POLITECNICO DI TORINO
Repository ISTITUZIONALE

The neural approximated virtual element method for elasticity problems

Original
The neural approximated virtual element method for elasticity problems / Berrone, S., Pintore, M., Teora, G.. - In: FINITE ELEMENTS IN ANALYSIS AND DESIGN. - ISSN 0168-874X. - 252:(2025), pp. 1-24. [10.1016/j.finel.2025.104467]

Availability:
This version is available at: 11583/3005353 since: 2025-11-23T08:57:16Z

Publisher:
Elsevier

Published
DOI:10.1016/j.finel.2025.104467

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

08 September 2026

Finite Elements in Analysis and Design 252 (2025) 104467

Contents lists available at ScienceDirect

Finite Elements in Analysis & Design

journal homepage: www.elsevier.com/locate/finel

The neural approximated virtual element method for elasticity problems

Stefano Berrone [a,1], Moreno Pintore [b], Gioana Teora [a,*,1]

[a] Dipartimento di Scienze Matematiche "G. L. Lagrange", Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino, 10129, Italy
[b] Laboratoire Jacques-Louis Lions, Sorbonne Université, MEGAVOLT Team, Inria, 4 place Jussieu, Paris, 75005, France

ARTICLE INFO

Keywords:
NAVEM
Elasticity
Virtual element method
Neural network
Basis functions
Polygonal meshes

ABSTRACT

We present the Neural Approximated Virtual Element Method to numerically solve elasticity problems. This hybrid technique combines classical concepts from the Finite Element Method and the Virtual Element Method with recent advances in deep neural networks. Specifically, it is a polygonal method where the virtual basis functions are element-wise approximated by a neural network, eliminating the need for stabilization or projection operators typically required in the standard Virtual Element Method. We present the discrete formulation of the problem together with theoretical results, and we provide numerical tests on both linear and non-linear elasticity problems, demonstrating the advantages of a simple discretization, particularly in handling non-linearities.

1. Introduction

Discretization methods relying on meshes comprising general polygons are receiving a great deal of attention from the scientific community. Indeed, polytopal methods can be very useful for a large variety of reasons, including the automatic handling of hanging nodes, the use of moving meshes, the ease of meshing complex geometries, and adaptivity [1,2]. Moreover, these methods have been successfully applied to several fields of computational mechanics, revealing several advantages over methods employing classical triangular/tetrahedral and quadrilateral elements. For instance, polygonal methods have shown great advantages in handling crack propagation and branching [3], contact problems [4], and topology optimization [5].

In this framework, an important polygonal method is the Virtual Element Method (VEM), introduced in [6,7] for elliptic problems and then extended to elasticity problems in [8,9]. VEM can be seen as a generalization of the standard Finite Element Method (FEM) [10,11] on more general meshes. One of the main features of the VEM, is that the local basis functions are not known in closed form. Indeed, even if they can be defined as the solution of local differential problems, they are never explicitly computed or approximated in the virtual element framework. It is therefore necessary to introduce suitable projection and stabilization operators to derive computable discrete bilinear forms associated with the Partial Differential Equation (PDE) of interest. In particular, the usage of a stabilization term is an undesirable aspect of the VEM method, as it lacks a theoretical foundation, remains problem-dependent, and an improper choice can significantly affect the method accuracy [12]. Moreover, when dealing with non-linearities, the presence of the stabilization term makes the construction of the method more involved and increases the computational effort needed to solve the problem. Finally, the need for a projector to access the point-wise evaluation of virtual functions makes the computation of post-process quantities more difficult [13].

* Corresponding author.
E-mail addresses: stefano.berrone@polito.it (S. Berrone), moreno.pintore@sorbonne-universite.fr (M. Pintore), gioana.teora@polito.it (G. Teora).
[1] The authors are members of the INdAM-GNCS.

https://doi.org/10.1016/j.finel.2025.104467
Received 8 July 2025; Received in revised form 10 September 2025; Accepted 26 September 2025
Available online 9 October 2025
0168-874X/© 2025 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Because of these reasons, in the last few years, several numerical methods have emerged to address these issues while preserving the main advantages of the method. A non-exhaustive list of such methods is the following. In [14,15], a stabilization-free formulation of the method, derived by enlarging the local projection polynomial order and enhancing the function spaces according to the polygon geometry, is presented. In [16,17], two approaches are proposed to locally approximate the local virtual basis functions and to get rid of any stabilization or projection operators: one based on the reduced basis method [18] and the other on the *Laplace solver* introduced in [19]. We observe that, in these last two methods, the basis functions are explicitly computed on each element, meaning that the "virtual" attribute only refers to the underlying discrete space.

Within the same context, the Neural Approximated Virtual Element Method (NAVEM) is introduced in [20,21] for general elliptic problems. The main idea of this novel method is to use neural networks [22] to accurately predict on the fly the local virtual basis functions, which are then used like in a standard FEM framework, to compute a discrete solution. For this reason, the NAVEM method falls within the general recent framework of Scientific Machine Learning (SciML) [23], by combining the advantages of the non-linear approximation capabilities of neural networks with those of more classical numerical methods. This framework is exponentially increasing in popularity because it represents the intersection between two active communities that rarely interacted before. We highlight that several libraries are available to easily implement and adapt complex neural networks. The most used are TensorFlow [24], PyTorch [25], and JAX [26].

In this paper, we present the NAVEM for linear and non-linear elasticity problems. Moreover, we further highlight some new key aspects of the neural network that allow us to improve the NAVEM accuracy. It is very important to notice that the neural networks used in the NAVEM framework are trained using only the geometric information of the elements. Therefore, after a potentially expensive training phase, the same neural networks can be used multiple times to solve problems characterized by different material coefficients, constitutive laws, body forces, initial configurations, but also, and most importantly, on various meshes. In this sense, it can be regarded as a method based on the so-called *offline-online* splitting.

The manuscript is organized as follows. In Section 2, we present the model problem and its VEM discretization, describing its main properties, which are essential to devise a proper neural network training strategy, and highlighting its main issues. In Section 3, we introduce the NAVEM for elasticity problems. In particular, in Section 3.1, we construct the function space from where the neural networks select the local basis functions in the NAVEM discretization, in Sections 3.2 and 3.3, we describe the neural networks architecture and training, respectively, in Section 3.4 we discuss the general structure of the NAVEM as a hybrid method mixing ideas from the machine learning and scientific computing communities and, in Section 3.5, we discuss the theoretical properties of the method. Later, in Section 4, we prove through numerical experiments the effectiveness of the method for elasticity problems, and the advantages of its simple formulation. The conclusion of the present work and the future perspectives are discussed in Section 5.

2. The model problem and the virtual element method

In the present section, we describe the problem considered in this paper and its virtual discretization. The virtual numerical scheme presented in this paper closely follows the formulation in [9] and is designed to automatically incorporate general non-linear constitutive laws, offering a flexible setting to deal with elastic problems.

2.1. The model problem

Let us consider a homogeneous isotropic elastic body $\Omega \subset \mathbb{R}^2$ with boundary $\Gamma = \partial\Omega$. We assume that this body is clamped on $\Gamma_D \subseteq \Gamma$, $|\Gamma_D| \neq 0$, and subjected to a body load $\boldsymbol{f} \in \left[L^2(\Omega)\right]^2$.

In the case of static problems, the governing equation of the solid in the initial configuration is

$$\begin{cases} -\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{x}, \nabla \boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{x}) & \text{in } \Omega, \\ \boldsymbol{u} = \boldsymbol{0} & \text{on } \Gamma_D, \\ \boldsymbol{\sigma}(\boldsymbol{x}, \nabla \boldsymbol{u})\boldsymbol{n} = \boldsymbol{0} & \text{on } \Gamma_N := \Gamma \setminus \Gamma_D, \end{cases} \tag{1}$$

where $\boldsymbol{u} : \Omega \to \mathbb{R}^2$ represents the body displacement, whereas the second-order tensor $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{x}, \nabla \boldsymbol{u})$ represents a potentially non-linear constitutive law for the material at a point $\boldsymbol{x} \in \Omega$. Moreover, $\boldsymbol{n}$ denotes the unit outward normal to the boundary Γ_N.

Let us consider the space

$$\boldsymbol{V} = \left[{H^1}_{0,\Gamma_D}(\Omega) \right]^2 := \{ \boldsymbol{v} \in \left[H^1(\Omega)\right]^2 : \ \boldsymbol{v} = \boldsymbol{0} \text{ on } \Gamma_D \}$$

and the bilinear form

$$a(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \boldsymbol{\sigma}(\boldsymbol{x}, \nabla \boldsymbol{u}(\boldsymbol{x})) : \nabla \boldsymbol{v}, \tag{2}$$

where, given two arbitrary second-order tensors $\mathbf{T}^1, \ \mathbf{T}^2 \in \mathbb{R}^{2\times 2}$, we define $\mathbf{T}^1 : \mathbf{T}^2 = \sum_{i,j=1}^2 \mathbf{T}^1_{ij} \mathbf{T}^2_{ij}$. The variational formulation of Problem (1) reads as: *Find* $\boldsymbol{u} \in \boldsymbol{V}$ *such that*

$$a(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}, \quad \forall \boldsymbol{v} \in \boldsymbol{V}. \tag{3}$$

We remark that here we are not considering internal constraints, such as incompressibility, which usually require suitable treatments.

2.2. The virtual element space

Let us consider a tessellation $\mathcal{T}_h$ of the domain Ω made up of polygonal elements E. We denote by h_E, N_E^v, $\boldsymbol{x}_E$, and $\mathcal{E}_{h,E}$ the diameter, the number of vertices/edges, the centroid and the set of edges of E, respectively. Moreover, we define $\mathcal{E}_h = \cup_{E\in\mathcal{T}_h}\mathcal{E}_{h,E}$. We further set, as usual, $h = \max_{E\in\mathcal{T}_h} h_E$. Given a generic domain $\omega \subset \mathbb{R}^d$, with $d = 1, 2$, $\mathbb{P}_1(\omega)$ denotes the space of polynomials of order less or equal to 1 defined on ω.

The virtual element space for the displacement is given by [8]:

$$\boldsymbol{V}_{h,1}(E) = \Big\{ \boldsymbol{v} \in \left[H^1(E)\right]^2 : (i)\ \Delta\boldsymbol{v} = 0 \\ (ii)\ \boldsymbol{v}_{|e} \in \left[\mathbb{P}_1(e)\right]^2,\ \forall e \in \mathcal{E}_{h,E} \text{ and } \boldsymbol{v} \in [C^0(\partial E)]^2 \Big\},$$

where $\Delta\boldsymbol{v} = \begin{bmatrix}\Delta v_1 \\ \Delta v_2\end{bmatrix}$ is the component-wise Laplacian operator, and $C^0(\partial E)$ is the space of continuous functions on the boundary ∂E of E. The degrees of freedom for a vector-valued function $\boldsymbol{v} \in \boldsymbol{V}_{h,1}(E)$ are the values of $\boldsymbol{v}$ at the vertices of the element E. Thus, the number of degrees of freedom for $\boldsymbol{V}_{h,1}(E)$ is

$$N_E^{\text{dof}} = 2N_E^v. \tag{4}$$

Let us define the set of local Lagrange basis functions $\{\boldsymbol{\varphi}_{i,E}\}_{i=1}^{N_E^{\text{dof}}}$ related to the aforementioned degrees of freedom. It is easy to check that

$$\boldsymbol{\varphi}_{i,E} = \begin{cases} \begin{bmatrix}\varphi_{i,E} \\ 0\end{bmatrix} & \text{if } i = 1, \dots, N_E^v, \\ \begin{bmatrix}0 \\ \varphi_{i-N_E^v,E}\end{bmatrix} & \text{if } i = N_E^v + 1, \dots, N_E^{\text{dof}}, \end{cases} \tag{5}$$

where $\{\varphi_{j,E}\}_{j=1}^{N_E^v}$ corresponds to the set of Lagrange basis functions related to degrees of freedom for the scalar virtual element space [6]:

$$\mathcal{V}_{h,1}(E) = \Big\{ v \in H^1(E) : (i)\ \Delta v = 0 \\ (ii)\ v_{|e} \in \mathbb{P}_1(e),\ \forall e \in \mathcal{E}_{h,E} \text{ and } v \in C^0(\partial E) \Big\}. \tag{6}$$

Indeed, it holds $\boldsymbol{V}_{h,1}(E) = [\mathcal{V}_{h,1}(E)]^2$.

By gluing together these local spaces, we obtain the global virtual discrete space $\boldsymbol{V}_{h,1}$ for the displacement. Its dimension, denoted by $N^{\text{dof}} = \dim \boldsymbol{V}_{h,1}$, corresponds to twice the number of non-Dirichlet vertices of $\mathcal{T}_h$. Finally, we denote by $\{\boldsymbol{\varphi}_i\}_{i=1}^{N^{\text{dof}}}$ the global Lagrange basis functions spanning $\boldsymbol{V}_{h,1}$.

2.3. The virtual element discretization

For each polygon E, we introduce the *computable* local polynomial projector $\Pi_1^{\nabla,E} : \left[H^1(E)\right]^2 \to \left[\mathbb{P}_1(E)\right]^2$, which is defined as

$$\begin{cases} \int_E \left(\nabla\boldsymbol{v} - \nabla\Pi_1^{\nabla,E}\boldsymbol{v}\right) : \nabla\boldsymbol{p} = 0 & \forall \boldsymbol{p} \in \left[\mathbb{P}_1(E)\right]^2, \\ \int_{\partial E} \Pi_1^{\nabla,E}\boldsymbol{v} = \int_{\partial E} \boldsymbol{v} & \forall \boldsymbol{v} \in \left[H^1(E)\right]^2. \end{cases}$$

Moreover, we denote by $\Pi_0^{0,E}$ both the vector-valued and the tensor-valued L^2-projectors onto constants. In particular, $\forall \boldsymbol{v} \in \left[H^1(E)\right]^2$, it holds [27]

$$\Pi_0^{0,E}\nabla\boldsymbol{v} = \nabla\Pi_1^{\nabla,E}\boldsymbol{v},$$

and

$$\Pi_0^{0,E}\text{div}\boldsymbol{v} = \Pi_0^{0,E}\text{tr}(\nabla\boldsymbol{v}) = \text{tr}(\Pi_0^{0,E}\nabla\boldsymbol{v}).$$

Here, $\text{div}\boldsymbol{v} = \nabla\cdot\boldsymbol{v}$ denotes the divergence of a vector-valued function $\boldsymbol{v}$, whereas $\text{tr}(\mathbf{T}) = \sum_{i=1}^2 \mathbf{T}_{ii}$ represents the trace of the second-order tensor $\mathbf{T} \in \mathbb{R}^{2\times2}$.

Specifically, by computable we mean that the projection of virtual functions can be computed only through the information provided by the degrees of freedom of such a function. Now, let us split the continuous bilinear form (2) according to $\mathcal{T}_h$, i.e.

$$a(\boldsymbol{u},\boldsymbol{v}) = \sum_{E\in\mathcal{T}_h} a^E(\boldsymbol{u},\boldsymbol{v}), \quad \text{with} \quad a^E(\boldsymbol{u},\boldsymbol{v}) = \int_E \boldsymbol{\sigma}(\boldsymbol{x},\nabla\boldsymbol{u}) : \nabla\boldsymbol{v}. \tag{7}$$

Given $\boldsymbol{w}_h \in \boldsymbol{V}_{h,1}$, for all $E \in \mathcal{T}_h$, and for all $\boldsymbol{u}_h,\ \boldsymbol{v}_h \in \boldsymbol{V}_{h,1}$, we define the discrete virtual bilinear form as [9]:

$$\begin{aligned} a_h^E(\boldsymbol{u}_h, \boldsymbol{v}_h; \boldsymbol{w}_h) := & \int_E \boldsymbol{\sigma}(\boldsymbol{x}, \Pi_0^{0,E}\nabla \boldsymbol{u}_h) : \Pi_0^{0,E}\nabla \boldsymbol{v}_h \\ & + \alpha_E(\boldsymbol{w}_h) S^E((I - \Pi_1^{\nabla,E})\boldsymbol{u}_h, (I - \Pi_1^{\nabla,E})\boldsymbol{v}_h). \end{aligned} \tag{8}$$

where I is the identity operator, the first term in (8) is usually called *consistency term*, and the *stability term* $S^E(\cdot,\cdot)$ is chosen in such a way it scales like $a^E(\cdot,\cdot)$ for a unitary material constants on the kernel of $\Pi_1^{\nabla,E}$ [6]. On the other hand, the *stabilizing parameter* $\alpha_E(\boldsymbol{w}_h)$ takes into account different material constants as well as non-linear materials for a given configuration $\boldsymbol{w}_h \in \boldsymbol{V}_{h,1}$. The discrete virtual element problem reads as: *Given* $\boldsymbol{w}_h \in \boldsymbol{V}_{h,1}$, *find* $\boldsymbol{u}_h \in \boldsymbol{V}_{h,1}$ *such that*

$$\sum_{E\in\mathcal{T}_h} a_h^E(\boldsymbol{u}_h, \boldsymbol{v}_h; \boldsymbol{w}_h) = \sum_{E\in\mathcal{T}_h} \int_E \boldsymbol{f} \cdot \Pi_0^{0,E}\boldsymbol{v}_h \quad \forall \boldsymbol{v}_h \in \boldsymbol{V}_{h,1}. \tag{9}$$

Finally, to solve Problem (9), we adopt the Newton–Raphson method to address the issue of non-linearity. Given $\boldsymbol{w}_h, \boldsymbol{u}_h \in \boldsymbol{V}_{h,1}$, let us introduce the following local bilinear form, for all $\boldsymbol{\delta}_h,\ \boldsymbol{v}_h \in \boldsymbol{V}_{h,1}$,

$$\begin{aligned} b_h^E(\boldsymbol{\delta}_h, \boldsymbol{v}_h; \boldsymbol{u}_h, \boldsymbol{w}_h) = & \int_E \left(\mathbb{A}(\boldsymbol{x}, \Pi_0^{0,E}\nabla \boldsymbol{u}_h) : \nabla \Pi_0^{0,E}\boldsymbol{\delta}_h\right) : \Pi_0^{0,E}\nabla \boldsymbol{v}_h \\ & + \alpha_E(\boldsymbol{w}_h) S^E((I - \Pi_1^{\nabla,E})\boldsymbol{\delta}_h, (I - \Pi_1^{\nabla,E})\boldsymbol{v}_h), \end{aligned} \tag{10}$$

where the elastic modulus $\mathbb{A}(\boldsymbol{x}, \nabla \boldsymbol{u})$ is defined as the fourth-order tensor [9]:

$$\mathbb{A}(\boldsymbol{x}, \nabla \boldsymbol{u}) = \frac{\partial \boldsymbol{\sigma}(\boldsymbol{x}, \nabla \boldsymbol{u})}{\partial \nabla \boldsymbol{u}}. \tag{11}$$

Given $\boldsymbol{u}_h,\ \boldsymbol{w}_h \in \boldsymbol{V}_{h,1}$, whose associated degrees of freedom are gathered in the vectors $\boldsymbol{u}_V,\ \boldsymbol{w}_V \in \mathbb{R}^{N^{\text{dof}}}$, for all $i, j = 1, \dots, N^{\text{dof}}$, we define the following virtual vectors and matrices:

$$\begin{aligned} \mathbf{A}_V(\boldsymbol{u}_V; \boldsymbol{w}_V) \in \mathbb{R}^{N^{\text{dof}}} \ : \quad & [\mathbf{A}_V(\boldsymbol{u}_V; \boldsymbol{w}_V)]_i = \sum_{E\in\mathcal{T}_h} a_h^E(\boldsymbol{u}_h, \boldsymbol{\varphi}_i; \boldsymbol{w}_h), \\ \boldsymbol{f}_V \in \mathbb{R}^{N^{\text{dof}}} \ : \quad & [\boldsymbol{f}_V]_i = \sum_{E\in\mathcal{T}_h} \int_E \boldsymbol{f} \cdot \Pi_0^{0,E}\boldsymbol{\varphi}_i, \\ \mathbf{K}_V(\boldsymbol{u}_V; \boldsymbol{w}_V) \in \mathbb{R}^{N^{\text{dof}}\times N^{\text{dof}}} \ : \quad & [\mathbf{K}_V(\boldsymbol{u}_V; \boldsymbol{w}_V)]_{ji} = \sum_{E\in\mathcal{T}_h} b_h^E(\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j; \boldsymbol{u}_h, \boldsymbol{w}_h). \end{aligned}$$

The discrete virtual problem in matrix form reads as: *Find* $\boldsymbol{u}_V \in \mathbb{R}^{N^{\text{dof}}}$ *such that*

$$\boldsymbol{r}_V(\boldsymbol{u}_V; \boldsymbol{w}_V) = \mathbf{A}_V(\boldsymbol{u}_V; \boldsymbol{w}_V) - \boldsymbol{f}_V = 0. \tag{12}$$

Given $\boldsymbol{w}_V \in \mathbb{R}^{N^{\text{dof}}}$, by starting from an initial guess $\boldsymbol{u}_V^0 \in \mathbb{R}^{N^{\text{dof}}}$ for the displacement, as in [28], the update rule for the Newton–Raphson method related to the virtual element problem (12) reads as:

$$\begin{aligned} \mathbf{K}_V(\boldsymbol{u}_V^k; \boldsymbol{w}_V)\boldsymbol{\delta}_V^k &= -\boldsymbol{r}_V(\boldsymbol{u}_V^k; \boldsymbol{w}_V) \\ \boldsymbol{u}_V^{k+1} &= \boldsymbol{u}_V^k + \boldsymbol{\delta}_V^k \end{aligned} \qquad \forall k \geq 0. \tag{13}$$

Among the commonly used numerical methods for solving systems of non-linear equations, the Newton–Raphson method is the fastest, offering a quadratic convergence provided that the initial estimate is sufficiently close to the solution. Therefore, selecting an initial estimate $\boldsymbol{u}_V^0$ close to the solution is crucial for accelerating the convergence. In solid mechanics, the initial estimate is usually set to the undeformed shape of the structure, i.e. the initial displacement is set to the all-zeros vector. Since the initial estimate usually starts from the zero displacement, the Newton–Raphson method converges quickly for small deformations. However, convergence becomes challenging when large displacements are considered or geometric instabilities arise. In such cases, it is advisable to consider the external load $\boldsymbol{f}_V$ as applied in a series of N increments $\lambda_n \boldsymbol{f}_V$, with $n = 1, \dots, N$ such that $\lambda_N = 1$. Clearly, the more increments taken, the easier it becomes to find a converged solution for each load step [29]. This approach requires solving the following problems iteratively

$$\bar{\boldsymbol{r}}_V(\boldsymbol{u}_V^n, \lambda_V^n; \boldsymbol{w}_V) = \mathbf{A}_V(\boldsymbol{u}_V^n; \boldsymbol{w}_V) - \lambda_V^n \boldsymbol{f}_V = 0 \quad \forall n = 1, \dots, N. \tag{14}$$

Different methods are available in the literature to choose the value for the scalar load parameter λ_V^n. Examples include, but are not limited to, the incremental force method [30] and the arc-length method [31]. The incremental force method, for instance, applies the load in constant increments, i.e. $\lambda_V^n = \frac{n}{N}$. Within each increment, the Newton–Raphson procedure to determine the corresponding displacement $\boldsymbol{u}_V^n$ remains unchanged. A new load increment is applied after the solution corresponding to the previous load increment has successfully converged [30]. The obtained solution becomes the initial guess for the Newton–Raphson algorithm at the next loading step. However, this method may fail in the presence of limit points in the load–displacement response [29]. These issues may be bypassed using the arc-length method. In the arc-length method, the load parameter λ_V^n is treated as the $(N^{\text{dof}} + 1)$th unknown, and an additional equation is added to the system. In this approach, both the displacement and the load parameter are expressed as functions of the curvilinear coordinate system along the load–displacement response of the structure. At

each loading step, a variation of the increment of the curvilinear coordinate is performed, and the corresponding non-linear system in the unknown $(\boldsymbol{u}_V^n, \lambda_V^n)$ is solved using the Newton–Raphson algorithm.

Usually, the function $\boldsymbol{w}_h$ in Eq. (8) is set equal to the solution found at the last loading increment step, i.e. $\boldsymbol{u}_h^{n-1}$. This choice is taken to simplify the Newton–Raphson application by avoiding the computation of derivatives of α_E in the related tangent matrix (10). Indeed, although $\boldsymbol{w}_h = \boldsymbol{u}_h^n$ is a more intuitive choice, it is also more computationally demanding [9,13]. Other possible choices include setting $\boldsymbol{w}_h = \boldsymbol{0}$, i.e. evaluating α_E in the undeformed configuration. However, this might lead to unsatisfactory results, also in the small deformations regime [13]. Moreover, different choices for α_E have been introduced:

- *norm-based stabilization* [9]:

$$\alpha_E(\boldsymbol{w}_h) = \left|\!\left|\!\left| \mathbb{A}(\boldsymbol{x}_E, \Pi_0^{0,E} \nabla \boldsymbol{w}_h) \right|\!\right|\!\right|,$$

  which is chosen to produce a strictly positive stabilization term. The operator $|||\cdot|||$ could be chosen as any fourth-order tensor norm, whereas $\boldsymbol{x}_E$ represents the centroid of E.
- *trace-based stabilization* [13]:

$$\alpha_E(\boldsymbol{w}_h) = \frac{1}{4} \mathrm{tr} \left(\mathbb{A}(\boldsymbol{x}_E, \Pi_0^{0,E} \nabla \boldsymbol{w}_h) \right).$$

  We observe that, with this choice, the stabilization may assume negative values.
- *stiffness-based stabilization* [32]:

$$\alpha_E(\boldsymbol{w}_h) = \frac{1}{4N^{\mathrm{dof}}} \left(\sum_{i=1}^{N^{\mathrm{dof}}} \left((\mathbf{K}_V^C)_{ii} \right)^2 \right)^{\frac{1}{2}} \quad \text{or} \quad \alpha_E(\boldsymbol{w}_h) = \frac{1}{4N^{\mathrm{dof}}} \sum_{i=1}^{N^{\mathrm{dof}}} (\mathbf{K}_V^C)_{ii},$$

  where $\mathbf{K}_V^C$ is the consistency part of the global tangent system matrix.

3. The neural approximated virtual element method

In the NAVEM method, we propose to approximate local virtual scalar basis functions $\varphi_{j,E} \in \mathcal{V}_{h,1}(E)$ by means of a neural network-based approach and to assemble the final linear system as in the more classical Finite Element Method [21].

For this purpose, we approximate $\varphi_{j,E}$ by a function $\varphi_{j,E}^{\mathcal{NN}}$ in a suitable functional space $\mathcal{H}_{j,E}^{\mathcal{NN}}$. A similar procedure is repeated to approximate the gradient of $\varphi_{j,E}$ (or, more generally, all the required derivatives). Finally, since the approximation $\varphi_{j,E}^{\mathcal{NN}}$ of $\varphi_{j,E}$ is known in closed form as well as its derivatives, it can be evaluated point-wise inside each element $E \in \mathcal{T}_h$ to numerically compute the integrals in (3), without introducing any projection or stability operator.

3.1. The local approximation space

In this section, we describe the local function space $\mathcal{H}_{j,E}^{\mathcal{NN}}$ in which the approximation $\varphi_{j,E}^{\mathcal{NN}}$ of $\varphi_{j,E}$ is sought. Such space coincides with the one adopted in [21]. Here, for the sake of clarity, we briefly recall the main concepts and refer to [21] for a more detailed description.

All the functions in $\mathcal{H}_{j,E}^{\mathcal{NN}}$ are defined on a reference squared region $S^{\mathcal{NN}} = [-R^{\mathcal{NN}}, R^{\mathcal{NN}}]^2 \subset \mathbb{R}^2$ independent of E, where $R^{\mathcal{NN}}$ is a user-defined positive scalar constant that scales like $\mathcal{O}(1)$. Therefore, we assume that a suitable affine map is available to map each element $E \in \mathcal{T}_h$ to a reference element $\hat{E} \subset S^{\mathcal{NN}}$. A convenient map is the one described in [33], but other choices are admissible.

Let $\mathbb{H}_{\ell^{\mathcal{NN}}}(S^{\mathcal{NN}})$ be the space of harmonic polynomials up to order $\ell^{\mathcal{NN}} \geq 0$. A basis of such a space is the set of scaled polynomials

$$\left\{ 1,\ \Re\left(\left(\frac{z}{R^{\mathcal{NN}}} \right)^{\ell} \right), \Im\left(\left(\frac{z}{R^{\mathcal{NN}}} \right)^{\ell} \right),\ \ell = 1, \dots, \ell^{\mathcal{NN}} \right\}. \tag{15}$$

Here z is the complex number $z = x_1 + ix_2$ associated with the point $\boldsymbol{x} = [x_1, x_2]^T \in \mathbb{R}^2$, and $\Re$ and $\Im$ denote the real and imaginary part of a complex quantity. Note that all the polynomials in (15) can be recursively retrieved as in [34].

As discussed in [21], harmonic polynomials are not enough to accurately approximate the VEM basis functions, we thus introduce additional functions mimicking the target virtual functions. Let us introduce the Laplace problem

$$\begin{cases} \Delta\tilde{\Phi} = 0 & \text{in } \Omega_{\Phi} = (-1,1)^2, \\ \tilde{\Phi} = 1 + x_2 & \text{on } \Gamma_{\Phi,1} = \{x_1 = 1 \text{ and } -1 \leq x_2 \leq 0\}, \\ \tilde{\Phi} = 1 - x_2 & \text{on } \Gamma_{\Phi,2} = \{x_1 = 1 \text{ and } 0 \leq x_2 \leq 1\}, \\ \tilde{\Phi} = 0 & \text{on } \partial\Omega_{\Phi} \setminus \{\Gamma_{\Phi,1} \cup \Gamma_{\Phi,2}\}, \end{cases}$$

and approximate its solution with a function Φ of the form

$$\Phi(z) = \sum_{\alpha=1}^{N^1} c_{\alpha}^1 \Re\left(\frac{d_{\alpha}}{z - z_{\alpha}} \right) + \sum_{\beta=0}^{N^2} c_{\beta}^2 \Re\left(\left(\frac{z}{2} \right)^{\beta} \right). \tag{16}$$

In this formula, $\Re\left(\left(\frac{z}{2}\right)^{\beta}\right)$, $\beta = 0, \dots, N^2$, are known harmonic polynomials and $\Re\left(\frac{d_\alpha}{z-z_\alpha}\right)$ are known harmonic rational functions associated with known poles $z_\alpha = 1 + 2\exp\left(-4(\sqrt{N_1} - \sqrt{\alpha})\right)$, for $\alpha = 1, \dots, N^1$. The sought coefficients c^1_α, $\alpha = 1, \dots, N^1$, and c^1_β, $\beta = 1, \dots, N^2$, are computed by solving a linear least square problem to minimize the difference between $\boldsymbol{\Phi}$ and $\tilde{\boldsymbol{\Phi}}$ on a given set of points on $\partial\Omega_{\boldsymbol{\Phi}}$. The function $\boldsymbol{\Phi}$ is then used to compute three auxiliary functions $\boldsymbol{\Phi}^{j-1}_{j,E}$, $\boldsymbol{\Phi}^{j}_{j,E}$, and $\boldsymbol{\Phi}^{j+1}_{j,E}$ through three different affine maps. Such maps map the point $(1,0)$ to the $(j-1)$, j, or $(j+1)$th vertex of E, and ensure that E is included in the image of $\Omega_{\boldsymbol{\Phi}}$. We note that, since $\boldsymbol{\Phi}$ does not depend on E or $\widehat{E}$, it is computed only once.

Finally, for any element $E \in \mathcal{T}_h$ and each vertex v_j of E, with $= 1, \dots, N^v_E$, we can define the space $\mathcal{H}^{\mathcal{N}\mathcal{N}}_{j,E}$, as:

$$\mathcal{H}^{\mathcal{N}\mathcal{N}}_{j,E} = \text{span}\left\{ \{\tilde{p}_\beta\}_{\beta=1}^{2\ell^{\mathcal{N}\mathcal{N}}+1},\ \boldsymbol{\Phi}^{j-1}_{j,E}, \boldsymbol{\Phi}^{j}_{j,E}, \boldsymbol{\Phi}^{j+1}_{j,E} \right\}, \tag{17}$$

where $\{\tilde{p}_\beta\}_{\beta=1}^{2\ell^{\mathcal{N}\mathcal{N}}+1}$ are the harmonic polynomials in (15). Therefore, the dimension of $\mathcal{H}^{\mathcal{N}\mathcal{N}}_{j,E}$ is

$$\dim \mathcal{H}^{\mathcal{N}\mathcal{N}}_{j,E} = 2\ell^{\mathcal{N}\mathcal{N}} + 4. \tag{18}$$

We observe that this dimension does not depend on the number of functions $N^1 + N^2$ used to approximate $\boldsymbol{\Phi}$.

3.2. The neural network architecture

For ease of notation, let us define

$$\mathcal{H}^{\mathcal{N}\mathcal{N}}_{j,E} = \text{span}\{h_k\}_{k=1}^{\dim \mathcal{H}^{\mathcal{N}\mathcal{N}}_{j,E}}, \tag{19}$$

where $\{h_k\}_{k=1}^{\dim \mathcal{H}^{\mathcal{N}\mathcal{N}}_{j,E}}$ coincides with the set of basis functions in (17). Therefore, given a pair (v_j, E), our goal is to find a set of coefficients $\{c^{\varphi}_k(v_j, E)\}_{k=1}^{\dim \mathcal{H}^{\mathcal{N}\mathcal{N}}_{j,E}}$ such that the *NAVEM function* $\varphi^{\mathcal{N}\mathcal{N}}_{j,E}$, expressed as

$$\varphi^{\mathcal{N}\mathcal{N}}_{j,E} = \sum_{k=1}^{\dim \mathcal{H}^{\mathcal{N}\mathcal{N}}_{j,E}} c^{\varphi}_k(v_j, E) h_k, \tag{20}$$

minimizes the distance from the local virtual element function $\varphi_{j,E}$ on the boundary of E where $\varphi_{j,E}$ is well-known. In this context, the involved neural network is a parametric function mapping the pair (v_j, E) to the vector $[c^{\varphi}_k(v_j, E)]_{k=1}^{\dim \mathcal{H}^{\mathcal{N}\mathcal{N}}_{j,E}}$. We remark that, since a neural network accepts only vectors of a given dimension as inputs, the pair (v_j, E) has to be suitably encoded into a vector $\boldsymbol{x}_0$. Multiple input encoding strategies are admissible. In this work, we adopt the approach described in [21], where an input encoding strategy is detailed, along with variability and input reduction strategies for generic polygons. To perform such an encoding, we first map the polygon E to a polygon $\overline{E}$ centered in the origin, with approximately unitary diameter, and such that the vertex v_j is mapped to $(1,0)$. Let $(\overline{x}^i_1, \overline{x}^i_2)$ be the mappings of the generic vertex v_i, the corresponding encoding vector is

$$\boldsymbol{x}_0 = [\overline{x}^{j+1}_1, \overline{x}^{j+1}_2,\ \overline{x}^{j+2}_1, \overline{x}^{j+2}_2,\ \dots,\ \overline{x}^{j+N^v-1}_1, \overline{x}^{j+N^v-1}_2], \tag{21}$$

where the superscripts are intended up to modulo N^v_E. We note that since this encoding strategy produces input with dimension $2(N^v_E - 1)$ that depends on the number of vertices N^v_E, a neural network must be trained for each class of polygons characterized by a different number of vertices to fix the input dimension.

Given the encoding $\boldsymbol{x}_0$ of the pair (v_j, E), the set of coefficients $\mathbf{c}^{\varphi}(\boldsymbol{x}_0) := \{c^{\varphi}_k(v_j, E)\}_{k=1}^{\dim \mathcal{H}^{\mathcal{N}\mathcal{N}}_{j,E}}$ is expressed as the output of the following fully-connected feed-forward neural network:

$$\begin{aligned} &\boldsymbol{x}^{\varphi}_0 := \boldsymbol{x}_0, \\ &\boldsymbol{x}^{\varphi}_\ell = \rho(A^{\varphi}_\ell \boldsymbol{x}^{\varphi}_{\ell-1} + b^{\varphi}_\ell), \qquad \ell = 1, \dots, L-1, \\ &\mathbf{c}^{\varphi}(\boldsymbol{x}_0) = A^{\varphi}_L \boldsymbol{x}^{\varphi}_{L-1} + b^{\varphi}_L. \end{aligned} \tag{22}$$

Eq. (22) describes the architecture of the employed neural network [35], that can be easily extended to admit multiple inputs at once. Note that more complex architectures exist and can be easily implemented thanks to libraries like TensorFlow [24], PyTorch [25], or JAX [26]. In this context, however, the input and output vectors have a fixed and known size and do not present a grid structure that may be exploited. For this reason, a simple fully-connected feed-forward neural network, also named Multi-Layer Perceptron (or MLP) is the optimal choice. Its main elements and hyperparameters are:

- the number of layers L: L is a strictly positive integer representing the number of affine transformations performed in (22). In the NAVEM framework, we suggest considering values of L between 2 and 10, in order to have good approximation capabilities and a small computational cost for each evaluation of the neural network.
- The non-linear activation function $\rho : \mathbb{R} \to \mathbb{R}$: ρ is a generic non-linear function that is applied element-wise to its input vector, i.e. given a vector $\mathbf{y} \in \mathbb{R}^m$, we define $\rho(\mathbf{y}) \in \mathbb{R}^m$ as the vector such that $[\rho(\mathbf{y})]_i = \rho(\mathbf{y}_i)$, for all $i = 1, \dots, m$. Several alternatives exist, some of the most commonly used ones are the Rectified Linear Unit (ReLU) $\rho(x) = \max(0, x)$, the logistic sigmoid function

$\rho(x) = 1/(1+e^{-x})$, and the hyperbolic tangent $\rho(x) = \tanh(x)$. In this work we only use the hyperbolic tangent, but we highlight that any activation function can be used because this choice does not affect the regularity of the NAVEM basis functions. In fact, the spatial coordinates in (20) appear only in the functions h_k, and the regularity of the activation function influences only the regularity of the coefficients c_k^{φ} with respect to the coordinates of the vertices (which are fixed when one considers a single element to construct the corresponding basis functions).

- The matrices and vectors $A_\ell^{\varphi} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and $b_\ell^{\varphi} \in \mathbb{R}^{N_\ell}$ for $\ell = 1, \dots, L$: these are the matrices and vectors containing the trainable coefficients of the neural network, which are optimized by minimizing a suitable loss function, discussed in the following. As for the number of layers L, each value N_ℓ, $\ell = 1, \dots, L-1$, is a hyperparameter that has to be chosen by the user, whereas the input dimension must be $N_0 = 2(N_E^v - 1)$ to accept inputs of size $2(N_E^v - 1)$, and the output dimension must be $N_L = \dim \mathcal{H}_{j,E}^{\mathcal{N},\mathcal{N}}$ to provide output of size compatible with (20). Once more, to reach a good compromise between approximation capabilities and computational efficiency, we suggest considering integer values between 10 and 200 for each N_ℓ related to intermediate layers. The initialization of all these matrices and vectors is performed randomly, with several strategies available in the literature. In this work, we consider the Glorot Normal initialization [36], which draws samples from a normal distribution with zero-mean and a standard deviation determined by the size of the matrices and vectors.

Let us now focus on the training of the introduced neural network. To optimize the trainable coefficients, we propose to minimize the distance between the traces of the functions $\varphi_{j,E}^{\mathcal{N},\mathcal{N}}$ and $\varphi_{j,E}$ on ∂E, i.e. we want to minimize the following quantity

$$\epsilon_{j,E}^{\varphi} = \|\varphi_{j,E}^{\mathcal{N},\mathcal{N}} - \varphi_{j,E}\|_{H^{1/2}(\partial E)}, \tag{23}$$

for each pair (v_j, E) in a given training dataset. We remark that this quantity is computable because it involves only the evaluations of $\varphi_{j,E}$ on the boundary of E, where this function is known in closed form.

Let $\boldsymbol{q}_{j,E} := \nabla \varphi_{j,E}$ be the gradient of $\varphi_{j,E}$. We observe that this gradient can be approximated by the gradient of the function $\varphi_{j,E}^{\mathcal{N},\mathcal{N}}$ defined in (20) [20], which can be exactly computed as:

$$\nabla \varphi_{j,E}^{\mathcal{N},\mathcal{N}} = \sum_{k=1}^{\dim \mathcal{H}_{j,E}^{\mathcal{N},\mathcal{N}}} c_k^{\varphi}(v_j, E) \nabla h_k. \tag{24}$$

However, such a strategy is observed to be suboptimal. For this reason, it is more convenient to introduce a second fully-connected feed-forward neural network to approximate $\boldsymbol{q}_{j,E}$ with vector-functions contained in $\nabla \mathcal{H}_{j,E}^{\mathcal{N},\mathcal{N}}$. This neural network, similarly to (22), can be expressed as:

$$\begin{aligned} &\boldsymbol{x}_0^{\boldsymbol{q}} := \boldsymbol{x}_0, \\ &\boldsymbol{x}_\ell^{\boldsymbol{q}} = \rho(A_\ell^{\boldsymbol{q}} \boldsymbol{x}_{\ell-1}^{\boldsymbol{q}} + b_\ell^{\boldsymbol{q}}), \qquad \ell = 1, \dots, L-1, \\ &\mathbf{c}^{\boldsymbol{q}}(\boldsymbol{x}_0) = A_L^{\boldsymbol{q}} \boldsymbol{x}_{L-1}^{\boldsymbol{q}} + b_L^{\boldsymbol{q}}. \end{aligned} \tag{25}$$

In this manuscript, we consider the same architecture and hyperparameters for the neural networks in (22) and in (25). However, we remark that this is not a restrictive constraint and can be relaxed without any implication on the nature or on the analysis of the method.

The output $\mathbf{c}^{\boldsymbol{q}}(\boldsymbol{x}_0) := [c_k^{\boldsymbol{q}}]_{k=1}^{\dim \mathcal{H}_{j,E}^{\mathcal{N},\mathcal{N}}}$ of (25) represents the set of coefficients that allows us to approximate $\boldsymbol{q}_{j,E}$ through the following vector

$$\boldsymbol{q}_{j,E}^{\mathcal{N},\mathcal{N}} = \sum_{k=1}^{\dim \mathcal{H}_{j,E}^{\mathcal{N},\mathcal{N}}} c_k^{\boldsymbol{q}}(v_j, E) \nabla h_k. \tag{26}$$

The trainable coefficients $A_\ell^{\boldsymbol{q}}$ and $b_\ell^{\boldsymbol{q}}$ are optimized by minimizing a *computable* loss function. This cost term is defined through the following quantity

$$\epsilon_{j,E}^{\boldsymbol{q}} = \| \left(\boldsymbol{q}_{j,E}^{\mathcal{N},\mathcal{N}} - \boldsymbol{q}_{j,E} \right) \cdot \boldsymbol{t} \|_{L^2(\partial E)}, \tag{27}$$

where $\boldsymbol{t}$ denotes the tangential vector to the boundary ∂E.

Remark 1. In [21], the authors proved that small values of $\epsilon_{j,E}^{\varphi}$ and $\epsilon_{j,E}^{\boldsymbol{q}}$ are enough to ensure a good approximation of $\varphi_{j,E}$ and of $\boldsymbol{q}_{j,E}$ on the entire element E, even if they are minimized only on ∂E. We observe that such an approach introduces a consistency error because (26) does not exactly coincide with the derivative of (20), but it significantly reduces the oscillations in the gradients of the NAVEM basis functions, leading to a more stable method. For further details, see Section 3.5.

3.3. The training phase

We highlight that employing the input encoding strategy presented in [21] requires defining a different network for each class of polygons characterized by a different number of vertices $N^v \geq 4$. For $N^v = 3$, the virtual element basis functions, as well as the NAVEM basis functions, coincide with the ones of the standard FEM.

Table 1
Final training loss functions computed using the neural networks in [21] and the ones in the current manuscript.

N^v	$\mathcal{L}_\varphi$ [21]	$\mathcal{L}_\varphi$	$\mathcal{L}_q$ [21]	$\mathcal{L}_q$
4	4.62e−03	5.13e−04	1.00e−02	2.94e−03
5	1.97e−03	2.81e−04	4.97e−03	1.84e−03
6	1.02e−03	1.12e−04	2.96e−03	1.07e−03
7	2.15e−03	3.40e−04	5.22e−03	2.07e−03
8	1.21e−03	3.26e−04	4.88e−03	2.47e−03

For each class of polygons, let $\mathcal{T}^{\text{train}}$ be a training dataset of size $\#\mathcal{T}^{\text{train}}$ containing polygons E with N^v vertices. Then, to train the neural networks (22) and (25), i.e. to approximate the basis functions and their gradients, we minimize the following loss functions

$$\mathcal{L}_\varphi = \sqrt{\frac{1}{\#\mathcal{T}^{\text{train}} N^v} \sum_{E\in\mathcal{T}^{\text{train}}} \sum_{j=1}^{N^v} \left(\epsilon_{j,E}^{\varphi}\right)^2},$$
$$\mathcal{L}_q = \sqrt{\frac{1}{\#\mathcal{T}^{\text{train}} N^v} \sum_{E\in\mathcal{T}^{\text{train}}} \sum_{j=1}^{N^v} \left(\epsilon_{j,E}^{q}\right)^2}, \tag{28}$$

respectively. In this expression, for computational purposes, the loss function (23) is approximated by $\|\varphi_{j,E}^{\mathcal{NN}} - \varphi_{j,E}\|_{L^2(\partial E)}$. Finally, a standard regularization term penalizing the L^2-norm of the trainable weights, with regularization coefficients 10^{-8}, is added to both losses. We observe that these formulas differ from the ones defined in [21]. Indeed, in [21], the used loss functions coincide with the squared losses $\mathcal{L}_\varphi^2$ and $\mathcal{L}_q^2$. Note that the global minima of these two pairs of losses are the same, but we empirically observed better performance minimizing (28). Moreover, we here decide to employ a new optimizer: after performing 5000 epochs with the first-order ADAM optimizer [37] as in [21], we carry out 5000 updates with the self-scaled BFGS optimizer described in [38,39]. This last advanced optimization technique enhances convergence by adaptively scaling the weights updates provided by the BFGS optimizer.

To appreciate the effectiveness of these new techniques, we compare the final values of the training losses by training the neural networks as described here and in [21]. More precisely, we here build new neural networks characterized by the same architecture used in [21], i.e. they all have 5 layers, with 50 neurons in each layer, and the *tanh* activation function in each hidden layer. For the network associated with the class of polygons with N^v vertices, the input dimension is $N_0 = 2(N^v - 1)$, while the output is always $N_L = 2\ell^{\mathcal{NN}} + 4$, as discussed in Section 3.1. We fix $\ell^{\mathcal{NN}} = 20$, independently of the associated class of polygons, and set $R^{\mathcal{NN}} = 3$. A fine-tuning strategy of all these values should be advisable, but it is beyond the scope of the current paper. Moreover, in order to train such networks, we use the same datasets RDQM, for $N^v = 4$, and VM, for $N^v \geq 5$, presented in [21]: the former contains randomly generated quadrilaterals, and the latter contains polygons sampled from Voronoi meshes generated through [40]. The entire code is written in Python, and the implementation of the definition, optimization, and evaluation of the neural networks heavily relies on the TensorFlow library [24].

The differences between the accuracy of the two approaches can be appreciated in Table 1. In particular, we report the value of $\mathcal{L}_\varphi$ and $\mathcal{L}_q$ for the networks used in [21] and for the networks used in the current manuscript. It is evident that the change of the loss and of the optimizer allows one to more accurately approximate both the basis functions and their gradients. We remark that these changes do not influence the overall computational cost of the training phase because the cost of the additional operations is negligible.

We would like to remark that the comparison is meaningful because we are aiming to approximate the same scalar virtual element space $\mathcal{V}_{h,1}(E)$.

3.4. The online phase and the NAVEM discretization

In this section, assuming that suitable trained neural networks are available to approximate the basis functions related to $\mathcal{T}_h$, we present the NAVEM discretization of the elastic problem (1). The overall NAVEM procedure can be divided into two main parts. The first part (steps **S.1–S.3**) is specific to the NAVEM approach. We observe that steps **S.1–S.3** are described for the case of a single input $\boldsymbol{x}_0$ at a time, but, for computational efficiency, multiple inputs can also be processed simultaneously. This latter case is illustrated in Fig. 1. The second part of the NAVEM algorithm (steps **S.4–S.5**) depends only on the problem and can be handled as in the standard Finite Element Method.

Let us describe the first part. For each element E in the tessellation $\mathcal{T}_h$ of the domain Ω, and for each vertex v_j of E, with $j = 1, \dots, N_E^v$, we

S.1 *classify* the pair (v_j, E) based on the number of vertices of E, as discussed in Section 3.3, to select the neural networks that have to be used to produce $\{\varphi_{j,E}^{\mathcal{NN}},\ \boldsymbol{q}_{j,E}^{\mathcal{NN}}\}$ to approximate the corresponding virtual basis function $\varphi_{j,E}$ and its gradient.

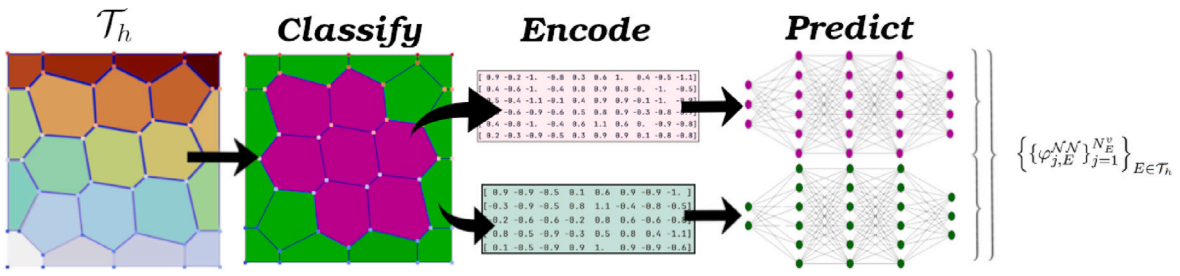


Fig. 1. A sketch of the NAVEM online phase for handling multiple inputs (steps S.1–S.3). Each elements of $\mathcal{T}_h$ is classified according to its number of vertices. In the figure, pentagons are colored in green and belong to the "green" class $\mathcal{C}^g$, whereas hexagons are shown in pink and belong to the "pink" class $\mathcal{C}^p$. Inputs from the same class are collected into input matrices of size $\#\mathcal{C}^g \times 8$ for pentagons and $\#\mathcal{C}^p \times 10$ for hexagons. Finally, these inputs are used to predict the associated basis functions.

S.2 *encode* the information (v_j, E) into the vector $\boldsymbol{x}_0$ that represents the input vector for the neural networks using (21) or alternative encoding strategies;

S.3 *predict* the coefficients $\mathbf{c}^{\varphi}(\boldsymbol{x}_0)$ and $\mathbf{c}^{\boldsymbol{q}}(\boldsymbol{x}_0)$ to compute $\varphi_{j,E}^{\mathcal{NN}}$ and $\boldsymbol{q}_{j,E}^{\mathcal{NN}}$ according to (20) and (26), respectively, by evaluating the neural network into the input $\boldsymbol{x}_0$.

Now, we define the local scalar NAVEM space $\mathcal{V}_{h,1}^{\mathcal{NN}}(E)$ as:

$$\mathcal{V}_{h,1}^{\mathcal{NN}}(E) = \operatorname{span}\left\{\varphi_{j,E}^{\mathcal{NN}},\ j = 1,\dots,N_E^v\right\},$$

which represent the approximation spaces for the scalar VEM space $\mathcal{V}_{h,1}(E)$, defined in (6).

The local NAVEM space for the displacement can be defined following a standard strategy, i.e.

$$\boldsymbol{V}_{h,1}^{\mathcal{NN}}(E) := \mathcal{V}_{h,1}^{\mathcal{NN}}(E) \times \mathcal{V}_{h,1}^{\mathcal{NN}}(E), \tag{29}$$

for each element $E \in \mathcal{T}_h$. By gluing together these local spaces, we obtain the global NAVEM discrete space $\boldsymbol{V}_{h,1}^{\mathcal{NN}}$ for the displacement. Finally, we observe that, even if the NAVEM functions are not exactly continuous across elements, we have one degree of freedom for each vertex of $\mathcal{T}_h$ that does not belong to the Dirichlet boundary, independently of the number of elements sharing that vertex, and

$$\dim \boldsymbol{V}_{h,1}^{\mathcal{NN}} = \dim \boldsymbol{V}_{h,1} = N^{\mathrm{dof}}, \tag{30}$$

i.e. the dimension of $\boldsymbol{V}_{h,1}^{\mathcal{NN}}$ is twice the number of these vertices.

Thus, let us denote by $\boldsymbol{v}_N \in \mathbb{R}^{N^{\mathrm{dof}}}$ the vector that collects the degrees of freedom related to the NAVEM function $\boldsymbol{v}_h^{\mathcal{NN}} \in \boldsymbol{V}_{h,1}^{\mathcal{NN}}$. Specifically, each NAVEM function $\boldsymbol{v}_h^{\mathcal{NN}} \in \boldsymbol{V}_{h,1}^{\mathcal{NN}}$ and its gradient can be written as:

$$\boldsymbol{v}_h^{\mathcal{NN}} = \sum_{i=1}^{N^{\mathrm{dof}}} [\boldsymbol{v}_N]_i \boldsymbol{\varphi}_i^{\mathcal{NN}}, \qquad \nabla \boldsymbol{v}_h^{\mathcal{NN}} = \sum_{i=1}^{N^{\mathrm{dof}}} [\boldsymbol{v}_N]_i \nabla \boldsymbol{\varphi}_i^{\mathcal{NN}}, \tag{31}$$

where

$$\boldsymbol{\varphi}_i^{\mathcal{NN}} = \begin{cases} \begin{bmatrix} \varphi_i^{\mathcal{NN}} \\ 0 \end{bmatrix} & \text{if } i = 1,\dots,N^{\mathrm{dof}}/2, \\ \begin{bmatrix} 0 \\ \varphi_{i-N^{\mathrm{dof}}/2}^{\mathcal{NN}} \end{bmatrix} & \text{if } i = N^{\mathrm{dof}}/2+1,\dots,N^{\mathrm{dof}}. \end{cases}$$

Moreover, we introduce the approximated gradient $\widetilde{\nabla}$ for NAVEM functions in $\boldsymbol{V}_{h,1}^{\mathcal{NN}}$ as $\widetilde{\nabla}\boldsymbol{v}_h^{\mathcal{NN}} = \sum_{i=1}^{N^{\mathrm{dof}}} [\boldsymbol{v}_N]_i \widetilde{\nabla}\boldsymbol{\varphi}_i^{\mathcal{NN}}$, where

$$\widetilde{\nabla}\boldsymbol{\varphi}_i^{\mathcal{NN}} = \begin{cases} \begin{bmatrix} \boldsymbol{q}_{1,i}^{\mathcal{NN}} & \boldsymbol{q}_{2,i}^{\mathcal{NN}} \\ 0 & 0 \end{bmatrix} & \text{if } i = 1,\dots,N^{\mathrm{dof}}/2, \\ \begin{bmatrix} 0 & 0 \\ \boldsymbol{q}_{1,i-N^{\mathrm{dof}}/2}^{\mathcal{NN}} & \boldsymbol{q}_{2,i-N^{\mathrm{dof}}/2}^{\mathcal{NN}} \end{bmatrix} & \text{if } i = N^{\mathrm{dof}}/2+1,\dots,N^{\mathrm{dof}}. \end{cases}$$

We observe that to evaluate such functions and their gradients in a generic point $\boldsymbol{x}_{\mathrm{eval}} \in E$, it is sufficient to evaluate functions $h_k \in \mathcal{H}_{j,E}^{\mathcal{NN}}$ and their gradients at $\boldsymbol{x}_{\mathrm{eval}}$, and then use the coefficients $\mathbf{c}^{\varphi}(v_j, E)$ and $\mathbf{c}^{\boldsymbol{q}}(v_j, E)$ (available after step S.3) to evaluate all the basis functions $\varphi_{j,E}^{\mathcal{NN}}$ and their gradients $\boldsymbol{q}_{j,E}^{\mathcal{NN}}$ according to (20) and (26).

Thus, given the output of the online phase S.1–S.3, the NAVEM implementation follows the exact steps performed to solved the elastic problem (3) with a standard Finite Element Method. In particular, the NAVEM discretization for the elastic problem (3) reads

as: *Find* $\boldsymbol{u}_h^{\mathcal{NN}} \in \boldsymbol{V}_{h,1}^{\mathcal{NN}}$ *such that*

$$\sum_{E\in\mathcal{T}_h}\int_E \boldsymbol{\sigma}(\boldsymbol{x},\widetilde{\nabla}\boldsymbol{u}_h^{\mathcal{NN}}) : \widetilde{\nabla}\boldsymbol{v}_h^{\mathcal{NN}} = \sum_{E\in\mathcal{T}_h}\int_E \boldsymbol{f}\cdot\boldsymbol{v}_h^{\mathcal{NN}} \quad \forall \boldsymbol{v}_h^{\mathcal{NN}} \in \boldsymbol{V}_{h,1}^{\mathcal{NN}}. \tag{32}$$

This problem is a non-linear problem that can be solved using a Newton–Raphson solver, eventually paired with an incremental force strategy or arc-length method. Given an initial guess $\boldsymbol{u}_N^0$, the update rule for the Newton–Raphson method related to the NAVEM problem (32) reads as:

$$\begin{aligned} \mathbf{K}_N(\boldsymbol{u}_N^k)\boldsymbol{\delta}_N^k &= -\boldsymbol{r}_N(\boldsymbol{u}_N^k) \\ \boldsymbol{u}_N^{k+1} &= \boldsymbol{u}_N^k + \boldsymbol{\delta}_N^k \end{aligned} \qquad \forall k \geq 0, \tag{33}$$

where, for all $i, j = 1, \dots, N^{\text{dof}}$,

$$\begin{aligned} \mathbf{A}_N(\boldsymbol{u}_N) \in \mathbb{R}^{N^{\text{dof}}} \ : \quad & [\mathbf{A}_N(\boldsymbol{u}_N)]_i = \sum_{E\in\mathcal{T}_h}\int_E \boldsymbol{\sigma}(\boldsymbol{x},\widetilde{\nabla}\boldsymbol{u}_h^{\mathcal{NN}}) : \widetilde{\nabla}\boldsymbol{\varphi}_i^{\mathcal{NN}}, \\ \boldsymbol{f}_N \in \mathbb{R}^{N^{\text{dof}}} \ : \quad & [\boldsymbol{f}_N]_i = \sum_{E\in\mathcal{T}_h}\int_E \boldsymbol{f}\cdot\boldsymbol{\varphi}_i^{\mathcal{NN}}, \\ \mathbf{K}_N(\boldsymbol{u}_N) \in \mathbb{R}^{N^{\text{dof}}\times N^{\text{dof}}} \ : \quad & [\mathbf{K}_N(\boldsymbol{u}_N)]_{ji} = \sum_{E\in\mathcal{T}_h}\int_E \left(\mathbb{A}(\boldsymbol{x},\widetilde{\nabla}\boldsymbol{u}_h^{\mathcal{NN}}) : \widetilde{\nabla}\boldsymbol{\varphi}_i^{\mathcal{NN}}\right) : \widetilde{\nabla}\boldsymbol{\varphi}_j^{\mathcal{NN}}, \end{aligned}$$

$\mathbb{A}$ is the elastic modulus defined in (11), and

$$\boldsymbol{r}_N(\boldsymbol{u}_N) = \mathbf{A}_N(\boldsymbol{u}_N) - \boldsymbol{f}_N. \tag{34}$$

In conclusion, the second part of the NAVEM discretization can be summarized as: Given $\boldsymbol{u}_N^0 \in \mathbb{R}^{N^{\text{dof}}}$, for each $k \geq 0$,

S.4 *assemble* the tangent stiffness matrix $\mathbf{K}_N(\boldsymbol{u}_N^k)$ and the right-hand side $\boldsymbol{r}_N(\boldsymbol{u}_N^k)$.

S.5 *solve* the system (33) to obtain $\boldsymbol{u}_N^{k+1}$.

The process is repeated until a certain stopping criterion is reached.

We observe that Steps **S.4**–**S.5** correspond to the respective steps of a generic finite element solver, since there is no need to add stabilizing terms. Nonetheless, in the NAVEM approach, polygons of arbitrary shapes can be taken into account. We remark that different strategies are available to compute integrals over polygons. The more straightforward one is to sub-triangulate polygons and use triangle quadrature points and weights. A more sophisticated strategy is to use quadrature rules built ad-hoc for general polygons, such as Vianello formulas [41], or compression rules [42]. In this work, we sub-triangulate polygons based on internal point and then we adopt triangle quadrature rules that are exact for polynomials of degree up to 2 as in a generic lowest-order virtual solver [43,44].

Remark 2 (*High-order Extension*)**.** Given $k \geq 1$, let us define the scalar VEM space $\mathcal{V}_{h,k}(E)$ of order k as

$$\begin{aligned} \mathcal{V}_{h,k}(E) = \Big\{ \ & v \in H^1(E) : (i)\ \Delta v \in \mathbb{P}_{k-2}(E) \\ & (ii)\ v_{|e} \in \mathbb{P}_k(e),\ \forall e \in \mathcal{E}_{h,E} \text{ and } v \in C^0(\partial E) \ \Big\}, \end{aligned}$$

which is the natural extension to the high-order of the space $\mathcal{V}_{h,1}(E)$ introduced in (6). See [6] for further details. The degrees of freedom associated with a function $v \in \mathcal{V}_{h,k}(E)$ are the union of:

1. boundary degrees of freedom:
   - the N_E^v values of v at the vertices of E;
   - the $k-1$ values of v at the $k-1$ Gauss–Lobatto quadrature points internal to each edge $e \in \mathcal{E}_{h,E}$;
2. internal degrees of freedom: the $\frac{(k-1)k}{2}$ internal scaled moments: $\frac{1}{|E|}\int_E vp$, $\forall p \in \mathbb{P}_{k-2}(E)$.

The local virtual element space of order k for the displacement is given by $\boldsymbol{V}_{h,k}(E) = \left[\mathcal{V}_{h,k}(E)\right]^2$.

It is thus clear that each component of a virtual function in $\boldsymbol{V}_{h,k}(E)$ is a virtual function $v \in \mathcal{V}_{h,k}(E)$ that can be expressed as the sum $v := v_0 + v_L$ of two functions satisfying

$$\begin{cases} \Delta v_0 = 0 & \text{in } E, \\ v_0 = v & \text{on } \partial E, \end{cases} \qquad \begin{cases} \Delta v_L = \Delta v & \text{in } E, \\ v_L = 0 & \text{on } \partial E. \end{cases}$$

In particular, we note that if v is a Lagrange virtual basis function associated with boundary degrees of freedom of type, then $v = v_0$ and $v_L = 0$. On the other hand, if v is a Lagrange virtual basis function associated with internal degrees of freedom, it holds that $v = v_L$ and $v_0 = 0$.

Since the strategy discussed in this manuscript aims to approximate harmonic functions known in ∂E, it can be used directly to approximate VEM basis functions of order $k \geq 1$ related to the boundary degrees of freedom. An additional neural network may

be used to approximate the basis functions associated with internal degrees of freedom. There exist different possibilities to devise a training strategy for this additional neural network. For example, it may be trained by minimizing the difference between the Laplacian of the approximate basis function and the Laplacian of v_L, which is computable from the internal degrees of freedom, in a PINN-like framework. Alternatively, it may be trained by minimizing the distance among the scaled internal moments of v_L and the known scaled moment of virtual basis functions, asking that $v_L \in \mathcal{H}_{j,E}^{\mathcal{N},\mathcal{N}} \cup \mathbb{P}_k(E)$.

3.5. Theoretical analysis

In the following, we denote by C an arbitrary constant independent of the mesh size h, with different meanings in different occurrences. Moreover, given two positive constants a and b, we also adopt the short-hand notation $a \lesssim b$ if $a \leq Cb$.

In this section, we discuss the theoretical properties of the method in the case of the linear version of the problem (1), in the small deformations regime, with homogeneous Dirichlet boundary conditions:

$$\begin{cases} -(2\mu\nabla\cdot\boldsymbol{\epsilon}(\boldsymbol{u}) + \lambda\nabla\mathrm{div}\boldsymbol{u}) = \boldsymbol{f} & \text{in } \Omega, \\ \boldsymbol{u} = \boldsymbol{0} & \text{on } \Gamma_D = \Gamma, \end{cases} \tag{35}$$

where λ and μ are the positive Lamé coefficients and $\boldsymbol{\epsilon}(\boldsymbol{u}) = \frac{1}{2}(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T)$.

The variational formulation of Problem (35) reads as: *Find* $\boldsymbol{u} \in \boldsymbol{V}$ *such that*

$$a(\boldsymbol{u},\boldsymbol{v}) = f(\boldsymbol{v}), \quad \forall \boldsymbol{v} \in \boldsymbol{V}, \tag{36}$$

where $a : \boldsymbol{V}\times\boldsymbol{V} \to \mathbb{R}$ is the continuous and coercive bilinear form

$$a(\boldsymbol{u},\boldsymbol{v}) = \int_\Omega [2\mu\boldsymbol{\epsilon}(\boldsymbol{u}) : \boldsymbol{\epsilon}(\boldsymbol{v}) + \lambda\mathrm{div}\boldsymbol{u}\ \mathrm{div}\boldsymbol{v}], \tag{37}$$

and

$$f(\boldsymbol{v}) = \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}. \tag{38}$$

We first observe that NAVEM basis functions $\{\boldsymbol{\varphi}_i^{\mathcal{N},\mathcal{N}}\}_{i=1}^{N^{\mathrm{dof}}}$ are not continuous across adjacent elements, since the scalar basis functions $\{\varphi_i^{\mathcal{N},\mathcal{N}}\}_{i=1}^{N^{\mathrm{dof}}/2}$ are computed element-wise. These jumps become smaller and smaller as the accuracy of the employed neural networks increases.

Let us introduce the function $\widetilde{\varphi}_{j,E}^{\mathcal{N},\mathcal{N}} \in H^1(E)$ such that

$$\boldsymbol{q}_{j,E}^{\mathcal{N},\mathcal{N}} = \nabla\widetilde{\varphi}_{j,E}^{\mathcal{N},\mathcal{N}}. \tag{39}$$

Remark 3. Given the expression for $\boldsymbol{q}_{j,E}^{\mathcal{N},\mathcal{N}}$ in (26), the function $\widetilde{\varphi}_{j,E}^{\mathcal{N},\mathcal{N}}$ can be expressed as

$$\widetilde{\varphi}_{j,E}^{\mathcal{N},\mathcal{N}} = \sum_{k=1}^{\dim\mathcal{H}_{j,E}^{\mathcal{N},\mathcal{N}}} c_k^q(v_j,E)h_k + \text{constant}. \tag{40}$$

We observe that there exist infinite functions $\widetilde{\varphi}_{j,E}^{\mathcal{N},\mathcal{N}}$ as the constant in (40) varies. In the following, we fix this constant in such a way $\widetilde{\varphi}_{j,E}^{\mathcal{N},\mathcal{N}}$ has the same mean value as $\varphi_{j,E}$. Thus, the following inequality follows from the second Poincaré inequality [45]:

$$\|\widetilde{\varphi}_{j,E}^{\mathcal{N},\mathcal{N}} - \varphi_{j,E}\|_{L^2(E)} \leq Ch_E\|\nabla\widetilde{\varphi}_{j,E}^{\mathcal{N},\mathcal{N}} - \nabla\varphi_{j,E}\|_{L^2(E)}. \tag{41}$$

We note that such functions define the new spaces

$$\widetilde{\mathcal{V}}_{h,1}^{\mathcal{N},\mathcal{N}} = \mathrm{span}\{\widetilde{\varphi}_i^{\mathcal{N},\mathcal{N}}\}_{i=1}^{N^{\mathrm{dof}}/2}, \qquad \widetilde{\boldsymbol{V}}_{h,1}^{\mathcal{N},\mathcal{N}} = \mathrm{span}\{\widetilde{\boldsymbol{\varphi}}_i^{\mathcal{N},\mathcal{N}}\}_{i=1}^{N^{\mathrm{dof}}}.$$

Now, we define $\widetilde{\boldsymbol{v}}_h^{\mathcal{N},\mathcal{N}} = \sum_{i=1}^{N^{\mathrm{dof}}}[\boldsymbol{v}_N]_i\widetilde{\boldsymbol{\varphi}}_i^{\mathcal{N},\mathcal{N}}$, whose exact gradient is given by

$$\nabla\widetilde{\boldsymbol{v}}_h^{\mathcal{N},\mathcal{N}} = \widetilde{\nabla}\boldsymbol{v}_h^{\mathcal{N},\mathcal{N}}.$$

Let us introduce the broken counterparts of (37) and (38):

$$a_h(\boldsymbol{u},\boldsymbol{v}) = \sum_{E\in\mathcal{T}_h}\int_E [2\mu\boldsymbol{\epsilon}(\boldsymbol{u}) : \boldsymbol{\epsilon}(\boldsymbol{v}) + \lambda\mathrm{div}\boldsymbol{u}\ \mathrm{div}\boldsymbol{v}],$$

$$f_h(\boldsymbol{v}) = \sum_{E\in\mathcal{T}_h}\int_E \boldsymbol{f}\cdot\boldsymbol{v}.$$

The NAVEM discrete version of problem (36) reads as: *Find* $\boldsymbol{u}_N \in \mathbb{R}^{N^{\mathrm{dof}}}$ *such that*

$$a_h(\widetilde{\boldsymbol{u}}_h^{\mathcal{N},\mathcal{N}}, \widetilde{\boldsymbol{v}}_h^{\mathcal{N},\mathcal{N}}) = f_h(\boldsymbol{v}_h^{\mathcal{N},\mathcal{N}}) \quad \forall \boldsymbol{v}_N \in \mathbb{R}^{N^{\mathrm{dof}}}, \tag{42}$$

where $\widetilde{\boldsymbol{v}}_h^{\mathcal{N},\mathcal{N}}$ and $\boldsymbol{v}_h^{\mathcal{N},\mathcal{N}}$ share the same set of coefficients $\boldsymbol{v}_N$ with respect to the basis of and $\widetilde{\boldsymbol{V}}_{h,1}^{\mathcal{N},\mathcal{N}}$ and $\boldsymbol{V}_{h,1}^{\mathcal{N},\mathcal{N}}$, respectively, and $\boldsymbol{u}_N$ is the set of coefficients of $\widetilde{\boldsymbol{u}}_h^{\mathcal{N},\mathcal{N}} \in \widetilde{\boldsymbol{V}}_{h,1}^{\mathcal{N},\mathcal{N}}$.

Remark 4. We observe that using arguments provided in [17], the bilinear form $a_h(\cdot,\cdot)$ is coercive on $\widetilde{\boldsymbol{V}}_{h,1}^{\mathcal{N}\mathcal{N}}$.

In [21], the following local estimates are proved for the basis functions $\{\varphi_j^{\mathcal{N}\mathcal{N}}\}_{j=1}^{N^{\text{dof}}/2}$ of $\mathcal{V}_{h,1}$ and for $\{\widetilde{\varphi}_j^{\mathcal{N}\mathcal{N}}\}_{j=1}^{N^{\text{dof}}/2}$. To highlight the dependence on the mesh size h, we report the proofs of these propositions.

Proposition 1. *Given a polygon E, and for all $j = 1, \dots, N_E^v$, it holds*

$$\|\varphi_{j,E} - \varphi_{j,E}^{\mathcal{N}\mathcal{N}}\|_{L^\infty(\partial E)} \le C\epsilon_{j,E}^{\varphi}, \qquad \|\varphi_{j,E} - \varphi_{j,E}^{\mathcal{N}\mathcal{N}}\|_{L^2(E)} \le Ch_E\epsilon_{j,E}^{\varphi}.$$

Proof. The first inequality follows from [17,21]. Concerning the second inequality, in [21, Proposition 1] we observe that

$$\begin{aligned}\|\varphi_{j,E} - \varphi_{j,E}^{\mathcal{N}\mathcal{N}}\|_{L^2(E)}^2 &= \int_E (\varphi_{j,E} - \varphi_{j,E}^{\mathcal{N}\mathcal{N}})^2 \\ &\lesssim |E|\|\varphi_{j,E} - \varphi_{j,E}^{\mathcal{N}\mathcal{N}}\|_{L^\infty(E)} \\ &\lesssim |E|\|\varphi_{j,E} - \varphi_{j,E}^{\mathcal{N}\mathcal{N}}\|_{L^\infty(\partial E)} \\ &\lesssim \left(\epsilon_{j,E}^{\varphi}\right)^2 |E| \le C\left(\epsilon_{j,E}^{\varphi}\right)^2 h_E^2,\end{aligned}$$

where we use the harmonicity of $\varphi_{j,E}$ and $\varphi_{j,E}^{\mathcal{N}\mathcal{N}}$ to bound $\|\varphi_{j,E} - \varphi_{j,E}^{\mathcal{N}\mathcal{N}}\|_{L^\infty(E)}$ by $\|\varphi_{j,E} - \varphi_{j,E}^{\mathcal{N}\mathcal{N}}\|_{L^\infty(\partial E)}$. This concludes the proof. □

Proposition 2. *Given a polygon E, and for all $j = 1, \dots, N_E^v$,*

$$\|\nabla\varphi_{j,E} - \widetilde{\nabla}\varphi_{j,E}^{\mathcal{N}\mathcal{N}}\|_{L^2(E)} \le Ch_E^{1/2}\epsilon_{j,E}^{\boldsymbol{q}}, \tag{43}$$

and

$$\|\varphi_{j,E} - \widetilde{\varphi}_{j,E}^{\mathcal{N}\mathcal{N}}\|_{H^1(E)} = \|\varphi_{j,E} - \widetilde{\varphi}_{j,E}^{\mathcal{N}\mathcal{N}}\|_{L^2(E)} + \|\nabla\varphi_{j,E} - \widetilde{\nabla}\varphi_{j,E}^{\mathcal{N}\mathcal{N}}\|_{L^2(E)} \le C\left(h_E^{3/2} + h_E^{1/2}\right)\epsilon_{j,E}^{\boldsymbol{q}}. \tag{44}$$

Proof. Let us define $\tilde{\Psi}_{j,E} = \varphi_{j,E} - \widetilde{\varphi}_{j,E}^{\mathcal{N}\mathcal{N}}$. Due to Remark 3, $\tilde{\Psi}_{j,E}$ has zero mean value. Thus, from the trace theorem [46], we have

$$\|\tilde{\Psi}_{j,E}\|_{L^2(\partial E)} \le Ch^{1/2}\|\nabla\tilde{\Psi}_{j,E}\|_{L^2(E)}. \tag{45}$$

Furthermore, since $\tilde{\Psi}_{j,E}$ is harmonic, we obtain

$$\begin{aligned}\|\nabla\tilde{\Psi}_{j,E}\|_{L^2(E)}^2 &= \int_E \nabla\tilde{\Psi}_{j,E}\cdot\nabla\tilde{\Psi}_{j,E} = \int_{\partial E}\tilde{\Psi}_{j,E}\nabla\tilde{\Psi}_{j,E}\cdot\boldsymbol{n} \\ &\le \|\tilde{\Psi}_{j,E}\|_{L^2(\partial E)}\|\nabla\tilde{\Psi}_{j,E}\cdot\boldsymbol{n}\|_{L^2(\partial E)} \\ &\le Ch_E^{1/2}\|\nabla\tilde{\Psi}_{j,E}\|_{L^2(E)}\|\nabla\tilde{\Psi}_{j,E}\cdot\boldsymbol{n}\|_{L^2(\partial E)}.\end{aligned}$$

Thus,

$$\|\nabla\tilde{\Psi}_{j,E}\|_{L^2(E)} \le Ch_E^{1/2}\|\nabla\tilde{\Psi}_{j,E}\cdot\boldsymbol{n}\|_{L^2(\partial E)}.$$

Now, we recall that the $L^2(\partial E)$-norm of the tangential derivatives is equivalent to the $L^2(\partial E)$-norm of the normal derivative for harmonic functions, as a consequence of Rellich's Identity [47, Corollary 2.7]. Thus, we can conclude that

$$\begin{aligned}\|\boldsymbol{q}_{j,E}^{\mathcal{N}\mathcal{N}} - \nabla\varphi_{j,E}\|_{L^2(E)} &= \|\nabla\tilde{\Psi}_{j,E}\|_{L^2(E)} \\ &\lesssim h_E^{1/2}\|\nabla\tilde{\Psi}_{j,E}\cdot\boldsymbol{n}\|_{L^2(\partial E)} \\ &\lesssim h_E^{1/2}\|\nabla\tilde{\Psi}_{j,E}\cdot\boldsymbol{t}\|_{L^2(\partial E)} \\ &= Ch_E^{1/2}\|\boldsymbol{q}_{j,E}^{\mathcal{N}\mathcal{N}}\cdot\boldsymbol{t} - \nabla\varphi_{j,E}\cdot\boldsymbol{t}\|_{L^2(\partial E)} = Ch_E^{1/2}\mathcal{L}_{j,E}^1. \quad \square\end{aligned}$$

Let us introduce the following constants

$$\epsilon^{\varphi} = \max_{E\in\mathcal{T}_h}\max_{j=1,\dots,N_E^{\text{dof}}/2}\epsilon_{j,E}^{\varphi} \qquad \text{and} \qquad \epsilon^{\boldsymbol{q}} = \max_{E\in\mathcal{T}_h}\max_{j=1,\dots,N_E^{\text{dof}}/2}\epsilon_{j,E}^{\boldsymbol{q}}, \tag{46}$$

that express the maximum values of test losses. Let us adopt the following notation, for all $e \in \mathcal{E}_h$,

$$[\![\boldsymbol{w}]\!]_e = \begin{cases}\boldsymbol{w}_{|E_e^1}, & \text{on } e \in \mathcal{E}_{h,\partial\Omega}, \\ (\boldsymbol{w}_{|E_e^1} - \boldsymbol{w}_{|E_e^2}), & \text{on } e \in \mathcal{E}_h\backslash\mathcal{E}_{h,\partial\Omega}.\end{cases}$$

Here $\mathcal{E}_{h,\partial\Omega}$ denotes the set of edges on $\partial\Omega$, E_e^1 and E_e^2 are the two elements whose borders contain the edge e, and $\boldsymbol{w}_{|E}$ denotes the function $\boldsymbol{w}$ defined over the element E. The vectors τ and $\boldsymbol{n}$ are the tangent and normal unit vectors of e from the perspective of E_e^1, the corresponding vectors from the perspective of E_e^2 are $-\boldsymbol{\tau}$ and $-\boldsymbol{n}$.

Lemma 1. *For each edge $e \in \mathcal{E}_h$ and for all $i = 1, \dots, N^{\text{dof}}$, it holds*

$$\|[\![\widetilde{\varphi}_i^{\mathcal{NN}}]\!]_e\|_{L^2(e)} \le Ch\epsilon^{q}. \tag{47}$$

Proof. Let us first observe that

$$\begin{aligned}\|[\![\widetilde{\varphi}_i^{\mathcal{NN}}]\!]_e\|_{L^2(e)} &= \|\widetilde{\varphi}_{i|E_e^1}^{\mathcal{NN}} - \varphi_{i|E_e^1} + \varphi_{i|E_e^2} - \widetilde{\varphi}_{i|E_e^2}^{\mathcal{NN}}\|_{L^2(e)} \\ &\le \|\widetilde{\varphi}_{i|E_e^1}^{\mathcal{NN}} - \varphi_{i|E_e^1}\|_{L^2(e)} + \|\varphi_{i|E_e^2} - \widetilde{\varphi}_{i|E_e^2}^{\mathcal{NN}}\|_{L^2(e)}.\end{aligned}$$

Now, for both $k = 1, 2$, it holds, as a consequence of the trace theorem (54),

$$\begin{aligned}\|\widetilde{\varphi}_{i|E_e^k}^{\mathcal{NN}} - \varphi_{i|E_e^k}\|_{L^2(e)} &\le \|\widetilde{\varphi}_{i|E_e^k}^{\mathcal{NN}} - \varphi_{i|E_e^k}\|_{L^2(\partial E_e^k)} \\ &\le Ch^{1/2}\|\nabla\widetilde{\varphi}_{i|E_e^k}^{\mathcal{NN}} - \nabla\varphi_{i|E_e^k}^{\mathcal{NN}}\|_{L^2(E_e^k)} \\ &\le Ch^{1/2}h^{1/2}\epsilon^{q}.\end{aligned}$$

This concludes the proof. □

In the following, we denote by $\|\cdot\|_{H^1(\Omega)}$ the broken version of the standard Sobolev norm in $H^1(\Omega)$.

Lemma 2. *Given an edge $e \in \mathcal{E}_h$, let us denote by $\mathcal{I}_{\varphi}^e$ the subset of indices i related to virtual basis functions that contain the two elements E_1 and E_2 adjacent to e in their support. For each edge $e \in \mathcal{E}_h$, we have*

$$\sum_{i\in\mathcal{I}_{\varphi}^e} \|[\boldsymbol{v}_N]_i[\![\widetilde{\boldsymbol{\varphi}}_i^{\mathcal{NN}}]\!]_e \cdot \boldsymbol{n}\|_{L^2(e)} \le C\epsilon^{q}\|\widetilde{\boldsymbol{v}}_h^{\mathcal{NN}}\|_{H^1(\Omega)}. \tag{48}$$

Proof. Since we are focusing on a two-dimensional problem, the assumption of quasi-uniform mesh implies $N^{\text{dof}} \approx h^{-2}$ and $|e| \approx h$. Thus, it holds [48, Lemma 4]

$$\|\boldsymbol{v}_N\|_2 \le Ch^{-1}\|\widetilde{\boldsymbol{v}}_h^{\mathcal{NN}}\|_{H^1(\Omega)}, \tag{49}$$

where $\|\cdot\|_2$ denotes the standard euclidean norm. We observe that the set $\mathcal{I}_{\varphi}^e$ contains at most N_{φ}^e edges (independently of h, and N^{dof}, because of the mesh regularity assumption). By using Cauchy-Schwartz, we can obtain, for each $e \in \mathcal{E}_h$,

$$\begin{aligned}\sum_{i\in\mathcal{I}_{\varphi}^e} \|[\boldsymbol{v}_N]_i[\![\widetilde{\boldsymbol{\varphi}}_i^{\mathcal{NN}}]\!]_e \cdot \boldsymbol{n}\|_{L^2(e)} &\le \sum_{i\in\mathcal{I}_{\varphi}^e} |[\boldsymbol{v}_N]_i|\ \|[\![\widetilde{\boldsymbol{\varphi}}_i^{\mathcal{NN}}]\!]_e \cdot \boldsymbol{n}\|_{L^2(e)} \\ &\le \sqrt{\sum_{i\in\mathcal{I}_{\varphi}^e}[\boldsymbol{v}_N]_i^2}\sqrt{\sum_{i\in\mathcal{I}_{\varphi}^e}\|[\![\widetilde{\boldsymbol{\varphi}}_i^{\mathcal{NN}}]\!]_e \cdot \boldsymbol{n}\|_{L^2(e)}^2} \\ &\le \|\boldsymbol{v}_N\|_2\sqrt{\sum_{i\in\mathcal{I}_{\varphi}^e}\|[\![\widetilde{\boldsymbol{\varphi}}_i^{\mathcal{NN}}]\!]_e \cdot \boldsymbol{n}\|_{L^2(e)}^2} \\ &\le Ch^{-1}\|\widetilde{\boldsymbol{v}}_h^{\mathcal{NN}}\|_{H^1(\Omega)}\sqrt{N_{\varphi}^e(Ch\epsilon^{q})^2} \\ &= C\epsilon^{q}\|\widetilde{\boldsymbol{v}}_h^{\mathcal{NN}}\|_{H^1(\Omega)},\end{aligned}$$

where we exploit Lemma 1. □

With the same arguments, the following bound can also be proved for each edge $e \in \mathcal{E}_h$:

$$\sum_{i\in\mathcal{I}_{\varphi}^e} \|[\boldsymbol{v}_N]_i[\![\widetilde{\boldsymbol{\varphi}}_i^{\mathcal{NN}}]\!]_e \cdot \boldsymbol{\tau}\|_{L^2(e)} \le C\epsilon^{q}\|\widetilde{\boldsymbol{v}}_h^{\mathcal{NN}}\|_{H^1(\Omega)}. \tag{50}$$

Following arguments provided in [17,49], the following a priori error estimate can be proved.

Theorem 1. *Let $\boldsymbol{u} \in H^2(\Omega)$ be the solution of Problem (35) and let $\widetilde{\boldsymbol{u}}_h^{\mathcal{NN}}$ be the solution of the discrete NAVEM problem (42). Then, the following a priori error bound holds:*

$$\|\boldsymbol{u} - \widetilde{\boldsymbol{u}}_h^{\mathcal{NN}}\|_{H^1(\Omega)} \le C\left(h + h^{-3/2}\epsilon^{q} + h^{-2}\epsilon^{\varphi}\right)\|\boldsymbol{f}\|_{L^2(\Omega)}. \tag{51}$$

Proof. Since the bilinear form a_h is coercive on $\widetilde{\boldsymbol{V}}_{h,1}^{\mathcal{NN}}$ (see Remark 4), we have, for all $\widetilde{\boldsymbol{v}}_h^{\mathcal{NN}} \in \widetilde{\boldsymbol{V}}_{h,1}^{\mathcal{NN}}$,

$$\begin{aligned}\alpha\|\widetilde{\boldsymbol{u}}_h^{\mathcal{NN}} - \widetilde{\boldsymbol{v}}_h^{\mathcal{NN}}\|_{H^1(\Omega)}^2 &\le a_h(\widetilde{\boldsymbol{u}}_h^{\mathcal{NN}} - \widetilde{\boldsymbol{v}}_h^{\mathcal{NN}}, \widetilde{\boldsymbol{u}}_h^{\mathcal{NN}} - \widetilde{\boldsymbol{v}}_h^{\mathcal{NN}}) \\ &= a_h(\boldsymbol{u} - \widetilde{\boldsymbol{v}}_h^{\mathcal{NN}}, \widetilde{\boldsymbol{u}}_h^{\mathcal{NN}} - \widetilde{\boldsymbol{v}}_h^{\mathcal{NN}}) \\ &\quad + a_h(\widetilde{\boldsymbol{u}}_h^{\mathcal{NN}}, \widetilde{\boldsymbol{u}}_h^{\mathcal{NN}} - \widetilde{\boldsymbol{v}}_h^{\mathcal{NN}}) - f_h(\boldsymbol{u}_h^{\mathcal{NN}} - \boldsymbol{v}_h^{\mathcal{NN}}) \\ &\quad + f_h(\boldsymbol{u}_h^{\mathcal{NN}} - \boldsymbol{v}_h^{\mathcal{NN}}) - a_h(\boldsymbol{u}, \widetilde{\boldsymbol{u}}_h^{\mathcal{NN}} - \widetilde{\boldsymbol{v}}_h^{\mathcal{NN}})\end{aligned}$$

$$= a_h(\boldsymbol{u} - \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}, \widetilde{\boldsymbol{u}}_h^{\mathcal{N}\mathcal{N}} - \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}) + f_h(\boldsymbol{u}_h^{\mathcal{N}\mathcal{N}} - \boldsymbol{v}_h^{\mathcal{N}\mathcal{N}}) - a(\boldsymbol{u}, \widetilde{\boldsymbol{u}}_h^{\mathcal{N}\mathcal{N}} - \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}),$$

since $\widetilde{\boldsymbol{u}}_h^{\mathcal{N}\mathcal{N}}$ satisfies Problem (42). By exploiting the continuity of the bilinear form a_h, we retrieve

$$\alpha\|\widetilde{\boldsymbol{u}}_h^{\mathcal{N}\mathcal{N}} - \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)} \le C\|\boldsymbol{u} - \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)} + \frac{|f_h(\boldsymbol{u}_h^{\mathcal{N}\mathcal{N}} - \boldsymbol{v}_h^{\mathcal{N}\mathcal{N}}) - a_h(\boldsymbol{u}, \widetilde{\boldsymbol{u}}_h^{\mathcal{N}\mathcal{N}} - \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}})|}{\|\widetilde{\boldsymbol{u}}_h^{\mathcal{N}\mathcal{N}} - \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)}}.$$

As a consequence, using the triangle inequality,

$$\begin{aligned}\|\boldsymbol{u} - \widetilde{\boldsymbol{u}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)} &\le \|\boldsymbol{u} - \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)} + \|\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}} - \widetilde{\boldsymbol{u}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)} \\ &\lesssim \|\boldsymbol{u} - \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)} + \frac{|f_h(\boldsymbol{u}_h^{\mathcal{N}\mathcal{N}} - \boldsymbol{v}_h^{\mathcal{N}\mathcal{N}}) - a_h(\boldsymbol{u}, \widetilde{\boldsymbol{u}}_h^{\mathcal{N}\mathcal{N}} - \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}})|}{\|\widetilde{\boldsymbol{u}}_h^{\mathcal{N}\mathcal{N}} - \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)}}.\end{aligned}$$

Thus, following [49], we have

$$\|\boldsymbol{u} - \widetilde{\boldsymbol{u}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)} \lesssim \inf_{\boldsymbol{v}_N \in \mathbb{R}^{N^{\mathrm{dof}}}} \|\boldsymbol{u} - \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)} + \sup_{\boldsymbol{v}_N \in \mathbb{R}^{N^{\mathrm{dof}}}} \frac{|f_h(\boldsymbol{v}_h^{\mathcal{N}\mathcal{N}}) - a_h(\boldsymbol{u}, \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}})|}{\|\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)}}. \tag{52}$$

Let us initially analyze the numerator of the second term of (52). By exploiting the fact that $\boldsymbol{u} \in H^2(\Omega)$ is the solution of Problem (35) along with the classical integration by parts formula [50], we obtain

$$\begin{aligned}a_h(\boldsymbol{u}, \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}) - f_h(\boldsymbol{v}_h^{\mathcal{N}\mathcal{N}}) &= \sum_{E\in\mathcal{T}_h}\left[2\mu\int_E \boldsymbol{\epsilon}(\boldsymbol{u}) : \boldsymbol{\epsilon}(\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}) + \int_E \lambda \mathrm{div}\boldsymbol{u}\ \mathrm{div}\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}} - \int_E \boldsymbol{f}\cdot\boldsymbol{v}_h^{\mathcal{N}\mathcal{N}}\right] \\ &= \sum_{E\in\mathcal{T}_h}\Big[\int_E (-2\mu\nabla\cdot\boldsymbol{\epsilon}(\boldsymbol{u}) - \lambda\nabla\mathrm{div}\boldsymbol{u} - \boldsymbol{f})\cdot\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}} \\ &\quad + \int_E \boldsymbol{f}\cdot\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}} - \int_E \boldsymbol{f}\cdot\boldsymbol{v}_h^{\mathcal{N}\mathcal{N}} + \sum_{e\in\mathcal{E}_{h,E}}\int_e A(\boldsymbol{u})\,\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}\cdot\boldsymbol{n} + \sum_{e\in\mathcal{E}_{h,E}}\int_e B(\boldsymbol{u})\,\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}\cdot\boldsymbol{\tau}\Big] \\ &= \sum_{E\in\mathcal{T}_h}\int_E \boldsymbol{f}\cdot(\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}} - \boldsymbol{v}_h^{\mathcal{N}\mathcal{N}}) + \sum_{e\in\mathcal{E}_h}\int_e A(\boldsymbol{u})\,[\![\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}]\!]_e\cdot\boldsymbol{n} + \sum_{e\in\mathcal{E}_h}\int_e B(\boldsymbol{u})\,[\![\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}]\!]_e\cdot\boldsymbol{\tau},\end{aligned}$$

where we have defined the following linear operators

$$A(\boldsymbol{u}) = 2\mu\boldsymbol{n}^T\boldsymbol{\epsilon}(\boldsymbol{u})\boldsymbol{n} + \lambda\mathrm{div}\boldsymbol{u},$$
$$B(\boldsymbol{u}) = 2\mu\boldsymbol{\tau}^T\boldsymbol{\epsilon}(\boldsymbol{u})\boldsymbol{n}.$$

We observe that it holds

$$\sum_{e\in\mathcal{E}_h}\|A(u)\|_{L^2(e)} \le Ch^{-1/2}\|\boldsymbol{u}\|_{H^2(\Omega)}, \qquad \sum_{e\in\mathcal{E}_h}\|B(u)\|_{L^2(e)} \le Ch^{-1/2}\|\boldsymbol{u}\|_{H^2(\Omega)}. \tag{53}$$

Indeed, we have, for instance,

$$\begin{aligned}\sum_{e\in\mathcal{E}_h}\|B(u)\|_{L^2(e)} &\lesssim \sum_{E\in\mathcal{T}_h}\|\boldsymbol{\tau}^T\boldsymbol{\epsilon}(\boldsymbol{u})\boldsymbol{n}\|_{L^2(\partial E)} \\ &\lesssim \sum_{E\in\mathcal{T}_h}\|\boldsymbol{\epsilon}(\boldsymbol{u})\|_{L^2(\partial E)} \\ &\lesssim \sum_{E\in\mathcal{T}_h}\left(h^{-1/2}\|\boldsymbol{\epsilon}(\boldsymbol{u})\|_{L^2(E)} + h^{1/2}|\boldsymbol{\epsilon}(\boldsymbol{u})|_{H^1(E)}\right) \qquad (54) \\ &\lesssim \sum_{E\in\mathcal{T}_h}\left(h^{-1/2}\|\nabla\boldsymbol{u}\|_{L^2(E)} + h^{1/2}|\nabla\boldsymbol{u}|_{H^1(E)}\right) \\ &\lesssim \left(h^{1/2} + h^{-1/2}\right)\sum_{E\in\mathcal{T}_h}\|\nabla\boldsymbol{u}\|_{H^1(E)} \\ &\lesssim \left(h^{1/2} + h^{-1/2}\right)\|\nabla\boldsymbol{u}\|_{H^1(\Omega)} \lesssim h^{-1/2}\|\boldsymbol{u}\|_{H^2(\Omega)},\end{aligned}$$

where the step (54) follows from the trace theorem [46, Lemma 2.1]. The bound related to A can be proved similarly.

By exploiting Lemma 2 and Eq. (50), we can prove that:

$$a_h(\boldsymbol{u}, \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}) - f_h(\boldsymbol{v}_h^{\mathcal{N}\mathcal{N}}) \lesssim \epsilon^q h^{-1/2}\|\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)}\|\boldsymbol{u}\|_{H^2(\Omega)} + \sum_{E\in\mathcal{T}_h}\int_E \boldsymbol{f}\cdot(\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}} - \boldsymbol{v}_h^{\mathcal{N}\mathcal{N}}). \tag{55}$$

The last term in Eq. (55) can be bounded as:

$$\sum_{E\in\mathcal{T}_h}\int_E \boldsymbol{f}\cdot(\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}} - \boldsymbol{v}_h^{\mathcal{N}\mathcal{N}}) \le \sum_{E\in\mathcal{T}_h}\|\boldsymbol{f}\|_{L^2(E)}\left\|\sum_{i=1}^{N^{\mathrm{dof}}}[\boldsymbol{v}_N]_i\left(\widetilde{\boldsymbol{\varphi}}_i^{\mathcal{N}\mathcal{N}} - \boldsymbol{\varphi}_i^{\mathcal{N}\mathcal{N}}\right)\right\|_{L^2(E)}$$

$$\leq \|\boldsymbol{f}\|_{L^2(\Omega)}\|\boldsymbol{v}_N\|_2 \sum_{E\in\mathcal{T}_h}\sum_{i=1}^{N^{\text{dof}}} \|\widetilde{\boldsymbol{\varphi}}_i^{\mathcal{N}\mathcal{N}} - \boldsymbol{\varphi}_i^{\mathcal{N}\mathcal{N}}\|_{L^2(E)}$$

$$\leq \|\boldsymbol{f}\|_{L^2(\Omega)} \left(Ch^{-1}\|\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)}\right) \sum_{E\in\mathcal{T}_h}\sum_{i=1}^{N^{\text{dof}}} \left(\|\widetilde{\boldsymbol{\varphi}}_i^{\mathcal{N}\mathcal{N}} - \boldsymbol{\varphi}_i\|_{L^2(E)} + \|\boldsymbol{\varphi}_i^{\mathcal{N}\mathcal{N}} - \boldsymbol{\varphi}_i\|_{L^2(E)}\right),$$

where the last inequality follows from (49) and the triangle inequality. By combining results obtained in (41), Propositions 1 and 2, and the assumption $N^{\text{dof}} \approx h^{-2}$, we obtain

$$\begin{aligned}\sum_{E\in\mathcal{T}_h}\sum_{i=1}^{N^{\text{dof}}} \left(\|\widetilde{\boldsymbol{\varphi}}_i^{\mathcal{N}\mathcal{N}} - \boldsymbol{\varphi}_i\|_{L^2(E)} + \|\boldsymbol{\varphi}_i^{\mathcal{N}\mathcal{N}} - \boldsymbol{\varphi}_i\|_{L^2(E)}\right) &\lesssim \sum_{E\in\mathcal{T}_h}\sum_{i=1}^{N^{\text{dof}}} \left(h\|\nabla\boldsymbol{\varphi}_i - \nabla\widetilde{\boldsymbol{\varphi}}_i^{\mathcal{N}\mathcal{N}}\|_{L^2(E)} + h\epsilon^{\varphi}\right)\\ &\lesssim N^{\text{dof}}\left(h^{3/2}\epsilon^{\boldsymbol{q}} + h\epsilon^{\varphi}\right)\\ &\lesssim h^{-2}\left(h^{3/2}\epsilon^{\boldsymbol{q}} + h\epsilon^{\varphi}\right).\end{aligned}$$

Thus, we conclude that

$$\sum_{E\in\mathcal{T}_h}\int_E \boldsymbol{f}\cdot(\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}} - \boldsymbol{v}_h^{\mathcal{N}\mathcal{N}}) \lesssim h^{-3}\left(h^{3/2}\epsilon^{\boldsymbol{q}} + h\epsilon^{\varphi}\right)\|\boldsymbol{f}\|_{L^2(\Omega)}\|\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)}.$$

By assuming that each element in the tessellation is convex, so that we can use the usual elliptic regularity result, i.e. $\|\boldsymbol{u}\|_{H^2(\Omega)} \lesssim \|\boldsymbol{f}\|_{L^2(\Omega)}$, we rewrite the last term in (52) as:

$$\sup_{\boldsymbol{v}_N\in\mathbb{R}^{N^{\text{dof}}}} \frac{|a_h(\boldsymbol{u},\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}) - f_h(\boldsymbol{v}_h^{\mathcal{N}\mathcal{N}})|}{\|\widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)}} \leq C\left(h^{-1/2}\epsilon^{\boldsymbol{q}} + h^{-3/2}\epsilon^{\boldsymbol{q}} + h^{-2}\epsilon^{\varphi}\right)\|\boldsymbol{f}\|_{L^2(\Omega)}. \tag{56}$$

Let $I_h\boldsymbol{u} \in \boldsymbol{V}_{h,1}$, $I_h^{\mathcal{N}\mathcal{N}}\boldsymbol{u} \in \boldsymbol{V}_{h,1}^{\mathcal{N}\mathcal{N}}$, and $\widetilde{I}_h^{\mathcal{N}\mathcal{N}}\boldsymbol{u} \in \widetilde{\boldsymbol{V}}_{h,1}^{\mathcal{N}\mathcal{N}}$ be the standard nodal interpolants of $\boldsymbol{u}$ in the three spaces. We now consider the first term of (52), and derive a bound by using classical VEM theoretical results in [6] and Propositions 1 and 2, following steps performed in [17],

$$\begin{aligned}\inf_{\boldsymbol{v}_N\in\mathbb{R}^{N^{\text{dof}}}} \|\boldsymbol{u} - \widetilde{\boldsymbol{v}}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)} &\leq \|\boldsymbol{u} - \widetilde{I}_h^{\mathcal{N}\mathcal{N}}\boldsymbol{u}\|_{H^1(\Omega)}\\ &\leq \|\boldsymbol{u} - I_h\boldsymbol{u}\|_{H^1(\Omega)} + \|I_h\boldsymbol{u} - \widetilde{I}_h^{\mathcal{N}\mathcal{N}}\boldsymbol{u}\|_{H^1(\Omega)}\\ &\leq Ch\|\boldsymbol{u}\|_{H^2(\Omega)} + \sum_{\pi=1}^{N^{\text{dof}}}\sum_{E\in\mathcal{T}_h} |[\boldsymbol{u}]_i|\|\boldsymbol{\varphi}_i - \tilde{\boldsymbol{\varphi}}_i^{\mathcal{N}\mathcal{N}}\|_{H^1(E)}\\ &\leq Ch\|\boldsymbol{u}\|_{H^2(\Omega)} + Ch^{-2}\|\boldsymbol{u}\|_{H^2(\Omega)}\left(h^{3/2} + h^{1/2}\right)\epsilon^{\boldsymbol{q}}\\ &\leq C\left(h + \left(h^{-1/2} + h^{-3/2}\right)\epsilon^{\boldsymbol{q}}\right)\|\boldsymbol{f}\|_{L^2(\Omega)},\end{aligned} \tag{57}$$

where $[\boldsymbol{u}]_i$ represents the ith nodal value of the exact solution $\boldsymbol{u}$, and we used the fact that we are summing over $N^{\text{dof}} \approx h^{-2}$ terms.

Bounding (52) by (56) and (57), we get the result (51) and conclude the proof. □

We highlight that, in [21], the error estimate

$$\|\boldsymbol{u} - \boldsymbol{u}_h^{\mathcal{N}\mathcal{N}}\|_{H^1(\Omega)} \leq C(h + h^{-2}\epsilon^{\varphi}) \tag{58}$$

was derived without taking into account the inconsistency error due to the usage of two different networks for approximating the functions and their gradients (see Remark 1). Here, the additional terms appear because we are considering both networks in the derivation of the new error estimate. We observe that, if ϵ^{φ} and $\epsilon^{\boldsymbol{q}}$ are of the same magnitude, the limiting factor scales like $h^{-2}\epsilon^{\varphi}$, making this additional inconsistency negligible.

As in the elliptic case [21], Eq. (51) implies that, if the neural networks are accurate enough (i.e. trained to reach a sufficiently low value of the loss function on a representative enough training dataset), the convergence rate for the error in $H^1(\Omega)$-norm decreases as $O(h)$, otherwise the accuracy of the neural networks represents the limiting factor.

4. Numerical results

In this section, we test the discussed method on linear and non-linear elasticity problems using different meshes. We highlight that in all the experiments, the NAVEM basis functions are predicted by exploiting the same neural networks built to perform the experiment in Section 3.3.

Denoting by $\boldsymbol{u}$ the exact displacement, we test the performance of the NAVEM by looking at the behavior of the following errors

$$\begin{aligned}\text{err}_0^N &= \sqrt{\sum_{E\in\mathcal{T}_h}\|\boldsymbol{u} - \boldsymbol{u}_h^{\mathcal{N}\mathcal{N}}\|_{L^2(E)}^2},\\ \text{err}_1^N &= \sqrt{\sum_{E\in\mathcal{T}_h}\|\nabla\boldsymbol{u} - \widetilde{\nabla}\boldsymbol{u}_h^{\mathcal{N}\mathcal{N}}\|_{L^2(E)}^2},\end{aligned} \tag{59}$$

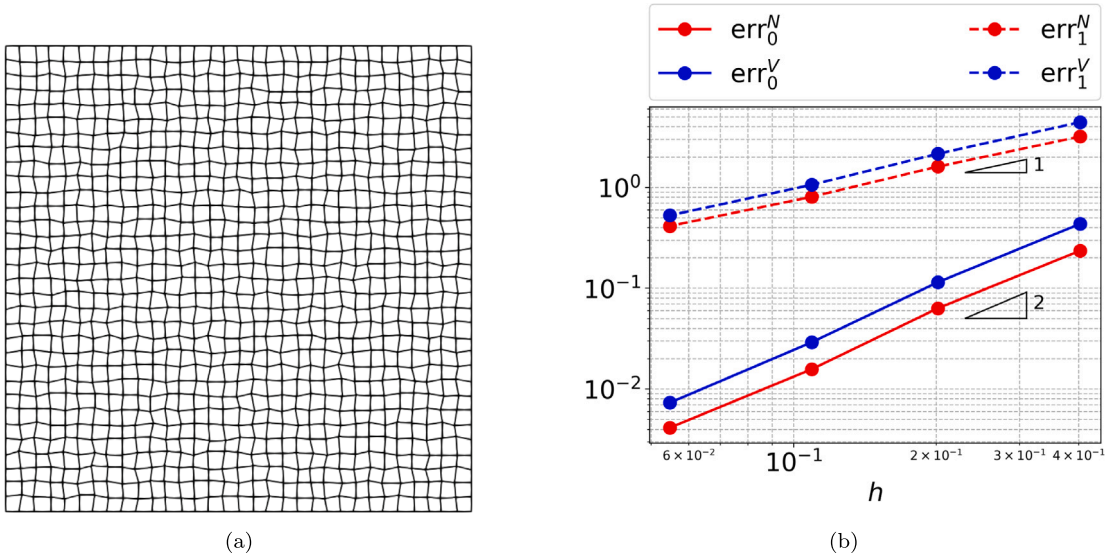


Fig. 2. Test 1: Left: Last refinement of the Distorted-Square family of meshes. Right: Behavior of errors (59)–(60) as the mesh parameter h decreases.

as the mesh parameter h decreases. For comparison purposes, we also solve these problems with the standard virtual element method. In this case, since we cannot access the point-wise evaluation of virtual functions, we define the VEM errors as usual (see, for instance, [7]), that is

$$\begin{aligned} \mathrm{err}_0^V &= \sqrt{\sum_{E\in\mathcal{T}_h} \|\boldsymbol{u} - \Pi_1^{0,E}\boldsymbol{u}_h^V\|^2_{L^2(E)}}, \\ \mathrm{err}_1^V &= \sqrt{\sum_{E\in\mathcal{T}_h} \|\nabla\boldsymbol{u} - \Pi_0^{0,E}\nabla\boldsymbol{u}_h^V\|^2_{L^2(E)}}, \end{aligned} \tag{60}$$

where we denote by $\boldsymbol{u}_h^V$ the continuous virtual element solution. We remark that to evaluate the high-order projection $\Pi_1^{0,E}\boldsymbol{u}_h^V$ in the virtual element method, an enhanced version of the local space should be taken into account [51].

4.1. Test 1: The linear case - convergence test

In this test, we consider the square domain $\Omega = (0,1)^2$ and the linear problem (36). Here, we take the material constant values $\mu = 1.5$, $\lambda = 3$ and we set the boundary conditions and the body load $\boldsymbol{f}$ in such a way the exact displacement is:

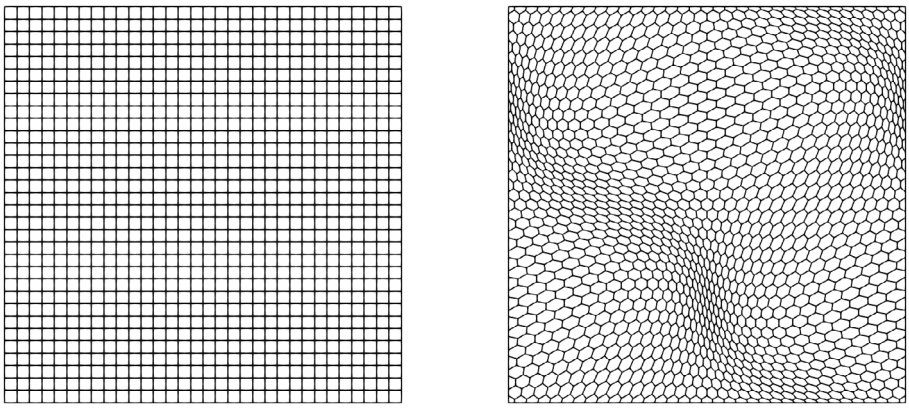
$$\boldsymbol{u}(\boldsymbol{x}) = \begin{bmatrix} u_1(\boldsymbol{x}) \\ 5u_1(\boldsymbol{x}) \end{bmatrix}, \qquad u_1(\boldsymbol{x}) = 16x_1(1-x_1)x_2(1-x_2) + 1.1. \tag{61}$$

In particular, we choose to clamp the displacement at $\Gamma_D = \{\boldsymbol{x}\in\Gamma : x_2 = 0\}\cup\{\boldsymbol{x}\in\Gamma : x_1 = 0\}$, whereas the remaining boundary is free.

We build a family of four distorted quadrilateral meshes obtained by randomly distorting square grids of 4×4, 8×8, 16×16, and 32×32 elements. The last refinement is shown in Fig. 2. In the Virtual Element Method, the stabilizing parameter is set to be 2μ.

The behavior of errors (59)–(60) are shown in Fig. 2. We note that the elements in this family of meshes are well represented by the training set. Thus, the network accuracy do not influence the results. Indeed, we observe that the expected order of convergence for both methods is achieved, i.e. the errors decrease as $O(h^2)$ for the L^2-error and as $O(h)$ for the error in the H^1-seminorm, coherently with theoretical results obtained in Section 3.5. Moreover, consistent with the results obtained for the scalar version of the NAVEM method in [21], we observe lower error constants with respect to VEM for both errors.

Finally, we want to remark that the evaluation of a neural network has an intrinsic cost that tends to be negligible for fine meshes, where the cardinality of the input of the neural networks grows, as shown in [21]. In a vector setting, as the one offered by the elasticity problems, this intrinsic cost is even lower because the neural networks are only used to construct the basis functions of $\mathcal{V}_{h,1}^{\mathcal{N}\mathcal{N}}(E)$. When a function in $\boldsymbol{V}_{h,1}^{\mathcal{N}\mathcal{N}}(E)$ has to be evaluated, the functions in $\mathcal{V}_{h,1}^{\mathcal{N}\mathcal{N}}(E)$ associated with each dimension are reused multiple times. This is possible because the vector space is just the Cartesian product of a scalar space, approximated with the neural network, with itself.



(a) (b)

Fig. 3. Test 2: Last refinements related to the two families of meshes.

4.2. Test 2: A benchmark non-linear elastic problem with analytical solution

In this section, we propose a test taken from [9] to show some advantages of using an approach that does not require any stabilization term, such as the NAVEM. Specifically, we consider $\Omega = (0,1)^2$ and set $\Gamma_D = \Gamma$. We select the following constitutive law

$$\boldsymbol{\sigma}(\boldsymbol{x}, \nabla \boldsymbol{u}(\boldsymbol{x})) = \mu(\boldsymbol{\epsilon}(\boldsymbol{u}))\boldsymbol{\epsilon}(\boldsymbol{u}), \tag{62}$$

where the scalar function μ is given by

$$\mu(\boldsymbol{\epsilon}(\boldsymbol{u})) = 3(1 + \|\boldsymbol{\epsilon}(\boldsymbol{u})\|^2) = 3(1 + \sum_{n,m=1}^{2} [\boldsymbol{\epsilon}(\boldsymbol{u})_{nm}]^2). \tag{63}$$

We further consider two external body forces, compatible with the following two analytical solutions:

Case 1: $\boldsymbol{u}(\boldsymbol{x}) = x_1(1-x_1)x_2(1-x_2)\begin{bmatrix}1\\1\end{bmatrix}$;

Case 2: $\boldsymbol{u}(\boldsymbol{x}) = 80x_1(1-x_1)x_2(1-x_2)\begin{bmatrix}1\\1\end{bmatrix}$.

In particular, we notice that in **Case 1** the solution is characterized by deformations of moderate magnitude, whereas much larger deformations occur in **Case 2**. For the current experiment, we consider two families of meshes. The former comprises four square meshes of 4×4, 8×8, 16×16, and 32×32 square elements each. The second family is made up of 4 regular Voronoi meshes subjected to sine distortions realized with the MATLAB library mVEM [40]. The last refinement for each family of meshes is shown in Fig. 3. Preliminarily, we remark that the neural networks used to predict the NAVEM basis functions are the ones used to make the experiment reported in Section 3.3.

Due to non-linearity and possible large deformations, we here decide to test the NAVEM and VEM performance with and without the incremental force method described in Section 2.3, along with the Newton–Raphson scheme. Specifically, we set $\lambda^n = \frac{n}{N}$, $n = 1, \dots, N$, with $N = 20$ for the incremental procedure and, for each $n = 1, \dots, N-1$, we stop the Newton–Raphson scheme after 20 non-linear iteration steps. At the incremental step $n = N$, we allow the Newton–Raphson scheme to perform all the non-linear iteration steps needed to reach the desired accuracy, i.e. we require that both the relative norm of the residual and the norm of the increment of solution between two consecutive steps are below 10^{-10}.

We observe that, when the incremental method is applied, the VEM stabilization is chosen as the norm-based stabilization described in Section 2.3, where $\boldsymbol{w}_h = \boldsymbol{u}_h^{n-1}$. On the other hand, when we apply the plain Newton–Raphson method, without the incremental force approach (i.e. we set $N = 1$), we set $\boldsymbol{w}_h = \boldsymbol{u}_h^0 = \boldsymbol{0}$. In particular, we observe that this last choice coincides with the "fixed scaling" described in [9].

Fig. 4 reports the behavior of the H^1-errors in (59) and (60) as the mesh parameter h decreases, for the test **Case 1**. Concerning this test case, we observe that the expected order of convergence is attained for both methods with and without employing the incremental strategy. Specifically, we note that the choice of stabilization does not seem to influence the accuracy of the VEM method for both families of meshes. Moreover, we observe that the NAVEM error curves are downward shifted with respect to the VEM error curves, as usual. Moreover, we highlight that the L^2-error curves follow a very similar behavior. Thus, we decide to not report this verbose information.

Fig. 5 reports the behavior of the H^1-errors (59)–(60) as the mesh parameter h decreases, for the test **Case 2**, where larger displacements are recorded. Unlike the previous test case, a good choice of the stabilization term is essential to obtain good quality

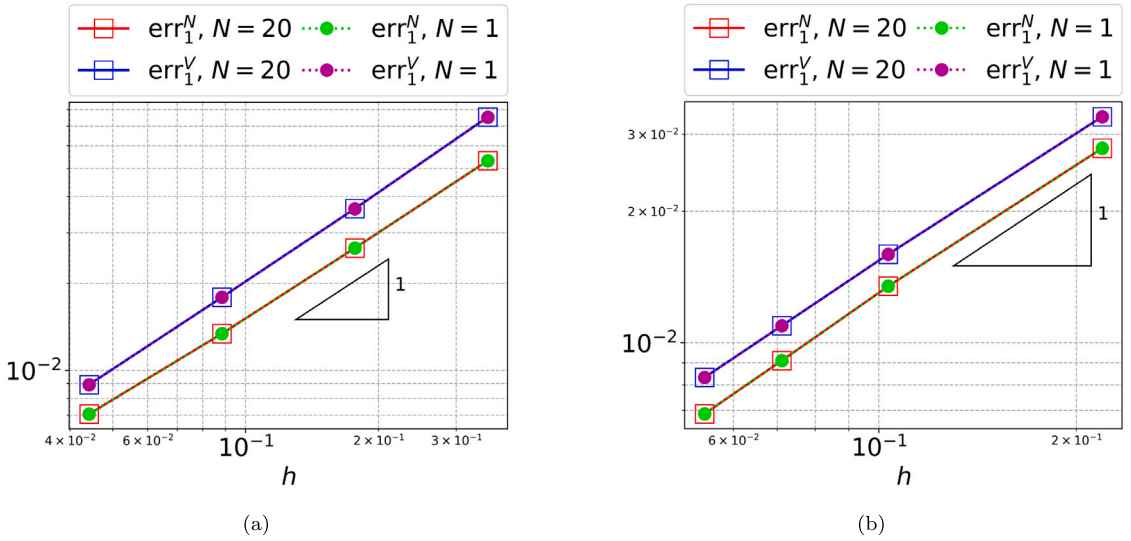


Fig. 4. Test 2, **Case 1**: Behavior of the H^1-errors (59)–(60) as the mesh parameter h decreases. N denotes the number of loading steps for the incremental force method. Left: Cartesian grids family. Right: Distorted-Voronoi grids family.

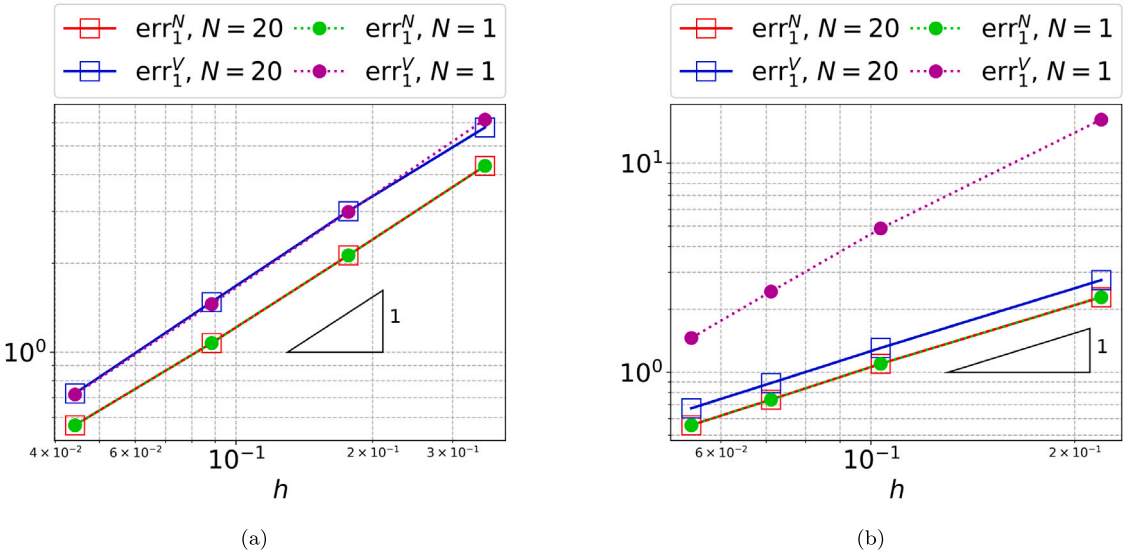


Fig. 5. Test 2, **Case 2**: Behavior of the H^1-errors (59)–(60) as the mesh parameter h decreases. N denotes the number of loading steps for the incremental force method. Left: Cartesian grids family. Right: Distorted-Voronoi grids family.

results when using the VEM method. Coherently with the results shown in [9], we observe that adopting the incremental strategy is mandatory for VEM to avoid to level-off the stabilization term with respect to the consistency term, losing accuracy, when Voronoi meshes are employed. On the other hand, we observe that this behavior is less stressed in the presence of quadrilateral elements, where the stabilization term is generally less pronounced. Concerning the NAVEM method, we observe that it does not need at all the incremental strategy, even in the presence of the larger displacement related to **Case 2**. Indeed, it behaves in the right way independently of the family of meshes and of the magnitude of the displacement for this test, while again showing a lower error constant with respect to VEM (see Figs. 5 and 9).

Moreover, in Fig. 6, we observe that NAVEM does not suffer of the same unstable spurious oscillating behavior, remarked also in [9], that characterizes the VEM solution when a fixed scaling is employed for the **Case 2**. In this figure, we draw a slice of the first component of the VEM solution, of the NAVEM solution, and of the exact solution, for $N = 1$. It is evident that the VEM solution is characterized by oscillations that ruin its accuracy, whereas the other two curves are almost exactly overlapped and are indistinguishable in the plot.

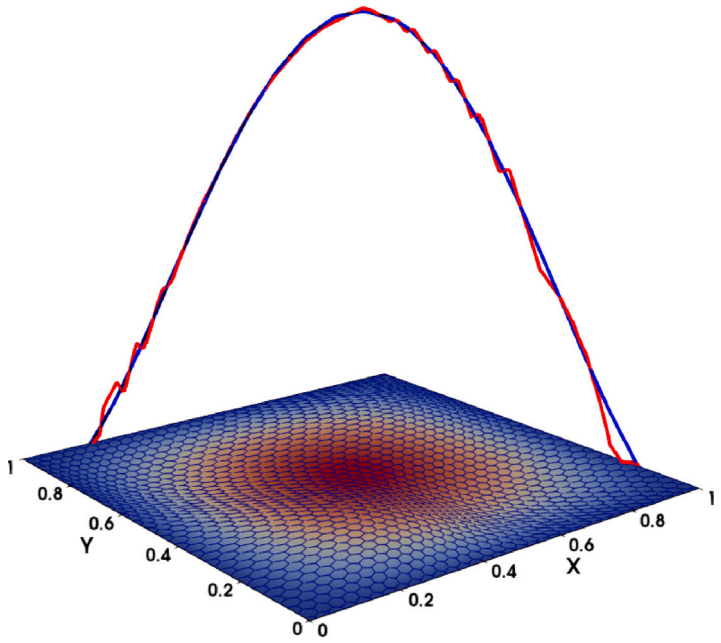


Fig. 6. Test 2, **Case 2**: Slice of the component u_1 related to **Case 2** and the last refinement of the Distorted-Voronoi family. Red: VEM. Blue: NAVEM. Black: exact displacement. We observe that the blue curve overlaps the black one. Solutions obtained with $N = 1$ as the number of loading steps for the incremental force method.

Table 2
Test 2, **Case 2**: Statistics related to each mesh belonging to the Distorted-Voronoi family by employing NAVEM with $N = 1$ and VEM with $N = 20$.

Id. Mesh	r(T)	r(k)	r(ATI)
0	7.85	3.80	2.06
1	7.37	3.73	1.98
2	3.99	3.67	1.09
3	4.32	3.65	1.19

Finally, we observe that the robustness that characterizes NAVEM translates to a higher computational efficiency with respect to VEM. To provide insight into the computational time of NAVEM with respect to VEM, Table 2 reports the time (in seconds) required to solve the non-linear problem **Case 2** on the Distorted-Voronoi family by employing NAVEM with $N = 1$ and VEM with $N = 20$ (i.e. the best scenario for each method). These times refer to a Python code that exploits TensorFlow for neural network operations, running on a Ubuntu 24.04 LTS 64-bit, 12th Gen Intel(R) Core(TM) i7-1255U CPU (4.7 GHz) and 16 GB RAM memory. Specifically, we report

1. **r(T)**: the ratio between the time T required by VEM and by NAVEM to solve problem;
2. **r(k)**: the ratio between the cumulative number of non-linear iteration k over the N loading steps in the VEM and in the NAVEM approach;
3. **r(ATI)**: the ratio between the average time per iteration (ATI in short) in VEM and NAVEM.

We note that the NAVEM time specifically corresponds to the time required to apply the steps **S.1**–**S.5** for solving the problem. Additionally, to exploit the TensorFlow vectorization capabilities and reduce the computational burden, we minimize the number of neural network evaluations by aggregating the inputs associated with multiple basis functions. We observe that the absence of a stabilization term in the NAVEM approach allows the method to save a lot of computational effort with respect to VEM, making NAVEM more convenient to solve non-linear problems.

4.3. Test 3: Neo-Hookean hyperelasticity

In this last test case, we consider Problem (1) with $\boldsymbol{\sigma}$ corresponding to the non-symmetric first Piola–Kirchhoff stress tensor $\mathbf{P}$. More precisely, we focus on a hyperelastic material of Neo-Hookean type by setting

$$\mathbf{P}(\mathbf{F}) = \mu(\mathbf{F} - \mathbf{F}^{-T}) + \lambda J\Theta(J)\Theta'(J)\mathbf{F}^{-T}, \tag{64}$$

Table 3
Test 3: Input parameters for the `arc_length_Crisfield_modified` routine.

Parameter description	Value
Load increment at the first loading step.	0.04
The maximum number of non-linear iterations per increment.	20
The desired number of non-linear iterations per loading step.	4
Tolerance for the convergence criterion.	1.0e−08

where $\mathbf{F}$ is the deformation gradient $\mathbf{F} = \mathbf{I} + \nabla \boldsymbol{u}$, $J \in \mathbb{R}_{>0}$ is the determinant of $\mathbf{F}$, and $\Theta : \mathbb{R}_{>0} \to \mathbb{R}$ is a smooth function such that $\Theta(J) = 0 \Leftrightarrow J = 1$ and $\Theta'(1) \neq 0$ [52]. In Eqs. (64), λ and μ are given positive constants.

We observe that, to carry out the discretization related to a Neo-Hookean material, a discrete version of the determinant J must be provided. In the Virtual Element framework, two possible discretization are considered. The simplest choice coincides with taking on each element $E \in \mathcal{T}_h$:

$$J = \det(\mathbf{I} + \nabla \boldsymbol{u}_h^V) \approx \det(\mathbf{I} + \Pi_0^{0,E} \nabla \boldsymbol{u}_h^V).$$

Again, since the virtual functions are not known in a closed form in the interior of the element, a projector operator should be introduced to access the point-wise evaluations of such functions. With this kind of approximation, the determinant J is constant on each element $E \in \mathcal{T}_h$. A more stable version is proposed in [13], where J is approximated with its mean value over $E \in \mathcal{T}_h$. More precisely, the authors in [13] approximate J as follows:

$$J \approx \frac{1}{|E|} \int_E J = \int_{\hat{E}} d\hat{\boldsymbol{x}} = \frac{|\hat{E}|}{|E|}, \tag{65}$$

where $|E|$ denotes the measure of the element $E \in \mathcal{T}_h$, whereas $\hat{\cdot}$ defines a quantity $\cdot$ in the current deformed configuration. In the NAVEM framework, we evaluate the determinant simply as

$$J = \det(\mathbf{I} + \widetilde{\nabla} \boldsymbol{u}_h^{\mathcal{N}\mathcal{N}}), \tag{66}$$

since we are able to evaluate NAVEM functions and their gradients everywhere.

We test the method considering

$$\Theta(J) = \log J, \quad \Theta'(J) = \frac{1}{J}, \quad \lambda = 5.1, \quad \mu = 1.0,$$

whose values of Lamé coefficients correspond to a Poisson ratio ≈ 0.45. We observe that in this case, the elastic modulus $\mathbb{A}$ in (11) in this specific case depends only on the deformation gradient and it is given by [52]:

$$\begin{aligned}\mathbb{A}(\mathbf{F}) = \frac{\partial \mathbf{P}(\mathbf{F})}{\partial \mathbf{F}} &= \mu(\mathbf{I} \overline{\otimes} \mathbf{I} + \mathbf{F}^{-T} \underline{\otimes} \mathbf{F}^{-1}) - \lambda J \Theta(J) \Theta'(J) \mathbf{F}^{-T} \underline{\otimes} \mathbf{F}^{-1} \\ &\quad + \lambda J \left[\Theta(J)\Theta'(J) + J(\Theta'(J))^2 + J\Theta(J)\Theta''(J) \right] \mathbf{F}^{-T} \otimes \mathbf{F}^{-T},\end{aligned}$$

where $\otimes$, $\underline{\otimes}$ and $\overline{\otimes}$ are defined such that $\{\mathbf{A} \otimes \mathbf{B}\}_{ijkl} = \mathbf{A}_{ij}\mathbf{B}_{kl}$, $\{\mathbf{A} \underline{\otimes} \mathbf{B}\}_{ijkl} = \mathbf{A}_{il}\mathbf{B}_{jk}$ and $\{\mathbf{A} \overline{\otimes} \mathbf{B}\}_{ijkl} = \mathbf{A}_{ik}\mathbf{B}_{jl}$, for all $i, j, k, l = 1, \dots, 2$ and for each pair of second-order tensors $\mathbf{A}$ and $\mathbf{B}$.

The initial configuration Ω is set as the unit square $[0, 1]^2$ and the displacement is clamped at $\Gamma_D = \{0\} \times [0, 1]$, while the remaining part of the boundary is free. The body load is given by

$$\boldsymbol{f}(x_1, x_2) = [100x_2^3, 0] \quad \forall (x_1, x_2) \in \Omega. \tag{67}$$

We simulate this test on the mesh shown in Fig. 8(a), generated as in [53], by exploiting the arc-length method. The employed code strictly follows the MATLAB routine arc_length_Crisfield_modified. Specifically, this routine implements Crisfield algorithm proposed in [31,54], in which the load parameter is introduced as the additional unknown satisfying the following $(N^{\text{dof}} + 1)$-hypersphere constraint, that is enforced at each non-linear iteration within every loading step:

$$(\boldsymbol{u}_*)^T(\boldsymbol{u}_*) + (\lambda_*)^2 = (\Delta S_*)^2, \quad \forall * \in \{V, N\},$$

where ΔS_* denotes the prescribed increment of the arc-length parameter along the load–displacement path, which may vary across different loading steps, and V and N denote the VEM and NAVEM variables, respectively. We observe that this constraint yields two possible values of λ at each non-linear step. In the implemented algorithm, the value of λ closest to 1 is selected. We further modify the MATLAB routine by imposing that whenever $|\lambda - 1.0| < 0.1$, a further final loading step must be performed by keeping fixed $\lambda = 1$. Fig. 7 reports the values for the converged λ_*, with $* \in \{V, N\}$, as the loading step n increases on the left axis, whereas the cumulative number of non-linear iterations over the loading steps is shown on the right axis. Moreover, Figs. 8(b) and 8(c) show the x_2-component of the final discrete displacement obtained with NAVEM and VEM, respectively, whereas Fig. 9(b) shows the result of the Finite Element discretization of the problem on the finer triangular mesh 9(a). These results are obtained by choosing as input parameter of arc_length_Crisfield_modified routine the values detailed in Table 3.

We first note that the NAVEM and VEM approaches employ approximately the same total number of iterations. Nevertheless, we can observe in Fig. 8 some spurious oscillations in the x_2-component of the virtual element displacement that are not present

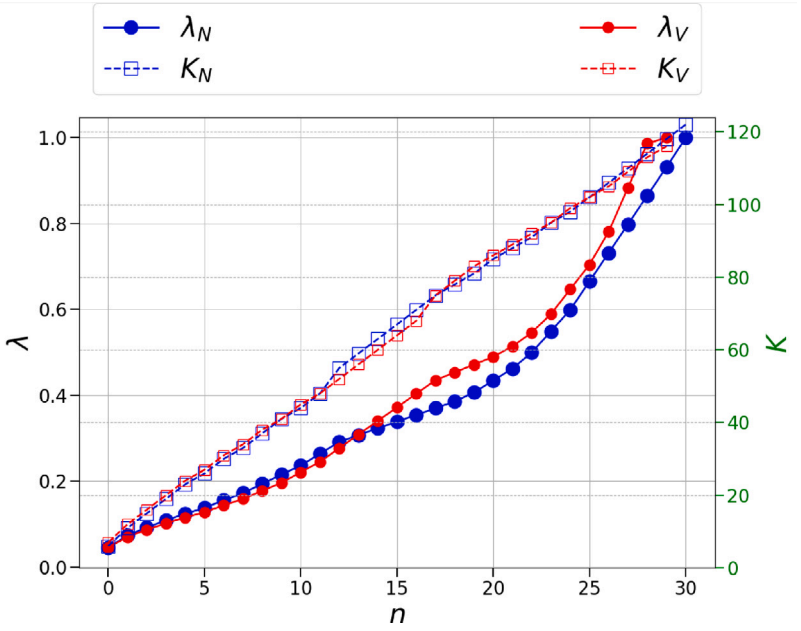


Fig. 7. Test 3: Left axis: Converged λ values as the loading step n increases. Right axis: Cumulative sum of non-linear iteration steps as the loading step n increases.

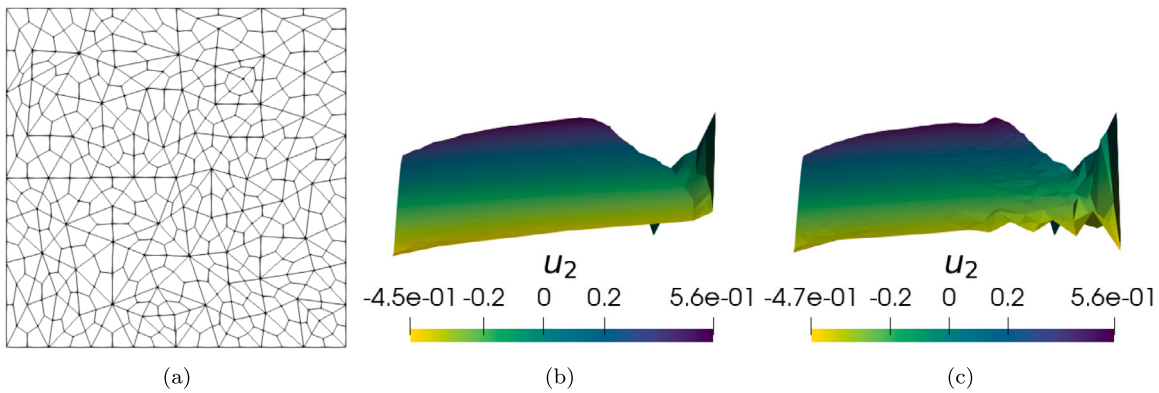


Fig. 8. Test 3: Left: Mesh used in the simulation. Center: Displacement in the x_2-direction obtained with the NAVEM method. Right: Displacement in the x_2-direction obtained with the VEM method.

in the NAVEM and FEM solutions. This further confirms that the NAVEM approach appears to be more robust in the presence of strong non-linearities.

Finally, we observe that we do not appreciate a significant difference between the two approximation procedures for computing the determinant in the virtual element procedure in this test case.

To conclude, we observe that the NAVEM has a simpler formulation than the VEM one and, in the provided numerical tests, it is more stable in the presence of large deformations. On the other hand, despite having a formulation similar to the FEM one, it ensures greater flexibility by allowing the use of more general meshes.

5. Conclusions

In this paper, we present the Neural Approximated Virtual Element Method to solve linear and non-linear elasticity problems. The NAVEM is a polygonal method that approximates via neural networks the function spaces used in the Virtual Element Method. This neural approximation provides closed-form local basis functions, eliminating the need for additional stabilization and projection operators. An a priori error analysis is also presented to strengthen the theoretical foundations of the method.

The absence of a stabilization term is particularly advantageous in non-linear problems, where designing a suitable and computable stabilization operator that preserves accuracy can be challenging. Elasticity problems represent a particular domain

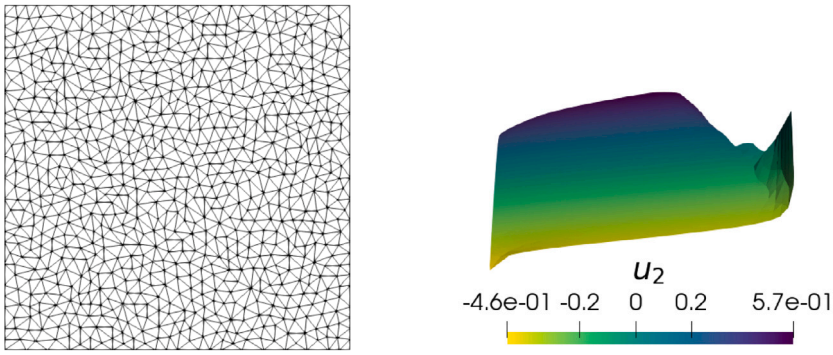


Fig. 9. Test 3: Left: Mesh used to compute the FEM reference solution. Right: Displacement in the x_2-direction obtained with the FEM method.

in which numerous different non-linearities may be present, depending on the simulated materials. For this class of problems, we discuss the NAVEM formulation, emphasizing the simpler form with respect to the standard VEM formulation. Numerical results are in agreement with what has been observed for elliptic problems in [20,21] and highlight the benefits of bypassing stabilization operators.

Future perspectives include extending NAVEM to more physically relevant problems, such as handling internal constraints like incompressibility. Additionally, crack propagation and adaptivity present promising applications for this method. Indeed, as observed in [21], the method can be easily optimized for a mesh comprising triangles with hanging nodes. This feature opens the possibility of integrating NAVEM with standard finite element methods by refining only elements where non-linearities are most intense: introducing hanging nodes instead of globally remeshing neighboring elements.

CRediT authorship contribution statement

Stefano Berrone: Writing – review & editing, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Moreno Pintore:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Gioana Teora:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The author S.B. kindly acknowledges partial financial support provided by PRIN project “Advanced polyhedral discretizations of heterogeneous PDEs for multiphysics problems” (No. 20204LN5N5_003), by PNRR M4C2 project of CN00000013 National Centre for HPC, Big Data and Quantum Computing (HPC) (CUP: E13C22000990001) and the funding by the European Union through project Next Generation EU, M4C2, PRIN 2022 PNRR project P2022BH5CB_001 “Polyhedral Galerkin methods for engineering applications to improve disaster risk forecast and management: stabilization-free operator-preserving methods and optimal stabilization methods”. The author M.P. kindly acknowledges financial support provided by PEPR/IA (https://www.pepr-ia.fr/). The author G.T. kindly acknowledges the financial support provided by INdAM-GNCS Project “Metodi numerici efficienti per problemi accoppiati in sistemi complessi” (CUP: E53C24001950001) and by the PRIN project 20227K44ME “Full and Reduced order modeling of coupled systems: focus on non-matching methods and automatic learning (FaReX)”.

Data availability

Data will be made available on request.

References

[1] S. Berrone, S. Ferraris, D. Grappein, G. Teora, F. Vicini, A 3D-1d virtual element method for modeling root water uptake, 2024, arXiv:2412.12884. URL https://arxiv.org/abs/2412.12884.

[2] S. Berrone, F. Vicini, Effective polygonal mesh generation and refinement for VEM, Math. Comput. Simulation 231 (2025) 239–258, http://dx.doi.org/10.1016/j.matcom.2024.12.007.

[3] S.E. Leon, D.W. Spring, G.H. Paulino, Reduction in mesh bias for dynamic fracture using adaptive splitting of polygonal finite elements, Internat. J. Numer. Methods Engrg. 100 (8) (2014) 555–576, http://dx.doi.org/10.1002/nme.4744.

[4] S.O.R. Biabanaki, A.R. Khoei, P. Wriggers, Polygonal finite element methods for contact-impact problems on non-conformal meshes, Comput. Methods Appl. Mech. Engrg. 269 (2014) 198–221, http://dx.doi.org/10.1016/j.cma.2013.10.025.

[5] A.L. Gain, G.H. Paulino, L.S. Duarte, I.F. Menezes, Topology optimization using polytopes, Comput. Methods Appl. Mech. Engrg. 293 (2015) 411–430, http://dx.doi.org/10.1016/j.cma.2015.05.007.

[6] L. Beirão da Veiga, F. Brezzi, A. Cangiani, G. Manzini, A. Russo, Basic principles of virtual element methods, Math. Models Methods Appl. Sci. 23 (01) (2013) 199–214, http://dx.doi.org/10.1142/S0218202512500492.

[7] L. Beirão da Veiga, F. Brezzi, L.D. Marini, A. Russo, Virtual element method for general second order elliptic problems on polygonal meshes, Math. Models Methods Appl. Sci. 26 (04) (2016) 729–750, http://dx.doi.org/10.1142/S0218202516500160.

[8] L. Beirão da Veiga, F. Brezzi, L.D. Marini, Virtual elements for linear elasticity problems, SIAM J. Numer. Anal. 51 (2) (2013) 794–812, http://dx.doi.org/10.1137/120874746.

[9] L. Beirão da Veiga, C. Lovadina, D. Mora, A virtual element method for elastic and inelastic problems on polytope meshes, Comput. Methods Appl. Mech. Engrg. 295 (2015) 327–346, http://dx.doi.org/10.1016/j.cma.2015.07.013.

[10] S.C. Brenner, L.R. Scott, The Mathematical Theory of Finite Element Methods, Texts in Applied Mathematics, vol. 15, Springer, 2008, http://dx.doi.org/10.1007/978-0-387-75934-0.

[11] P.G. Ciarlet, The finite element method for elliptic problems, Classics in Applied Mathematics, vol. 40, Society for Industrial and Applied Mathematics (SIAM), Philadephia, 2002, http://dx.doi.org/10.1137/1.9780898719208.

[12] S. Berrone, S. Scialò, G. Teora, The mixed virtual element discretization for highly-anisotropic problems: the role of the boundary degrees of freedom, Math. Eng. 5 (6) (2023) 1–32, http://dx.doi.org/10.3934/mine.2023099.

[13] H. Chi, L.B. da Veiga, G. Paulino, Some basic formulations of the virtual element method (VEM) for finite deformations, Comput. Methods Appl. Mech. Engrg. 318 (2017) 148–192, http://dx.doi.org/10.1016/j.cma.2016.12.020.

[14] A. Chen, N. Sukumar, Stabilization-free serendipity virtual element method for plane elasticity, Comput. Methods Appl. Mech. Engrg. 404 (2023) 115784, http://dx.doi.org/10.1016/j.cma.2022.115784.

[15] A. Chen, N. Sukumar, Stabilization-free virtual element method for plane elasticity, Comput. Math. Appl. 138 (2023) 88–105, http://dx.doi.org/10.1016/j.camwa.2023.03.002.

[16] F. Credali, S. Bertoluzza, D. Prada, Reduced basis stabilization and post-processing for the virtual element method, Comput. Methods Appl. Mech. Engrg. 420 (2024) 116693, http://dx.doi.org/10.1016/j.cma.2023.116693.

[17] M. Trezzi, U. Zerbinati, When rational functions meet virtual elements: the lightning virtual element method, Calcolo 61 (3) (2024) 35, http://dx.doi.org/10.1007/s10092-024-00585-1.

[18] J.S. Hesthaven, G. Rozza, B. Stamm, et al., Certified Reduced Basis Methods for Parametrized Partial Differential Equations, SpringerBriefs in Mathematics, vol. 590, Springer, 2016, http://dx.doi.org/10.1007/978-3-319-22470-1.

[19] A. Gopal, L.N. Trefethen, Solving Laplace problems with corner singularities via rational functions, SIAM J. Numer. Anal. 57 (5) (2019) 2074–2094, http://dx.doi.org/10.1137/19M125947X.

[20] S. Berrone, D. Oberto, M. Pintore, G. Teora, The lowest-order neural approximated virtual element method, in: A. Sequeira, A. Silvestre, S.S. Valtchev, J.a. Janela (Eds.), Numerical Mathematics and Advanced Applications ENUMATH 2023, Volume 1, Springer Nature Switzerland, Cham, 2025, pp. 129–138, http://dx.doi.org/10.1007/978-3-031-86173-4_13.

[21] S. Berrone, M. Pintore, G. Teora, The lowest-order neural approximated virtual element method on polygonal elements, Comput. Struct. 314 (2025) 107753, http://dx.doi.org/10.1016/j.compstruc.2025.107753.

[22] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[23] S. Cuomo, V.S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics–Informed neural networks: Where we are and what's next, J. Sci. Comput. 92 (3) (2022) 88, http://dx.doi.org/10.1007/s10915-022-01939-z.

[24] M. Abadi, et al., TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, Software available from tensorflow.org. URL http://tensorflow.org/.

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An imperative style, high-performance deep learning library, in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., 2019, pp. 8024–8035, http://dx.doi.org/10.5555/3454287.3455008.

[26] J. Bradbury, R. Frostig, P. Hawkins, M.J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of python+numpy programs, 2018, http://github.com/google/jax.

[27] L. Beirão da Veiga, F. Brezzi, L.D. Marini, A. Russo, The virtual element method, Acta Numer. 32 (2023) 123–202, http://dx.doi.org/10.1017/S0962492922000095.

[28] P. Wriggers, Nonlinear Finite Element Methods, Springer Berlin Heidelberg, 2008, http://dx.doi.org/10.1142/11797.

[29] J. Bonet, A. Gil, R. Wood, Nonlinear Solid Mechanics for Finite Element Analysis:Statics, Cambridge University Press, 2016, http://dx.doi.org/10.1017/9781316336144.

[30] N. Kim, Introduction to Nonlinear Finite Element Analysis, Springer New York, NY, 2014, http://dx.doi.org/10.1007/978-1-4419-1746-1.

[31] M. Fafard, B. Massicotte, Geometrical interpretation of the arc-length method, Comput. Struct. 46 (4) (1993) 603–615, http://dx.doi.org/10.1016/0045-7949(93)90389-U.

[32] P. Wriggers, F. Aldakheel, B. Hudobivnik, Virtual Element Methods in Engineering Sciences, Springer International Publishing, 2023, http://dx.doi.org/10.1007/978-3-031-39255-9.

[33] S. Berrone, G. Teora, F. Vicini, Improving high-order VEM stability on badly-shaped elements, Math. Comput. Simulation 216 (2024) 367–385, http://dx.doi.org/10.1016/j.matcom.2023.10.003.

[34] J.B. Perot, C. Chartrand, A mimetic method for polygons, J. Comput. Phys. 424 (2021) 109853, http://dx.doi.org/10.1016/j.jcp.2020.109853.

[35] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep Learning, vol. 1, MIT press Cambridge, 2016.

[36] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, Vol. 9, PMLR, 2010, pp. 249–256.

[37] D. Kingma, J. Ba, Adam: a method for stochastic optimization, Int. Conf. Learn. Represent. (2014) http://dx.doi.org/10.48550/arXiv.1412.6980.

[38] S. Wright, J. Nocedal, et al., Numerical Optimization, vol. 35, Springer, 1999, p. 7.

[39] E. Kiyani, K. Shukla, J.F. Urbán, J. Darbon, G.E. Karniadakis, Optimizing the optimizer for physics-informed neural networks and Kolmogorov-Arnold networks, Comput. Methods Appl. Mech. Engrg. 446 (2025) 118308, http://dx.doi.org/10.1016/j.cma.2025.118308.
[40] Y. Yu, mVEM: A MATLAB software package for the virtual element methods, 2022, http://dx.doi.org/10.48550/arXiv.2204.01339, arXiv:2204.01339.
[41] A. Sommariva, M. Vianello, Gauss–Green cubature and moment computation over arbitrary geometries, J. Comput. Appl. Math. 231 (2) (2009) 886–896, http://dx.doi.org/10.1016/j.cam.2009.05.014.
[42] A. Sommariva, M. Vianello, Compression of multivariate discrete measures and applications, Numer. Funct. Anal. Optim. 36 (9) (2015) 1198–1223, http://dx.doi.org/10.1080/01630563.2015.1062394.
[43] S. Berrone, A. Borio, G. Teora, F. Vicini, POLYDIM: A C++ library for polytopal discretization methods, 2025, http://dx.doi.org/10.48550/arXiv.2505.14063, ArXiv.
[44] F. Dassi, Vem++, a C++ library to handle and play with the virtual element method, Numer. Algorithms (2025) http://dx.doi.org/10.1007/s11075-025-02059-z.
[45] A. Bonito, C. Canuto, R.H. Nochetto, A. Veeser, Adaptive finite element methods, Acta Numer. 33 (2024) 163–485, http://dx.doi.org/10.1017/S0962492924000011.
[46] L. Beirão da Veiga, L. Mascotto, Interpolation and stability properties of low-order face and edge virtual element spaces, IMA J. Numer. Anal. 43 (2) (2022) 828–851, http://dx.doi.org/10.1093/imanum/drac008.
[47] H. Ammari, H. Kang, Reconstruction of Small Inhomogeneities from Boundary Measurements, Lecture Notes in Mathematics, vol. 1846, Springer Berlin, Heidelberg, 2004, http://dx.doi.org/10.1007/b98245.
[48] S. Berrone, C. Canuto, M. Pintore, Solving PDEs by variational physics-informed neural networks: an a posteriori error analysis, Ann. Dell'Univ. Ferrara 68 (2022) 575–595, http://dx.doi.org/10.1007/s11565-022-00441-6.
[49] P. Ciarlet, Basic error estimates for elliptic problems, in: Finite Element Methods (Part 1), in: Handbook of Numerical Analysis, vol. 2, Elsevier, 1991, pp. 17–351, http://dx.doi.org/10.1016/S1570-8659(05)80039-0.
[50] D. Boffi, F. Brezzi, M. Fortin, Mixed Finite Element Methods and Applications, Springer Series in Computational Mathematics, Springer Berlin Heidelberg, 2013, http://dx.doi.org/10.1007/978-3-642-36519-5.
[51] B. Ahmad, A. Alsaedi, F. Brezzi, L. Marini, A. Russo, Equivalent projectors for virtual element methods, Comput. Math. Appl. 66 (3) (2013) 376–391, http://dx.doi.org/10.1016/j.camwa.2013.05.015.
[52] M. Abbas, A. Ern, N. Pignet, Hybrid high-order methods for finite deformations of hyperelastic materials, Comput. Mech. 62 (4) (2018) 909–928, http://dx.doi.org/10.1007/s00466-018-1538-0.
[53] A. Lamperti, M. Cremonesi, U. Perego, A. Russo, C. Lovadina, A Hu–Washizu variational approach to self-stabilized virtual elements: 2D linear elastostatics, Comput. Mech. 71 (2023) 935–955, http://dx.doi.org/10.1007/s00466-023-02282-2.
[54] M.A. Crisfield, A fast incremental/iterative solution procedure that handles "snap-through", Comput. Struct. 13 (1) (1981) 55–62, http://dx.doi.org/10.1016/0045-7949(81)90108-5.