# GIẢI PHƯƠNG TRÌNH ĐẠO HÀM RIÊNG SỬ DỤNG MẠNG NEURAL NHÂN TẠO

HO DAC QUAN, HUYNH TRUNG HIEU
*Industrial University Of Ho Chi Minh City;*
*hodacquan@iuh.edu.vn*

**Tóm tắt.** Phương trình đạo hàm riêng đã được ứng dụng rộng rãi trong các lĩnh vực khác nhau của đời sống như vật lý, hóa học, kinh tế, xử lý ảnh vv. Trong bài báo này chúng tôi trình bày một phương pháp giải phương trình đạo hàm riêng (partial differential equation - PDE) thoả điều kiện biên Dirichlete sử dụng mạng neural truyền thẳng một lớp ẩn (single-hidden layer feedfordward neural networks - SLFN) gọi là phương pháp mạng neural (neural network method – NNM). Các tham số của mạng neural được xác định dựa trên thuật toán huấn luyện mạng lan truyền ngược (backpropagation - BP). Kết quả nghiệm PDE thu được bằng phương pháp NNM chính xác hơn so với nghiệm PDE giải bằng phương pháp sai phân hữu hạn.
**Từ khoá.** Phương trình đạo hàm riêng, Mạng neural truyền thẳng 1 lớp ẩn, Thuật toán lan truyền ngược.

# SOLVING PARTIAL DIFFERENTIAL EQUATIONS USING ARTICIFICAL NEURAL NETWORKS

**Abstract.** Partial differential equations have been widely applied in various fields of human knowledge, such as physics, chemistry, economics, image processing, etc. In this paper, we presented a method for solving the problem of partial differential equations (PDEs) with Dirichlet boundary conditions (This method) using single-hidden layer feedforward neural network (SLFN) called neural network method (NNM). The parameters of SLFNs are determined by training the neural network with backpropagation. The results show that NNM can obtain accuracy higher finite difference method.
**Keywords.** Partial differential equations, Single hidden layer feedforward neural network - SLFN, Backpropagation

## 1 GIỚI THIỆU

Trong thực tế các hiện tượng khoa học và kỹ thuật dẫn đến các bài toán giải phương trình đạo hàm riêng (partial differential equation-PDE). PDE thường xuất hiện trong các bài toán ứng dụng trong thực tế như vật lý, kỹ thuật, sinh học, kinh tế, xử lý ảnh v.v. [1, 2]. Vì vậy việc tìm nghiệm của PDE là một yêu cầu quan trọng trong khoa học cũng như thực tiễn. Trong một số trường hợp đơn giản, nghiệm có thể tìm được nhờ vào nghiệm tường minh của bài toán dưới dạng các công thức sơ cấp, các tích phân hay các chuỗi hàm. Tuy nhiên đại đa số các bài toán trong thực tế là các bài toán phi tuyến, các bài toán có miền tính toán phức tạp thì nghiệm tường minh của bài toán không tìm được hoặc quá phức tạp để tìm chúng. Trong những trường hợp đó việc tìm nghiệm của PDE phải dựa vào phương pháp giải gần đúng. Các phương pháp số tìm nghiệm gần đúng thông thường giải PDE như phương pháp phần tử hữu hạn (finite element method - FEM), phương pháp sai phân hữu hạn (finite difference method - FDM). Phương pháp số này có thể xác định nghiệm gần đúng bằng cách thay miền xác định liên tục bằng một số hữu hạn các điểm gọi là các nút lưới của lưới sai phân và tính toán nghiệm PDE tại các nút lưới này [3, 4]. Như vậy để xác định nghiệm gần đúng của PDE tại một điểm bất kỳ trong miền xác định bằng các phương pháp số trên đòi hỏi phải chia lưới miền tính toán chứa nút lưới cần tính và tính giá trị tại các nút lưới của lưới sai phân vừa chia dựa vào các điều kiện biên và điều kiện ban đầu cho trước của PDE.

Để khắc phục việc giải PDE bằng chia lưới miền tính toán nêu trên, một cách tiếp cận để tìm nghiệm gần đúng của PDE dưới dạng một hàm là sử dụng mạng neural (artificial neural network - ANN). ANN là công cụ tìm nghiệm PDE thích hợp nhất dựa vào việc điều chỉnh tham số của mạng bằng cách tăng cường việc huấn luyện mạng [5] . Ưu điểm sử dụng ANN để xác định nghiệm PDE không cần phải tính toán lại giá trị tại các nút lưới trong miền xác định dựa trên các điểm chia trên lưới sai phân đã tính toán trước đó [6]. Ngoài ra ANN có thể xấp xỉ một hàm bất kỳ nếu hàm truyền được chọn một cách thích hợp [7].

Trong bài báo này chúng tôi trình bày mạng neural truyền thẳng một lớp ẩn sử dụng thuật toán huấn luyện lan truyền ngược để xác định nghiệm phương trình đạo hàm riêng với điều kiện biên Dirichlet. Nghiệm NNM của PDE thu được so sánh với nghiệm của nó giải bằng FDM.

Bố cục bài báo được tổ chức như sau: Mục 2 trình bày phương pháp sai phân hữu hạn. Mạng neural truyền thẳng một lớp ẩn để giải PDE được mô tả trong Mục 3. Kết quả thực nghiệm được mô tả ở Mục 4 và cuối cùng là kết luận.

## 2 PHƯƠNG PHÁP SAI PHÂN HỮU HẠN

Cho phương trình đạo hàm riêng cấp 2 với hai biến độc lập [8]

$$G\left(x_1, x_2, u(x_1, x_2), \nabla u(x_1, x_2), \nabla^2 u(x_1, x_2)\right) = 0\ , (x_1, x_2)\ \in\ \Omega \subset R^2, \tag{1}$$

thoả điều kiện biên Dirichlet

$$u(x_1, x_2) = g(x_1, x_2)\ \ (x_1, x_2) \in \partial\Omega$$

$G$, $g$ là các hàm cho trước, $\nabla$ toán tử vi phân, $u(x_1, x_2)$ là nghiệm của (1), $\partial\Omega$ là biên của $\Omega$

Trong mặt phẳng $(x_1,\ x_2)$ cho miền chữ nhật $\Omega$: $\Omega = \{(x_1, x_2) | a < x_1 < b, c < x_2 < d\}$ với a, b, c, d là các số cho trước, $\partial\Omega$ nằm trên các đường thẳng $x_1$ = a, $x_1$ = b, $x_2$= c, $x_2$ = d. FDM là một trong những phương pháp số thường được sử dụng vì tính đơn giản và phổ dụng. Ý tưởng chính của phương pháp này là chia hình chữ nhật $\Omega$ thành M đoạn theo trục $x_1$ và $x_2$ và xấp xỉ các đạo hàm của phương trình (1) sử dụng công thức sai phân. Trong bài báo này minh họa giải PDE bằng FDM với phương trình Poisson sau:

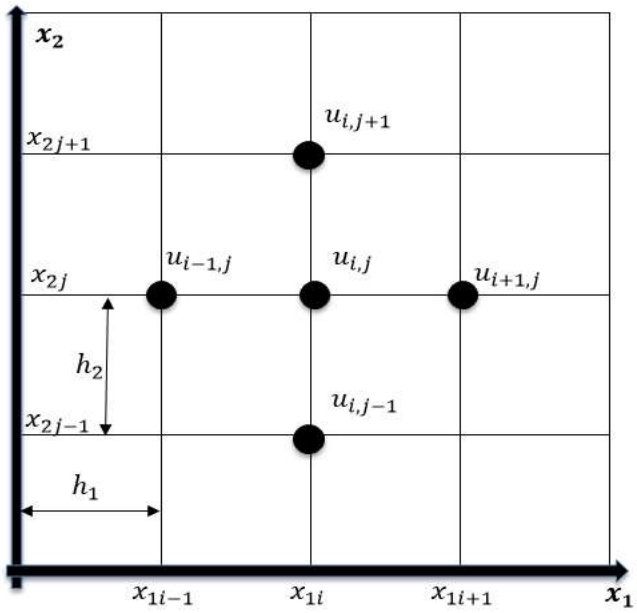
$$\frac{\partial^2 u(x_1,x_2)}{\partial x_1^2} + \frac{\partial^2 u(x_1,x_2)}{\partial x_2^2} = f(x_1, x_2), (x_1, x_2)\ \in\ \Omega\ \subset R^2, \tag{2}$$

thoả điều kiện biên Dirichlet

$u(x_1, x_2) = g(x_1, x_2), (x_1, x_2) \in \partial\Omega$, $f, g$ là các hàm cho trước.

Khi đó chúng ta xấp xỉ đạo hàm cấp hai của phương trình Poisson sử dụng sai phân trung tâm và biến đổi ta có một hệ gồm $(M+1)^2$ phương trình với $(M+1)^2$ ẩn. Giải hệ này ta tính được giá trị tại các nút lưới của lưới sai phân.

### 2.1 Lưới sai phân và hàm lưới




**Hình 1**: Lưới sai phân hữu hạn

Chọn số tự nhiên M>0. Chia miền $\Omega$ thành các ô lưới như sau:

Chia đoạn [a,b] thành M đoạn bằng nhau bởi M+1 điểm chia có tọa độ $x_{1i} = a + (i-1)h_1$, $h_1 = \frac{b-a}{M}$ là độ dài đoạn chia, với *i : 1 → M+1*

Chia đoạn [c,d] thành M đoạn bằng nhau bởi M+1 điểm chia có tọa độ $x_{2i} = c + (i-1)h_2$, $h_2 = \frac{d-c}{M}$ là độ dài đoạn chia. Mỗi điểm $(x_{1i}, x_{2j})$ tương ứng với một nút lưới $(i,j)$, với *i, j : 1 → M+1*

Một hàm số xác định tại các nút lưới gọi là hàm lưới. Giá trị của hàm lưới tại nút lưới $(x_{1i}, x_{2j})$ ký hiệu $u_{i,j}$ xác định bởi :

$$u_{i,j} = u(x_{1i}, x_{2j}), \forall i, j : 1 \rightarrow M+1.$$

Tập $\Omega_{h_1,h_2} = \{(x_{1i}, x_{2j}) | (x_{1i}, x_{2j}) \in \Omega\}$ gọi là tập *nút trong*

Tập $\partial\Omega_{h_1,h_2} = \{(x_{1i}, x_{2j}) | (x_{1i}, x_{2j}) \in \partial\Omega\}$ gọi là tập *nút biên*

## 2.2 Lược đồ sai phân

Từ phương trình $\frac{\partial^2 u(x_1,x_2)}{\partial x_1^2} + \frac{\partial^2 u(x_1,x_2)}{\partial x_2^2} = f(x_1,x_2), a < x_1 < b, c < x_2 < d$, để xấp xỉ đạo hàm cấp hai theo biến $x_1$ và $x_2$ tại điểm $(x_{1i}, x_{2j})$, chúng ta sử dụng công thức sai phân trung tâm ta được

$$\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h_1^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h_2^2} = f_{i,j} \quad (3)$$

Thay (3) vào (2) và biến đổi thành dạng bài toán sai phân hữu hạn hạn

$$\begin{cases} u_{i-1,j} + u_{i+1,j} + \lambda u_{i,j-1} + \lambda u_{i,j+1} - 2(1+\lambda)u_{i,j} = h_1^2 f_{i,j}, \lambda = \frac{h_1^2}{h_2^2} & (4) \\ u_{1,j} = g_a(x_{2j}), \quad j: 1 \rightarrow M+1 & (5) \\ u_{M+1,j} = g_b(x_{2j}), \quad j: 1 \rightarrow M+1 & (6) \\ u_{i,1} = g_c(x_{1i}), \quad i: 1 \rightarrow M+1 & (7) \\ u_{i,M+1} = g_d(x_{1i}), \quad i: 1 \rightarrow M+1 & (8) \end{cases}$$

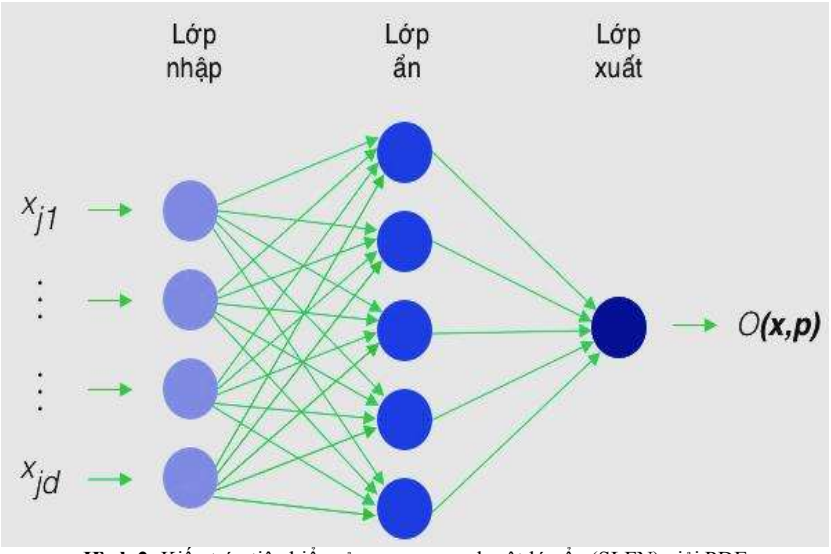
Từ phương trình (4) cho ta mối quan hệ giữa $u_{i,j}$ và bốn nút lưới xung quanh của $u_{i,j}$. Giải hệ phương trình đại số tuyến tính (4-8) ta xác định được giá trị của hàm lưới $u_{i,j}$ với $i,j : 1 \rightarrow M+1$.

# 3 MẠNG NEURAL TRUYỀN THẲNG MỘT LỚP ẨN ĐỂ GIẢI PDE (NNM)

## 3.1 Mạng truyền thẳng một lớp ẩn.

Nhiều kiến trúc khác nhau của mạng neural đã và đang phát triển mạnh. Tuy nhiên người ta đã chứng minh được rằng mạng neural truyền thẳng một lớp ẩn (single hidden layer feedfordward neural network - SLFN) có thể xấp xỉ một hàm bất kỳ nếu số nút ẩn và hàm truyền được chọn một cách thích hợp [7].

Kiến trúc SLFN giải PDE với d nút ở lớp nhập, N nút ở lớp ẩn và 1 nút ở lớp xuất có thể minh họa như hình 2:




**Hình 2**: Kiến trúc tiêu biểu của mạng neural một lớp ẩn (SLFN) giải PDE

Với $\boldsymbol{x_j} = [x_{j1}, x_{j2}, \dots, x_{jd}]^T \in R^d$ là ngõ vào, $w_{mj}$ là véc tơ trọng số kết nối từ các nút nhập *j* đến nút ẩn thứ m, $\mathbf{w}_\mathrm{m} = [w_{m1}, w_{m2}, \dots, w_{md}]^T$, $b_m$ là độ lệch (bias) của nút ẩn thứ m, $\beta_m$ là vector trọng số kết nối từ lớp ẩn thứ m đến nút xuất , $\boldsymbol{w}_m \cdot \boldsymbol{x}_j$ tích vô hướng của hai vecto $\boldsymbol{w}_m$ và $\boldsymbol{x}_j$, $\sigma(.)$ là hàm truyền (trong nghiên cứu này, tác giả sử dụng hàm sigmoid). Ứng với mỗi vector đầu vào $\boldsymbol{x_j}$ ngõ xuất $o_\mathrm{j}$ được xác định

$$o_j = \sum_{m=1}^{N} \beta_m \sigma(\boldsymbol{w}_m \cdot \boldsymbol{x}_j + b_m) \tag{9}$$

### 3.2 Nghiệm PDE sử dụng SLFN

Cho phương trình đạo hàm riêng cấp hai với có dạng tổng quát [9]:

$$G\big(\boldsymbol{x}, u(\boldsymbol{x}), \nabla \mathrm{u}(\boldsymbol{x}), \nabla^2 u(\boldsymbol{x})\big) = 0\,, \boldsymbol{x} \in \Omega \subset R^d, d \in N \tag{10}$$

thỏa điều kiện biên (boundary conditions – BCs) Dirichlete, với $\boldsymbol{x} = (x_1, \dots, x_d) \in R^d$, $u(\boldsymbol{x})$ là nghiệm của phương trình (10) cần xác định. Biên của miền $\Omega$ kí hiệu là $\partial\Omega$.

Nghiệm $u(\boldsymbol{x})$ gần đúng của (10) có thể được xác định bằng cách chia lưới miền $\Omega$ và $\partial\Omega$ và tìm giá trị của $u$ tại các nút lưới thuộc miền $\widehat{\Omega} = \{\boldsymbol{x}_j \in \Omega; j = 1, \dots, n\}$. Rõ ràng việc tìm giá trị u tại các nút lưới thuộc $\widehat{\Omega}$ đưa về giải hệ phương trình (11)

$$G\left(\boldsymbol{x}_j, u(\boldsymbol{x}_j), \nabla \mathrm{u}(\boldsymbol{x}_j), \nabla^2 u(\boldsymbol{x}_j)\right) = 0\,, \boldsymbol{x}_j \in \widehat{\Omega}, \tag{11}$$

$u$ phải thỏa điều kiện biên Dirichlet.

Thay vì xác định giá trị tại các nút lưới thuộc $\widehat{\Omega}$ bằng cách giải hệ phương trình, gọi $u_t(\boldsymbol{x}, \boldsymbol{p})$ là nghiệm gần đúng của phương trình (14), trong đó $\boldsymbol{p}$ là tham số có thể điều chỉnh sao cho cực tiểu hàm lỗi (12)

$$E(\boldsymbol{p}) = \sum_{\boldsymbol{x}_j \in \widehat{\Omega}} \big(G(\boldsymbol{x}_j, u_t(\boldsymbol{x_j}, \boldsymbol{p}), \nabla u_t(\boldsymbol{x_j}, \boldsymbol{p}), \nabla^2 u_t(\boldsymbol{x_j}, \boldsymbol{p}))\big)^2, \tag{12}$$

$u_t$ thỏa điều kiện biên Dirichlet.

Nghiệm gần đúng $u_t(\boldsymbol{x}, \boldsymbol{p})$ được xác định sử dụng SLFN thỏa điều kiện biên Dirichlet có thể viết dưới dạng tổng của 2 số hạng [9], trong đó $\boldsymbol{p}$ là trọng số của SLFN được thay bởi $\boldsymbol{W}, \boldsymbol{\beta}$.

$$u_t(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{\beta}) = A(\boldsymbol{x}) + F(\boldsymbol{x})\, O(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{\beta}), \tag{13}$$

trong đó $O(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{\beta}) = \boldsymbol{\beta^T} \sigma(\boldsymbol{W}\overline{\boldsymbol{x}})$, (14)

với $\overline{\boldsymbol{x}} = [\boldsymbol{x}^\mathrm{T}, 1]^\mathrm{T}$giá trị đầu vào, $\mathbf{w} = [\boldsymbol{w}_1^T, \boldsymbol{w}_2^T, \dots, \boldsymbol{w}_N^T]^T$, $\boldsymbol{b} = [b_1, b_2, \dots, b_N]^T, \boldsymbol{W} = [\boldsymbol{w}, \boldsymbol{b}] \in \mathrm{R}^{\mathrm{Nxd+1}}, \boldsymbol{\beta} = [\beta_1, \beta_2, \dots, \beta_N]^T$ trọng số của SLFN, $O(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{\beta})$ là ngõ xuất của mạng neural, *A(x)* hàm không chứa trọng số của SLFN và thoả điều kiện biên Dirichlet, *F(x)* là hàm không cần thoả điều kiện biên Dirichlet.

Trong bài báo này nghiệm gần đúng $u_t(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{\beta})$ của (10) được xác định sử dụng SLFN với thuật toán lan truyền ngược để cực tiểu hàm lỗi (12) qua việc điều chỉnh trọng số $\boldsymbol{W}, \boldsymbol{\beta}$ của mạng.

Như vậy việc xác định nghiệm gần đúng phương trình (10) tương ứng với việc xác định $u_t(\boldsymbol{x}, \boldsymbol{W}^*, \boldsymbol{\beta}^*)$, với $\boldsymbol{W}^* = argmin_{\boldsymbol{W}} E(\boldsymbol{W}, \boldsymbol{\beta})$, $\boldsymbol{\beta}^* = argmin_{\boldsymbol{\beta}} E(\boldsymbol{W}, \boldsymbol{\beta})$

### 3.3 Gradient của ngõ xuất đối với đầu vào, trọng số của SLFN

Để cực tiểu hàm lỗi (12) SLFN được huấn luyện để điều chỉnh trọng số của mạng sao cho với mỗi giá trị $\boldsymbol{x}_j$ ta có G($\boldsymbol{x}_j$) xấp xỉ với giá trị 0. Một trong các tiếp cận phổ biến để tìm $\boldsymbol{W}, \boldsymbol{\beta}$ thỏa điều kiện trên là dựa trên giảm gradient. Trong đó, ngõ xuất $O(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{\beta})$ và các đạo hàm của ngõ xuất đối với ngõ vào, các trọng số của mạng cần phải được xác định.

#### 3.3.1 Gradient ngõ xuất đối với ngõ vào của SLFN

Xét kiến trúc mạng neural SLFN như hình 2, với đầu vào $\boldsymbol{x} = [x_1, x_2, \dots, x_d]^T \in R^d$, ngõ xuất của mạng được xác định

$$O = \sum_{m=1}^{N} \beta_m \sigma(z_m), \tag{15}$$

với $z_m = \sum_{j=1}^{d} w_{mj}x_j + b_m$

Khi đó ta có:

$$\frac{\partial O}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\sum_{m=1}^{N} \beta_m \sigma\left(\sum_{j=1}^{d} w_{mj}x_j + b_m\right)\right) = \sum_{m=1}^{N} \beta_m w_{mj}^k \sigma^{(1)}, \quad (16)$$

với $\sigma^{(1)} = \frac{\partial \sigma(x)}{\partial x}$

Tương tự với đạo hàm bậc k của $O$ theo biến ngõ vào $x_j^k$ được tính theo công thức

$$\frac{\partial^k O}{\partial x_j^k} = \sum_{m=1}^{N} \beta_m w_{ij}^k \sigma_m^{(k)} \quad (17)$$

với $\sigma_m = \sigma(z_m)$, $\sigma^{(k)}$ là đạo hàm bậc k của hàm sigmoid $\sigma$

Hơn nữa, ta có thể dễ dàng kiểm chứng [9]

$$\frac{\partial^{\lambda_1}}{\partial x_1^{\lambda_1}}\frac{\partial^{\lambda_2}}{\partial x_2^{\lambda_2}}\ldots\frac{\partial^{\lambda_d}}{\partial x_d^{\lambda_d}} O = \sum_{m=1}^{N} \beta_m P_m \sigma_m^{(\Lambda)} \quad (18)$$

với $P_m = \prod_{k=1}^{d} w_{mk}^{\lambda_k}$, $\Lambda = \sum_{m=1}^{d} \lambda_m$ (19)

### 3.3.2 Gradient ngõ xuất đối với các trọng số của SLFN

Dựa vào vế trái công thức (18) ta thấy đạo hàm của ngõ xuất $O$ đối với ngõ vào tương đương với một mạng truyền thẳng 1 lớp ẩn (SLFN). Ngõ xuất của SLFN này gọi là $O_d$ với SLFN có trọng số kết nối từ lớp nhập đến nút ẩn thứ m là $\boldsymbol{w}_m$, độ dịch nút ẩn thứ m là $b_m$, trọng số kết nối nút ẩn thứ m đến ngõ xuất là $\beta_m P_m$ hàm truyền các nút ẩn là $\sigma_m^{(\Lambda)}$, như vậy $O_d = \sum_{m=1}^{N} \beta_m P_m \sigma_m^{(\Lambda)}$. Khi đó từ công thức (18) ta có:

$$\frac{\partial O_d}{\partial \beta_m} = P_m \sigma_m^{(\Lambda)} \quad (20)$$

$$\frac{\partial O_d}{\partial b_m} = \beta_m P_m \sigma_m^{(\Lambda+1)} \quad (21)$$

$$\frac{\partial O_d}{\partial w_{mj}} = x_j \beta_m P_m \sigma_m^{(\Lambda+1)} + \beta_m \lambda_j w_{mj}^{\lambda_j - 1}\left(\prod_{k=1, k\neq j} w_{mk}^{\lambda_k}\right)\sigma_i^{(\Lambda)} \quad (22)$$

## 3.4 Các bước xác định nghiệm của PDE bằng NNM với điều kiện biên Dirichlet

### 3.4.1 Nghiệm phương trình Poisson sử dụng SLFN

Để minh họa cho việc xác định nghiệm của PDE bằng NNM trong bài báo này chọn phương trình Poisson hai chiều có dạng [9] :

$$\frac{\partial^2 u(x_1,x_2)}{\partial x_1^2} + \frac{\partial^2 u(x_1,x_2)}{\partial x_2^2} = f(x_1,x_2), \text{ với } x_1 \in [0,1], x_2 \in [0,1], \quad (23)$$

thoả điều kiện biên Dirichlet

$$\begin{aligned} u(0,x_2) &= f_0(x_2), u(1,x_2) = f_1(x_2), \\ u(x_1,0) &= g_0(x_1), u(x_1,1) = g_1(x_1). \end{aligned} \quad (24)$$

Nghiệm NNM của phương trình (23) xác định bằng SLFN thỏa (24) có dạng [9]:

$$u_t(x_1,x_2,\boldsymbol{W},\boldsymbol{\beta}) = A(x_1,x_2) + x_1(1-x_1)x_2(1-x_2)O(x_1,x_2,\boldsymbol{W},\boldsymbol{\beta}), \quad (25)$$

với $A(x_1,x_2) = (1-x_1)f_0(x_2) + x_1 f_1(x_2)$
$+ (1-x_2)\{g_0(x_1)$
$- [(1-x_1)g_0(0) + x_1 g_0(1)]\} + y\{g_1(x_1)$
$- [(1-x_1)g_1(0) + x_1 g_1(1)]\}$ (26)

Để xác định nghiệm gần đúng $u_t(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{\beta})$ của phương trình (23) tham số $\boldsymbol{W}, \boldsymbol{\beta}$ cần được điều chỉnh bằng cách huấn luyện SLFN sao cho hàm lỗi (27) đạt giá trị nhỏ nhất.

$$E(x_1, x_2, \boldsymbol{W}, \boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{n}\{\frac{\partial^2 u_t(x_{1i}, x_{2i})}{\partial x_1^2} + \frac{\partial^2 u_t(x_{1i}, x_{2i})}{\partial x_2^2} - f(x_{i1}, x_{i2})\}^2, \quad (27)$$

với $(x_{1i}, y_{2i}) \in [0,1]$ x $[0,1]$, $\frac{\partial^2 u_t(x_{1i}, x_{2i})}{\partial x_1^2} =$
$\frac{\partial^2 u_t(x_1, x_2)}{\partial x_1^2}|_{x_1 = x_{1i}, x_2 = x_{2i}}$, với n số mẫu huấn luyện mạng. (28)

### 3.4.2 Các bước xác định nghiệm NNM sử dụng thuật toán huấn luyện lan truyền ngược

Bước 1. Khởi tạo n mẫu $\boldsymbol{x}_j = (x_{1j}, x_{2j}), j = 1,\ldots,n$ .

Bước 2. Khởi tạo giá trị ngẫu nhiên trọng số mạng neural $\boldsymbol{W}, \boldsymbol{\beta}$.

Bước 3. Tính giá trị ngõ xuất $o_j$ của SLFN theo công thức (9).

Bước 4. Tính giá trị của $\frac{\partial E(x_1,x_2,\boldsymbol{W},\boldsymbol{\beta})}{\partial \mathbf{W}}$, $\frac{\partial E(x_1,x_2,\boldsymbol{W},\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$ theo công thức (17), (20-22).

Bước 5. Cập nhật vector trọng số của mạng theo công thức (29).

$$\boldsymbol{W}^k = \boldsymbol{W}^{k-1} - \eta\frac{\partial E(x_1,x_2,\boldsymbol{W},\boldsymbol{\beta})}{\partial \mathbf{W}}, \eta \text{ hệ số học}$$
$$\boldsymbol{\beta}^k = \boldsymbol{\beta}^{k-1} - \theta\frac{\partial E(x_1,x_2,\boldsymbol{W},\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}, \theta \text{ hệ số học} \quad (29)$$

trong đó $\mathbf{W}^k$ , $\boldsymbol{\beta}^k$ là cập nhật trọng số bước lặp thứ k , $E(x_1, x_2, \boldsymbol{W}, \boldsymbol{\beta})$ là hàm lỗi được tính theo công thức (27).

Bước 6. Nếu $E(x_1, x_2, \boldsymbol{W}, \boldsymbol{\beta}) < \varepsilon$ ($\varepsilon$ hằng số dương) nhảy đến Bước 7, ngược lại quay lại Bước 3

Bước 7. Thay bộ trọng số đã tính ở Bước 5 vào công thức (25) để tính nghiệm NNM của phương trình (23). Bộ trọng số này sẽ dùng kiểm tra để đánh giá nghiệm NNM so với nghiệm FDM của phương trình (23).

## 4 KẾT QUẢ THỰC NGHIỆM

Trong phần này chúng tôi minh họa cách xác định nghiệm NNM của PDE bằng thuật giải đã trình bày trong mục 3.32 và so sánh với nghiệm FDM. Lỗi nghiệm NNM được xác định :

*Lỗi nghiệm NNM = Nghiệm giải tích – Nghiệm NNM*
*Lỗi nghiệm FDM = Nghiệm giải tích – Nghiệm FDM* (30)

Xét phương trình PDE sau

$$\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} = 0 \quad (31)$$

với $0 \le x_1 \le 1, 0 \le x_2 \le 1$, với điều kiện biên Dirichlet

$$u(0, x_2) = sin2\pi x_2, u(1, x_2) = 0$$
$$u(x_1, 0) = sin2\pi x_1, u(x_1, 1) = 0 \quad (32)$$

Nghiệm giải tích của phương trình (30) thỏa điều kiên biên Dirichlet (31):

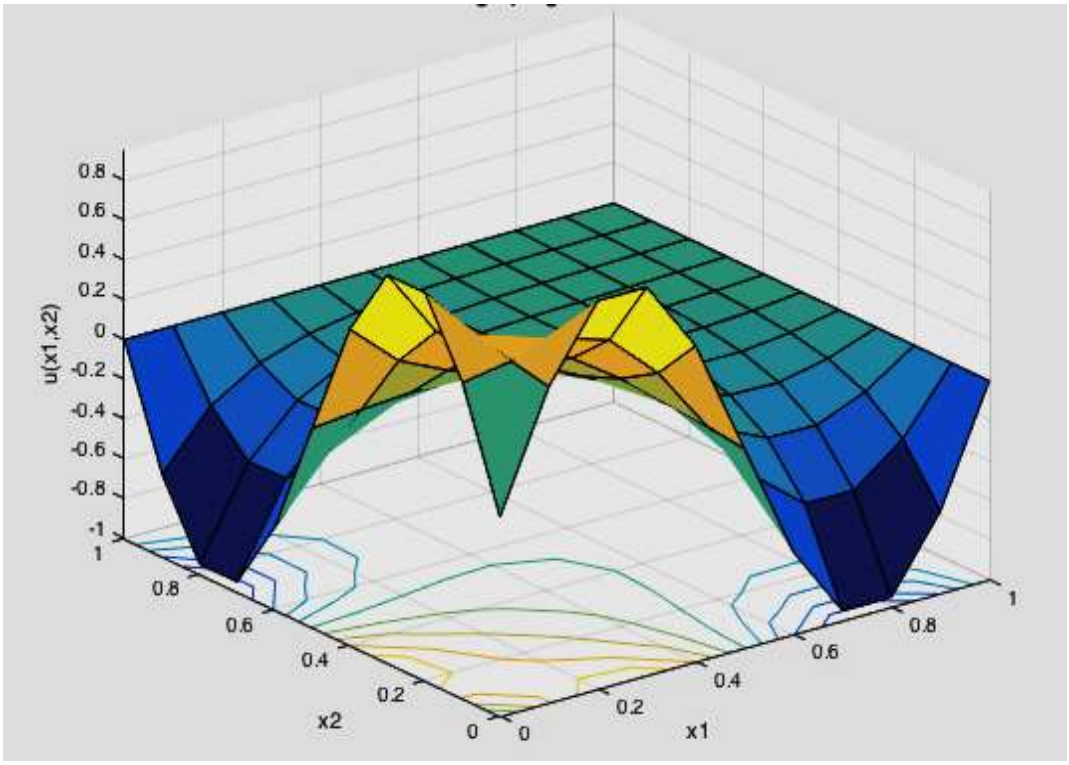
$$u_e(x_1, x_2) = \frac{1}{\sinh 2\pi}(\sinh[2\pi(1-x_2)]\, sin(2\pi x_1) + \sinh[2\pi(1-x_1)]\, sin(2\pi x_2) \quad (33)$$

Dựa vào dạng nghiệm SLFN của phương trình (25), nghiệm NNM của (30) có dạng:

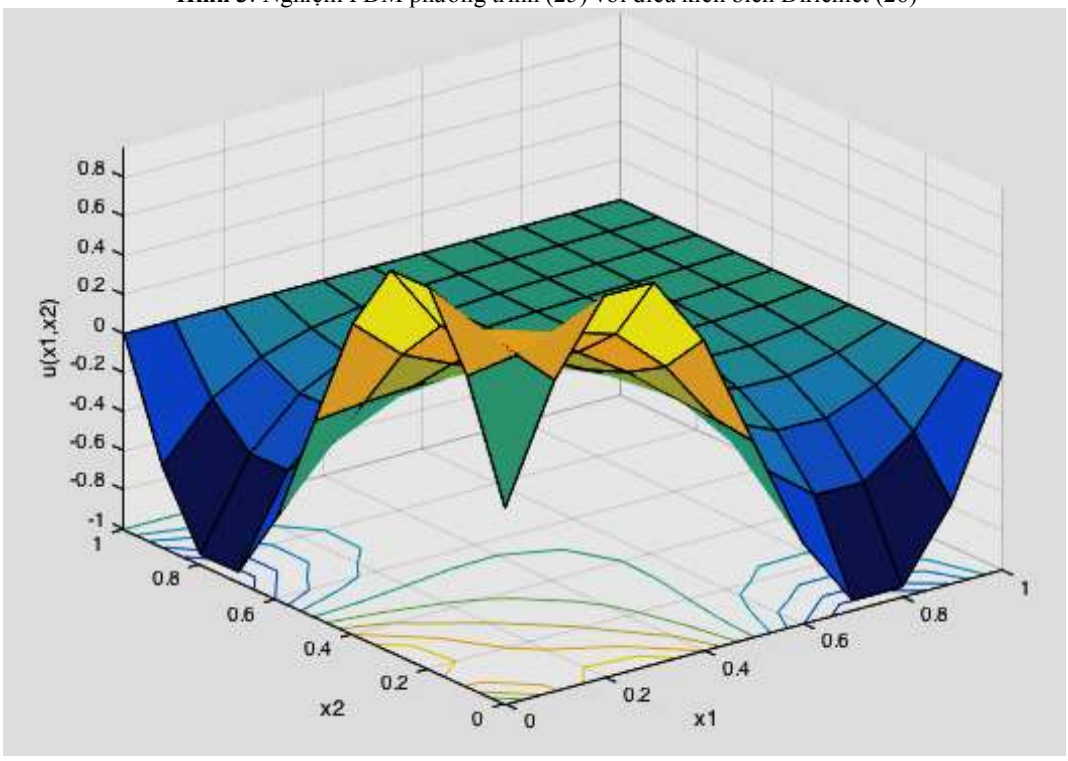
$$u_t(x_1, x_2, \boldsymbol{W}, \boldsymbol{\beta}) = A(x_1, x_2) + x_1(1-x_1)x_2(1-x_2)O(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{\beta}) \quad (34)$$

Trong đó $A(x_1, x_2)$ được tính dự vào công thức (26)

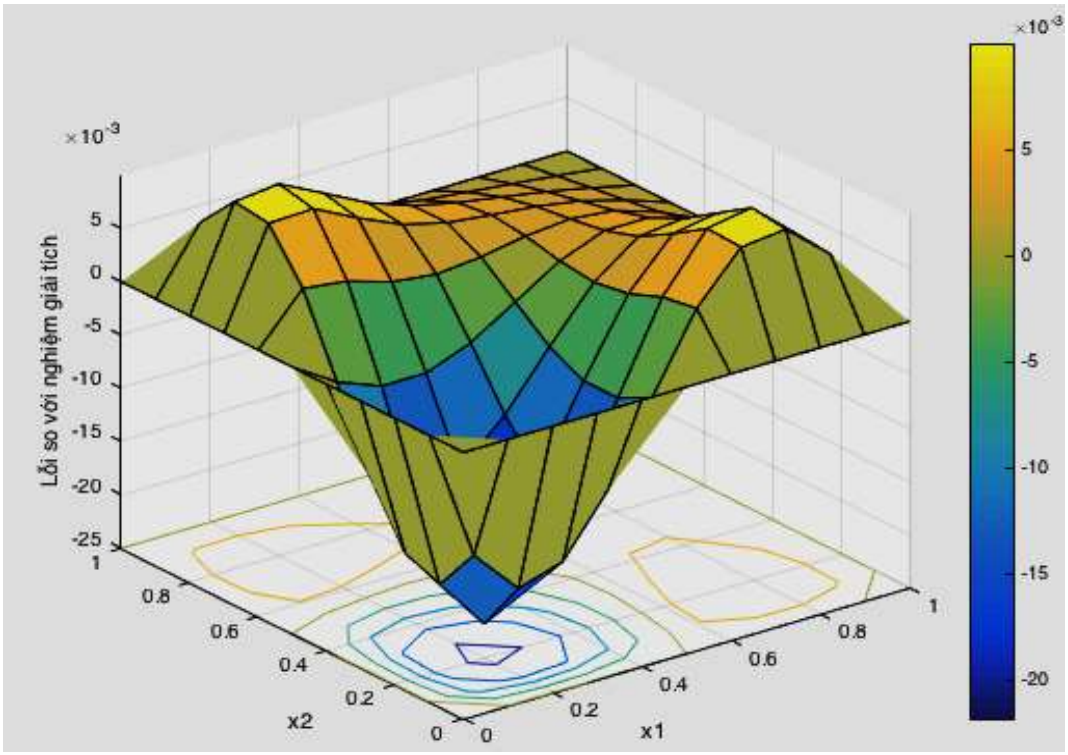
$$A(x_1, x_2) = (1-x_1)sin2\pi x_2 + (1-x_2)sin2\pi x_1 \quad (35)$$



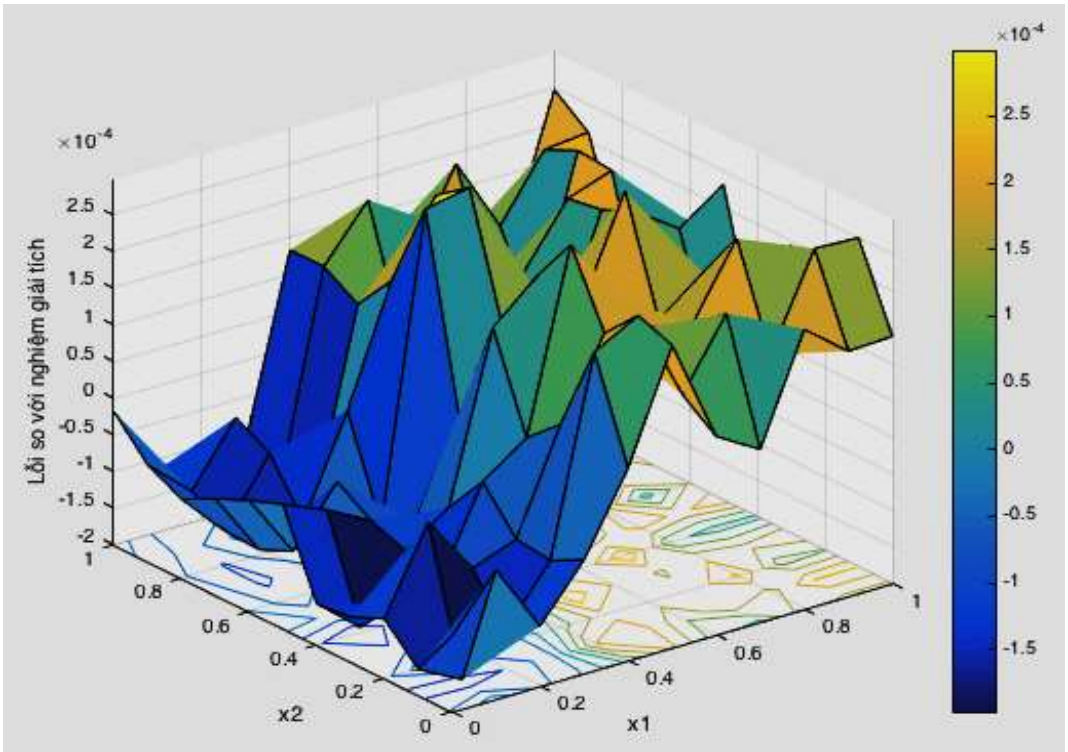

**Hình 3**: Nghiệm FDM phương trình (25) với điều kiên biên Dirichlet (26)




**Hình 4**: Nghiệm NNM $u_t(x_1, x_2, \boldsymbol{W}, \boldsymbol{\beta})$ phương trình (25) với điều kiên biên Dirichlet (26)




**Hình 5**: Lỗi nghiệm FDM và nghiệm giải tích thuộc [$-2.10^{-2}$, $5x10^{-3}$]




**Hình 5**: Lỗi nghiệm NNM và nghiệm giải tích
$u_e(x_1, x_2) - u_t(x_1, x_2, \boldsymbol{W}, \boldsymbol{\beta})$ thuộc [$-1.5.10^{-4}$, $2.5.10^{-4}$]

Để xác định nghiệm gần đúng phương trình (31-32) giải bằng NNM sử dụng SLFN với 10 nút ẩn dựa trên thuật toán huấn luyện lan truyền ngược, tập huấn luyện cho SLFN được chọn 121 điểm (bằng cách chia lưới [0,1]x[0,1] dọc theo $x_1, x_2$ ). Từ đó xác định bộ trọng số $(\mathbf{W}, \boldsymbol{\beta})$ của SLFN và xác định được $u_t(x_1, x_2, \boldsymbol{W}, \boldsymbol{\beta})$. Biểu đồ nghiệm FDM, nghiệm NNM, lỗi giữa nghiệm giải tích so với FDM, lỗi giữa nghiệm giải tích so với NNM của thực nghiệm trên được thể hiện như Hình 3, Hình 4, Hình 5, Hình 6 tương ứng.

Dựa vào hình 5 cho thấy *lỗi nghiệm NNM* của PDE sử dụng NNM $u_t(x_1, x_2, \boldsymbol{W}, \boldsymbol{\beta})$ so với nghiệm giải tích $u_e(x_1, x_2)$ thuộc đoạn $[-1.5.10^{-4}, 2.5.10^{-4}]$ .

Dựa vào hình 6 cho thấy *lỗi nghiệm FDM* của PDE sử dụng FDM so với nghiệm giải tích $u_e(x_1, x_2)$ có lỗi thuộc đoạn $[-2.10^{-2}, 5\text{x}10^{-3}]$.

Kết quả cho thấy nghiệm gần đúng PDE giải bằng NNM chính xác hơn nghiệm gần đúng giải bằng FDM.

## 5 KẾT LUẬN

Trong bài báo này, chúng tôi trình bày phương pháp NNM để giải phương trình Poisson hai chiều với điều kiện biên Dirichle sử dụng mạng truyền thẳng một lớp ẩn với thuật toán huấn luyện mạng lan truyền ngược. Các tham số của SLFN được cập nhật theo công thức (24) bởi một số bước lặp. Với phương pháp sử dụng NNM thì nghiệm $u_t(x_1, x_2, \boldsymbol{W}, \boldsymbol{\beta})$ cho kết quả chính xác hơn so với phương pháp FDM. Các tham số cần tìm của NNM ít hơn rất nhiều so với việc tính giá trị tại các nút lưới của lưới sai phân sử dụng FDM và yêu cầu về bộ nhớ tính toán ít hơn hẳn so với FDM [10] . Phương pháp NNM để giải PDE có dạng nghiệm tổng quát có thể áp dụng cho phương trình vi phân, hệ phương trình vi phân và các PDE dạng khác [9]. Hướng phát triển tiếp theo của nghiên cứu là sử dụng mạng tích chập (convolutional neural network) để giải PDE nhằm tăng độ chính xác nghiệm của PDE và sử dụng mạng học sâu (deep learning) để giải các dạng PDE bậc cao.

## LỜI CẢM ƠN

Bài báo được thực hiện với sự hỗ trợ từ quỹ đề tài nghiên cứu khoa học trường đại học Công nghiệp TP HCM, mã số 182.CNTT01/HD-DHCN

## TÀI LIỆU THAM KHẢO

[1] Ricardo, H.J., A modern introduction to differential equations2009: Academic Press.

[2] Boyce, W.E., R.C. diprima, and D.B. Meade, Elementary differential equations and boundary value problems. Vol. 9. 1992: Wiley New York.

[3] Smith, G.D., Numerical solution of partial differential equations: finite difference methods1985: Oxford university press.

[4] Dill, E.H., The finite element method for mechanics of solids with ANSYS applications2011: CRC press.

[5] Demuth, H., M. Beale, and M. Hagan, MATLAB User's Guide, version 4.0: Neural network toolbox. Mathworks Inc.: Natick, MA, USA, 2005.

[6] Shirvany, Y., M. Hayati, and R. Moradian, Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations. Applied Soft Computing, 2009. **9**(1): p. 20-29.

[7] Huang, G.-B. And H.A. Babri, Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Transactions on Neural Networks, 1998. **9**(1): p. 224-229.

[8] Thomas, J.W., Numerical partial differential equations: finite difference methods. Vol. 22. 2013: Springer Science & Business Media.

[9] Lagaris, I.E., A. Likas, and D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations. IEEE transactions on neural networks, 1998. **9**(5): p. 987-1000.

[10] Rudd, K., G. Di Muro, and S. Ferrari, A constrained backpropagation approach for the adaptive solution of partial differential equations. IEEE transactions on neural networks and learning systems, 2014. **25**(3): p. 571-584.

*Ngày nhận bài: 03/05/2019*
*Ngày chấp nhận đăng: 20/06/2019*