Solving the Poisson Equation using Virtual Element Method and Artificial Neural Networks

Paulo Akira Figuti Enabe[1], Rodrigo Provasi[1]

[1]*Dept. of Structural and Geotechnical Engineering, Polytechnic School, Universisty of São Paulo*
Avenida Professor de Almeida Prado tv. 2 n.83, 05508070, São Paulo, Brasil
paulo.enabe@usp.br, provasi@usp.br

Abstract. The Virtual Element Method (VEM) is a recent method that proposes a generalization of the classical Finite Element Method (FEM). VEM formulation is complex and requires a strong mathematical base. VEM models are able to use any simple polygon (convex or non-convex polygons) as mesh discretization elements, which leads to the functions associated with each of those elements being not strictly polynomials. Thus, the core proposal of VEM is to compute those functions implicitly, by projecting then in a polynomial space. The Virtual Element Method was originally formulated for the Poisson Equation, which is largely explored both in mathematics and natural sciences due to its versatility.

Approximation methods like VEM are widely applied to solve partial differential equations (PDE). A less common approach is to use artificial intelligence techniques like Artificial Neural Networks (ANNs). Solve PDEs with ANNs is not new, but with the increasing popularity of deep learning techniques and new frameworks (e.g Pytorch and Keras), this approach is becoming more relevant. In the present work, the Poisson Equation is transformed into an optimization problem and them ANNs are used to compose a trial solution, aiming to obtain the best one (i.e., the one that reduces the error as much as possible). Artificial Neural Networks are widely used in this kind of approach since they are very popular in classification problems.

Throughout the article, the formulation of VEM and the ANNs approach is presented, highlighting their main characteristics. The implementation of both approaches is discussed and the results are compared.

Keywords: virtual element method, artificial neural networks, poisson equation

1 Introduction

The Virtual Element Method (VEM) proposes to generalize the classical Finite Element Method (FEM) with respect to the mesh discretization. Any simple polygon (convex or non-convex polygons) can be used as a discretization element with VEM. Consequently, the shape functions are not restricted to polynomials. In this sense, the main idea of VEM is to compute these functions implicitly, without knowing their explicit form. Also, accordingly to Mengolini et al. [1], the method has a good performance with complex geometries and is more flexible than the FEM regarding to the mesh quality. These characteristics make the Virtual Element Method a powerful tool to work with engineering problems. For example, there are already application of the method in fluid mechanics (see da Veiga et al. [2]), topology optimization (see Paulino and Gain [3], Antonietti et al. [4]) and contact problems (see Wriggers et al. [5], Aldakheel et al. [6]).

The original formulation of the Virtual Element Method is based on the Poisson Equation as it can be seen in the canonical papers da Veiga et al. [7] and da Veiga et al. [8]. Thus, the method applied to the Poisson Equation is well established in the literature. For example, in Sutton [9], a implementation of the VEM in MATLAB is presented illustrating the main characteristics of the method's implementation. And in Ortiz-Bernardin et al. [10], the VEM application for the Poisson Equation is given as particularization of the linear elasticity formulation. Also, the mathematical aspects of the method are explored in da Veiga et al. [11] where the stability and mesh regularization properties are also discussed, and in da Veiga et al. [12] where the method is formulated for general elliptic problems. From literature, the systematization of the VEM applied to Poisson Equation is given by:

1. Discretization of the geometry;

2. Construction of the virtual element space;
3. Introduction of the projection operator;
4. Construction of the local stiffness matrix;
5. Construction of the load vector.

The steps are discussed in details on da Veiga et al. [7] and in the next section a brief presentation of them is made.

A less common approach regarding to solve partial differential equation (PDE) is the usage of machine learning techniques, more specifically artificial neural networks (ANNs). This approach is not new, but it has been put aside because more powerful techniques have been developed at the time. Since machine learning is one of the trend topics in computer science, ANNs applied to differential equations are being revisited fueled by new deep learning frameworks like *Tensorflow* and *Pytorch*.

In Lagaris et al. [13], a formulation to solve differential equations with boundary conditions is proposed. The authors introduce a trial solution in which the first part is related to the initial and boundary condition and has no modifiable parameter. The second part is related to the artificial neural network with modifiable parameters and it is constructed in way not to affect the boundary conditions. The used network is a feedforward neural network and the modifiable parameters are called weights. In this work, this approach was chosen once the implementation of this formulation is not difficult and can be adapted to be used with popular deep learning frameworks. In the last years other approaches using more robust networks were presented as in Tang et al. [14] that use convolutional neural networks and in Sirignano and Spiliopoulos [15] where a neural networks and Galerkin method are used together.

The main objective of this work is to compare the Virtual Element Method and the artificial neural network approach regarding the Poisson Equation. The formulation of both approaches are briefly presented and, then they are applied to a problem with analytical solution. In this way, it will be possible to highlight the main characteristic of each method.

2 The Poisson Equation

Let Ω be a polygonal domain. The continuous Poisson Equation (also called strong formulation) is given with Dirichlet boundary condtions is given by:

$$\begin{cases} -\Delta u = f, \quad in \quad \Omega \\ u = 0, \quad on \quad \partial\Omega \end{cases}, \tag{1}$$

where $u \in C^2(\Omega) \cap C(\overline{\Omega})$ and $f \in L^2(\Omega)$. The solution u is very demanding once it is to find a function that the first two derivatives are continuous. In this way the idea is reformulate the problem to obtain less restrictive solutions. Choosing a test function $v \in H_0^1(\Omega)$ and multiplying eq. (1), one obtains:

$$\int_\Omega (-\Delta u - f)v dx = 0 \Rightarrow \int_\Omega \nabla u \cdot \nabla v dx - \int_\Omega f v dx = 0 \Rightarrow \int_\Omega u \cdot v dx = \int_\Omega f v dx. \tag{2}$$

It is important to mention that $H_0^1(\Omega)$ is the closure of the space of $C_c^\infty(\Omega)$ in the Sobolev space $H^1(\Omega)$ and $L^2(\Omega)$ is the Lebesgue space. Equation (2) is called weak form of the Poisson Equation, once it is only required to the solution u be one time derivable and integrable, i.e., $u \in H_0^1(\Omega)$.

It is also possible to rewrite the problem defining the bilinear form as

$$a(u, v) = \int_\Omega u \cdot v dx. \tag{3}$$

This operator also defines the inner product associated to $H_0^1(\Omega)$.

3 The Virtual Element Method

In this section, the Virtual Element Method formulation for the Poisson Equation is briefly presented. This formulation is based on the canonical papers [7] and [8]. For more details, the reader can refer to these papers.

The Virtual Element Formulation starts from the weak form of Poisson Equation given by eq. (2). The goal is introduce a discrete form of the weak formulation by choosing a decomposition T_h of Ω into simple polygons (convex or non-convex polygons) E and defining h as the maximum polygonal diameter. Thus, the discrete problem is given by

$$a_h(u_h, v_h) =< f_h, v_h >, \tag{4}$$

where $u_h, v_h \in V_h \subset H_0^1(\Omega)$, $f_h \in V_h'$, V_h is called virtual element space and V_h' is the dual space of V_h. The idea is to define the virtual element space $V_{h,E} = V_h|_E$ for each polygon E, build the discrete bilinear form a_h and build the load term f_h.

3.1 Virtual Element Space

One important hypothesis is that the polynomial space of k order of accuracy $\mathbb{P}_k(E)$ is contained in the local virtual element space $V_{h,E}$, defined as

$$V_{h,E} = \{v_h \in H_0^1(E) : v_h|_{\partial E} \in \mathbb{B}(\partial E), \quad \Delta v_h|_E \in \mathbb{P}_{k-2}(E)\}, \tag{5}$$

where $\mathbb{B}(E) = \{v_h \in C^0(\partial E)|\ \ v_h|_e \in \mathbb{P}_k(e),\ \ \forall e \in \partial E\}$, e is an edge of polygon E and ∂E is the boundary of polygon E. Basically, as the shape functions are not known in the first moment, the choice of the space $\mathbb{B}(E)$ is made such that these functions behave like polynomials on ∂E. As can be seen, the virtual element space has both polynomials and non-polynomial functions.

If $k = 1$, the virtual element space becomes a harmonic function space because $v_h|_E \in \mathbb{P}_{-1}(E) = \{0\}$. In this sense, it is possible to prove that the function are defined by their values at the vertices of the polygon. Now, if $k = 2$, the Laplacian is equal to a constant once $\Delta v_h \in \mathbb{P}_0(E)$. Therefore, the function v_h is defined by its value in each vertex and on one point on the edges. For $k = 3$, to define v_h it also necessary the value in the internal points. Thus, the degrees of freedom are chosen accordingly and can be verified in [7]. The choice of degrees of freedom are made in order to define an unique $v \in V_{h,E}$.

3.2 The Bilinear Form

To build the bilinear form the idea is to introduce a projection operator $\Pi^\nabla : V_{h,E} \longrightarrow \mathbb{P}_k(E)$ which is responsible for projecting elements from the virtual element space to the polynomial space. In this way, the local virtual element space can be written as $V_{h,E} = \Pi^\nabla V_{h,E} + (1 - \Pi^\nabla)V_{h,E}$. Two conditions must be satisfied in order to find a discrete bilinear form $a_{h,E}$:

Hypothesis 1. *For each integer $k \geq 1$ and for all $E \in T_h$:*

- ***Consistency**: it holds that $a_{h,E}(q, v) = a_E(q, v)$, for all $q \in \mathbb{P}_k$ and for all $v \in V_{h,E}$,*
- ***Stability**: there exists constants $C_1, C_2 \geq 0$ independent of h and E such that $C_1 a_E(v, v) \leq a_{h,E}(v, v) \geq C_2 a_E(v, v)$, for all $v \in V_{h,E}$.*

The first choice for the discrete bilinear form would be $a_{h,E}(u, v) = a_E(\Pi^\nabla u, \Pi^\nabla v)$. Although this choice does not satisfy the stability condition because as the functions are no entirely polynomials, the projection residues make it impossible to find the constants C_1 and C_2. Thus, a term to handle the residues and guarantee stability is introduced:

$$S_E(u_h, v_h) = \sum_{i=1}^{N_{dof}} dof_i(u_h) dof_i(v_h), \tag{6}$$

where N_{dof} is the total number of degrees of freedom and $dof_i : V_{h,E} \longrightarrow \mathbb{R}$ gives value of the i-th degree of freedom. It can be seen that S_K is a symmetric bilinear form and other choices regarding to its definition can be made as discussed in da Veiga et al. [11]. Finally, the discrete bilinear form is given by:

$$a_{h,E}(u_h, v_h) = a_E(\Pi^\nabla u_h, \Pi^\nabla v_h) + S_E(u_h - \Pi^\nabla u_h, v_h - \Pi^\nabla v_h). \tag{7}$$

3.3 Load term

The load term can be constructed using a similar idea to the bilinear form. The idea is to project function from $L^2(E)$ spaces into polynomial spaces $\mathbb{P}_k(E)$ using the projector $L_k^E : L^2(E) \longrightarrow \mathbb{P}_k(E)$. As in this work, only the linear case is being considered, for $k = 1$ the load vector is given by:

$$\langle f_h, v_h \rangle = \sum_{E \in T_h} L_0^K \frac{1}{n_E} \sum_{i=1}^{n_E} v_h(V_i), \tag{8}$$

where $\{V_i\}_{i \in [1, n_E]}$ is the set o vertices of the polygon E and n_E is the number of vertices. In this case, the term f_h is pieceswise constant and can be computed directly once the values of v_h in the vertices are known.

4 Artificial Neural Network Approach

Differently from the VEM, with the Artificial Neural Network approach the start point is strong formulation given by eq. (1) A trial solution u_t is chosen to satisfy the boundary condition. Simultaneously, the solution must solve the Poisson Equation in the interior of the domain by learning a parameter w, called weight. The form of the trial solution is:

$$u_t(x, w) = BC(x) + P(x, NN(x, w)), \tag{9}$$

where BC is the function responsible to handle the boundary conditions, P is responsible to handle the solution in Ω and NN is a feedforward neural network with modifiable parameter w. As in eq. (1), $u = 0$ on $\partial\Omega$, then $BC(x) \equiv 0$, for all $x \in \partial\Omega$.

Using the trial solution in eq. (9), the idea is to minimize the loss function given by:

$$E(w) = \min_w \sum_i \{-\Delta u_t(x_i, w) - f(x_i)\}^2, \tag{10}$$

where $x_i \in \Omega$ are discrete points. In order to optimize this minimization and using the formulation in Aggarwal and Ugail [16], the following neural network is considered:

$$NN(x) = \sum_{i=1}^{S} e_i \mathcal{S}(z_i), \tag{11}$$

where $\mathcal{S}$ is the Sigmoid function, S is the number of units associated to Sigmoid functions, e_i is the external weight (weight from hidden unit i to the output value) and

$$z_i = \sum_{j=1}^{n} w_{ij} x_j + b_i. \tag{12}$$

In this notation, w_{ij} are the internal weights (weights from input unit j to the hidden layer i) and b_i is the bias of the unit i. Thus, the gradient can be computed as:

$$\frac{\partial^k NN}{\partial x_j^k} = \sum_{i=1}^{S} e_i w_{ij}^k \mathcal{S}_i^{(k)}. \tag{13}$$

Considering a simplified model similar to a perceptron, the artificial neural network architecture is shown in Fig. 1. In practical terms, the model is based on a feedforward network with a backward propagation and gradient descent.

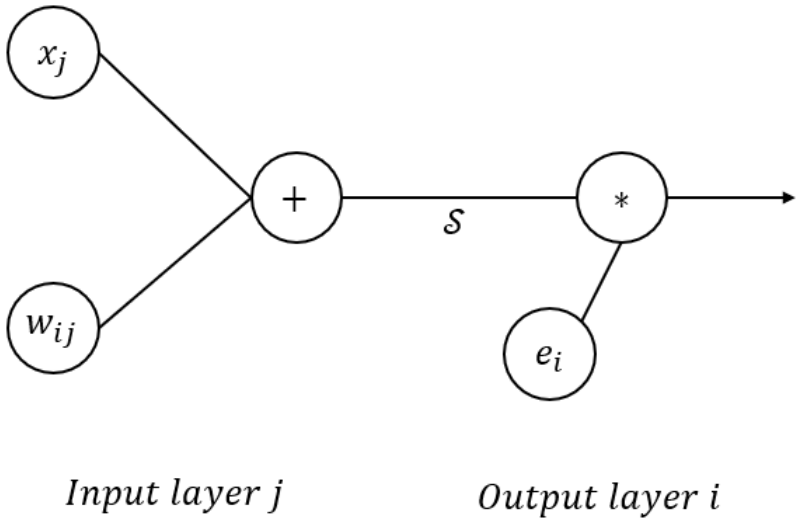


Figure 1. Artificial Neural Network Architecture

5 Results

The geometrical domain is a unitary square plate and the Poisson Equation is given by:

$$\begin{cases} \Delta u = -\sin(\pi x_1)\sin(\pi x_2), & in \quad \Omega \\ u = 0, & on \quad \partial\Omega \end{cases}. \tag{14}$$

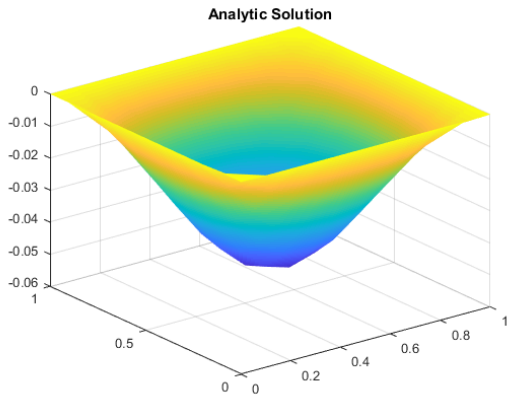
The analytical solution is:

$$u(x_1, x_2) = -\frac{\sin(\pi x_1)\sin(\pi x_2)}{2\pi^2}. \tag{15}$$

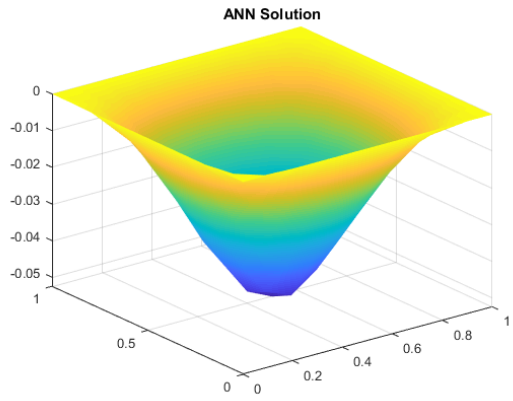
For the Virtual Element Method a uniform square mesh with 100 elements and 121 nodes is used. And for the Artificial Neural Network approach a discretization with 121 points is used. The chosen hyperparameters to train the neural network are the learning rate equals to 0.001, the number of epochs equals to 100 and one hidden layer. Recalling that hyperparameters are attributes that are not estimated from the input data and most of the times they are chosen empirically.

Figure 2 shows the plot of the analytical, VEM and ANN solutions. Figure 3 shows: (a) the deviation between the analytical solution and the ANN approach; (b) the deviation between the analytical solution and the VEM approach; (c) the deviation between the ANN and the VEM approach. It can be seen that for the same number of degrees of freedom, the Virtual Element Method presents a smaller error than the Artificial Neural Network approach. This may occur because the ANNs depends on the calibration of hyperparametrs that might not be in their optimal values.

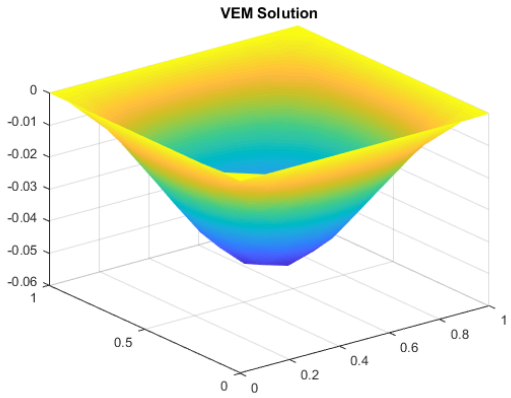
Using a higher number of points and the same hyperparameters in the ANN approach do not lead to a better performance, instead it was observed that the error went higher. This can be justified once for the presented formulation the hyperparameters must be adjusted for each different input data. Also, keeping the learning rate and the number of epochs unchanged and using a higher number of hidden layers is not effective. The problem of gradient vanishing is observed, i.e., due to the length of the neural network information is lost through it.



(a) Analytical Solution

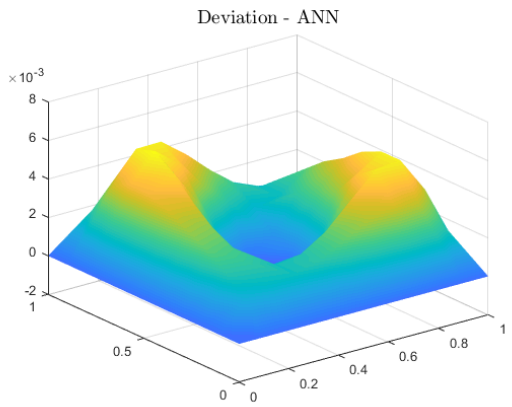


(b) ANN Solution

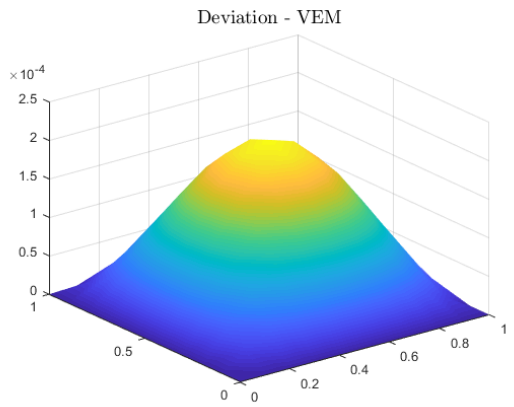


(c) VEM Solution

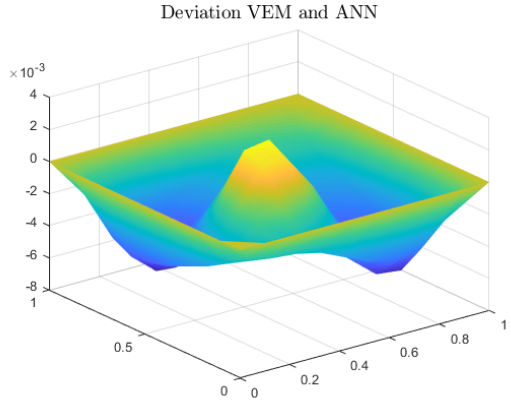
Figure 2. Analytical and numerical solution surface representation for the Poisson Equation.



(a) ANN deviation



(b) VEM deviation



(c) VEM and ANN deviation

Figure 3. Deviation between solutions.

6 Conclusions

The Virtual Element Method proposes a generalization with respect the mesh discretization. In this method any convex or non-convex polygon can be used, consequently the shape functions are not necessary polynomial. The idea is to compute those functions implicitly by projecting them in a polynomial space. The systematization of the method consists on discretizing the domain, constructing the virtual element space and choosing the degrees of freedom, introducing the projection operator, constructing the stiffness matrix upon the consistency and stability criteria and constructing the load vector.

The Artificial Neural Network approach considers a trial solution with two terms. The first term is responsible to satisfy the boundary conditions and the second term is related to the neural network. This last term is built in way not to interfere with the boundary conditions. From the solution, the loss function is defined and it is used to minimize the approximation error. The neural network model presented here was formulated to solve differential equations, more specifically the Poisson Equation.

In this project a simple ANN architecture is built and the results are compared with a implementation of the VEM. The ANN approach presented a larger error when compared to the VEM as it was shown, even though the artificial neural networks model is simpler and it is much easier to implement. This makes the neural network faster than the Virtual Element Method in terms of computational time. As for the ANNs no weak formulation is needed in the case of Poisson Equation, the regularity of the solution is preserved. On the other hand, the Virtual Element Method is much more robust in terms of solving differential equation not requiring any parameters obtained empirically. Also, the VEM can be used with a wider variety of geometrical domains.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] M. Mengolini, M. F. Benedetto, and A. M. Aragón. An engineering perspective to the virtual element method and its interplay with the standard finite element method. *Computer Methods in Applied Mechanics and Engineering*, vol. 350, pp. 995 – 1023, 2019.

[2] da L. B. Veiga, C. Lovadina, and G. Vacca. Virtual Elements for the Navier-Stokes problem on polygonal meshes. *SIAM Journal on Numerical Analysis*, vol. 56, n. 3, pp. 1210–1242, 2017a.

[3] G. H. Paulino and A. L. Gain. Bridging art and engineering using Escher-based virtual elements. *Structural and Multidisciplinary Optimization*, vol. 51, n. 4, pp. 867–883, 2015.

[4] P. F. Antonietti, M. Bruggi, S. Scacchi, and M. Verani. On the virtual element method for topology optimization on polygonal meshes: A numerical study. *Computers and Mathematics with Applications*, vol. 74, n. 5, pp. 1091–1109, 2017.

[5] P. Wriggers, W. T. Rust, and B. D. Reddy. A virtual element method for contact. *Computational Mechanics*, vol. 58, n. 6, pp. 1039–1050, 2016.

[6] F. Aldakheel, B. Hudobivnik, E. Artioli, L. Beirão da Veiga, and P. Wriggers. Curvilinear virtual elements for contact mechanics. *Computer Methods in Applied Mechanics and Engineering*, vol. 372, pp. 113394, 2020.

[7] da L. B. Veiga, F. Brezzi, A. Cangiani, G. Manzini, L. D. Marini, and A. Russo. Basic principles of virtual element methods. *Mathematical Models and Methods in Applied Sciences*, vol. 23, pp. 199–214, 2013.

[8] da L. B. Veiga, F. Brezzi, L. D. Marini, and A. Russo. The hitchhiker's guide to the virtual element method. *Mathematical Models and Methods in Applied Sciences*, vol. 2, pp. 1541–1573, 2014.

[9] O. Sutton. The virtual element method in 50 lines of matlab. *Numerical Algorithms*, vol. 75, pp. 1141–1159, 2016.

[10] A. Ortiz-Bernardin, C. Alvarez, N. Hitschfeld-Kahler, A. Russo, R. Silva-Valenzuela, and E. Olate-Sanzana. Veamy: an extensible object-oriented c++ library for the virtual element method. *Numerical Algorithms*, vol. 82, pp. 1189–1220, 2019.

[11] da L. B. Veiga, C. Lovadina, and A. Russo. Stability analysis for the virtual element method. *Mathematical Models and Methods in Applied Sciences*, vol. 27, n. 13, pp. 2557–2594, 2017b.

[12] da L. B. Veiga, F. Brezzi, L. D. Marini, and A. Russo. Virtual Element Method for general second-order elliptic problems on polygonal meshes. *Mathematical Models and Methods in Applied Sciences*, vol. 26, n. 4, pp. 729–750, 2016.

[13] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, vol. 9, n. 5, pp. 987–1000, 1998.

[14] W. Tang, T. Shan, X. Dang, M. Li, F. Yang, S. Xu, and J. Wu. Study on a poisson's equation solver based on deep learning technique. In *2017 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, pp. 1–3, 2017.

[15] J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, vol. 375, pp. 1339–1364, 2018.

[16] R. Aggarwal and H. Ugail. On the solution of poisson's equation using deep learning. *2019 13th International Conference on Software, Knowledge, Information Management and Applications, SKIMA 2019*, 2019.