(B^j(\mathbf{x}|x_j=1) - A^{j-1}(\mathbf{x}|x_j=1)\big), \end{aligned} \tag{7}$$

where $j = 2, \ldots, m$ and the D in the superscript for A has been dropped; B^j is the boundary function for dimension j (e.g., if $m = 2$, with $x_1 = x$ and $x_2 = y$, $B^1(\mathbf{x}|x_1 = 0) = f_0(y)$, $B^1(\mathbf{x}|x_1 = 0) = f_1(y)$, $B^2(\mathbf{x}|x_2 = 0) = g_0(x)$, and $B^2(\mathbf{x}|x_2 = 0) = g_1(x)$).

$$\frac{\partial \psi}{\partial t} - \frac{\partial^2 \psi}{\partial x^2} = 0, \tag{8}$$

is first-order in time, so only a Dirichlet (initial) condition is needed at $t = 0$. The equation is second-order in space, and thus we need Dirichlet (boundary) conditions at $x = 0$ and $x = 1$. This is a two-dimensional problem, so we must determine $A^1(t, x)$ and $A^2(t, x)$. Assume the initial condition is given by $f_0(x)$ and the boundary conditions by $g_0(t)$ and $g_1(t)$. By treating t as the first dimension, we compute

$$A^1(t, x) = (1 - t) f_0(x) \tag{9}$$

(the $tf_1(x)$ term has been omitted) and so

$$A^2(t, x) = A^1(t, x) + (1 - x)\big[g_0(t) - A^1(t, 0)\big] + x\big[g_1(t) - A^1(t, 1)\big]. \tag{10}$$

3. Summary

We have shown, for second-order partial differential equations with Dirichlet boundary conditions, the boundary function $A(\mathbf{x})$ in the trial function method,[3] which uses a trial function, ψ_t, of the following form:

$$\psi_t(\mathbf{x}, \boldsymbol{\theta}) = A(\mathbf{x}) + P(\mathbf{x})N(\mathbf{x}, \boldsymbol{\theta}), \tag{11}$$

can be represented in a recursive form of Eq. (7), which is an alternative to the method of Lagari *et al.*[12] that requires symbolic algebra.

ORCID

E. L. Winter https://orcid.org/0000-0001-5226-2107
R. S. Weigel https://orcid.org/0000-0002-9521-5228

References

1. S. Chakraverty and S. Mall, *Artificial Neural Networks for Engineers and Scientists: Solving Ordinary Differential Equations* (CRC Press, 2017).
2. N. Yadav, A. Yadav and M. Kumar, *An Introduction to Neural Network Methods for Differential Equations* (Springer Netherlands, 2015).

3. I. Lagaris, A. Likas and D. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, *IEEE Trans. Neural Netw.* **9**(5) (1998) 987–1000.
4. A. Malek and R. S. Beidokhti, Numerical solution for high order differential equations using a hybrid neural network optimization method, *Appl. Math. Comput.* **183**(1) (2006) 260–271.
5. F. Hamza-Lup, I. E. Iacob and J. Orgeron, Finding approximate analytical solutions of differential equations using neural networks with self-adaptive training sets, in *13th Int. Conf. on Electronics, Computers and Artificial Intelligence (ECAI)* (IEEE, 2021), pp. 116–121.
6. A. Verma and M. Kumar, *Multilayer Perceptron Artificial Neural Network Approach to Solve Sixth-Order Two-Point Boundary Value Problems* in Lecture Notes in Networks and Systems (Springer Nature Singapore, 2022), pp. 107–118.
7. I. Lagaris, A. Likas and D. Papageorgiou, Neural-network methods for boundary value problems with irregular boundaries, *IEEE Trans. Neural Netw.* **11**(5) (2000) 1041–1049.
8. K. McFall and J. Mahan, Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions, *IEEE Trans. Neural Netw.* **20** (2009) 1221–1233.
9. M. Baymani, S. Effati, H. Niazmand and A. Kerayechian, Artificial neural network method for solving the Navier–Stokes equations, *Neural Comput. Appl.* **26** (2014) 765–773.
10. W. Guasti Junior and I. Santos, Solving differential equations using feedforward neural networks, in *Computational Science and Its Applications–ICCSA 2021*, eds. O. Gervasi, B. Murgante, S. Misra, C. Garau, I. Blečić, D. Taniar, B. O. Apduhan, A. M. A. C. Rocha, E. Tarantino and C. M. Torre (Springer International Publishing, Cham, 2021), pp. 385–399.
11. T. Hou, P. Zhen, N. Wong, Q. Chen, G. Shi, S. Wang and H.-B. Chen, Multilayer perceptron-based stress evolution analysis under DC current stressing for multisegment wires, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **42** (2023) 544–557.
12. P. Lagari, L. Tsoukalas, S. Safarkhani and I. Lagaris, Systematic construction of neural forms for solving partial differential equations inside rectangular domains, subject to initial, boundary and interface conditions, *Int. J. Artif. Intell. Tools* **29** (2020) 2050009.