An Arbitrary-Order Virtual Element Method for the Helmholtz Equation Applied to Wave Field Calculation in Port

Ronan Dupont*, Mathias Dauphin†, Romain Mottier‡

Abstract

The Virtual Element Method (VEM), as a high-order polytopal method, offers significant advantages over traditional Finite Element Methods (FEM). In particular, it allows the handling of polytopal or non-conforming meshes which greatly simplificates the mesh generation procedure. In this paper, the VEM is used for the discretization of the Helmholtz equations with a Robin-type absorbing boundary condition. This problem is crucial in various fields, including coastal engineering, oceanography and the design of offshore structures. Details of the VEM implementation with Robin boundary condition are given. Numerical results on test cases with analytical solutions show that the methods can provide optimal convergence rates for smooth solutions. Then, as a more realistic test case, the computation of the eigenmodes of the port of Cherbourg is carried out.

Modelling Subjects Classification.

Keywords. Virtual Element Method, Polytopal Methods, Robin Boundary Condition, Helmholtz Equation, Wave Propagation, Mild-Slope Equation, Coastal Engineering.

Table of content

*GEOSCIENCES-M, Univ Montpellier, CNRS, Montpellier, France and IMAG, Univ Montpellier, CNRS, Montpellier, France. Email adress: ronan.dupont@umontpellier.fr

†Scuola Superiore Meridionale, Napoli, Italy. Email adress: mathias.dauphin-ssm@unina.it

‡CEA, DAM, DIF, Arpajon, France, CERMICS, Ecole des Ponts, Marne la Vallée cedex 2, and SERENA Project-Team, INRIA Paris, Paris France. Email adress: romain.mottier@enpc.fr

1 Introduction

Nowadays, coastal modelling has become a major challenge in the face of climate change. The study of coastal regions encompasses a wide array of topics, including (large-scale) ocean modelling, harbor modelling and many other subjects such as morphodynamics. In this study, we are particularly interested in port modelling using the Helmholtz equation [21]. The Helmholtz equation is a well-established model, widely applicable in different fields such as electromagnetics and acoustics. Traditionally, in the context of the modelling of port wave fields [10], the Helmholtz model is employed for flat seabeds. More elaborate models such as the mild-slope equation [6] can be utilized for seabeds with varying topography, although it is typically constrained to slopes not exceeding $1/3$ [7].
The Virtual Element Method (VEM) is a brand-new numerical method recently introduced in [8]. It can be seen as an extension of the classical Finite Element Method (FEM), however, it was historically developed from the mimetic finite difference (MFD) method [11]. In fact, the real advantage of the VEM lies in the fact that it allies the great generality allowed by the MFD method for the geometry of the elements and the classical discretization path of the FEM. It allows the natural handling of polyhedral and non-conforming meshes while retaining the H^1-conformity [5]. For example, with such a property, mesh refinement procedure, which often introduces hanging nodes, can easily be set up.
The paper is organized as follows. First, we introduce the physical context with the hydrodynamic and the model problem we studied in this paper (in the section 2). Then, we devised the virtual element discretization of these problems (in the section 3). A guide to the implementation of VEM and Robin's boundary conditions will then be presented in section 4. Finally, in the section 5, validation results and an application example will be given on a concrete example from the port of Cherbourg.

2 Physical Context

In this section, we will present the physical and mathematical modelling of wave reflection. First, we will derive the mild-slope and Helmholtz equations, which will capture wave behavior over variable and constant seabed depths, respectively. Then, we will present the Helmholtz problem with mixed boundary conditions, enabling the analysis of wave interactions within confined domains.

2.1 Hydrodynamics for Wave Reflection

We are interested in deriving a model equation to study wave reflection in ports. Let us consider an inviscid fluid with constant density evolving in a 3D canal ending with a wall. As depicted in Fig. 1, the z-axis points upward with origin $z = 0$ set at the mean water level. The sea bottom is defined below the origin by $z = -h(x, y)$, where h is the water column height that does not vary in time. We then assume that the sea bottom is not susceptible to accretion or erosion. Finally, the free-surface elevation is defined above the origin by $z = \zeta(x, y, t)$.
The flow is assumed to be incompressible and irrotational. Hence, as it is well-known (see e.g. [17]), there exists a velocity potential $\Phi(x, y, z, t)$ that satisfies the Laplace equation:

$$\Delta\Phi = 0.$$

Moreover, assuming a no-slip condition $\boldsymbol{\nabla}\Phi \cdot \boldsymbol{n} = 0$ at the bottom $z = -h$, where $\boldsymbol{n}$ is the outward normal to the seabed, and restricting the study of the free-surface elevation at $z = 0$ to the context of Airy's linear approximation wave theory [3], one can express Φ as a free-surface potential ϕ in the (x, y)-plane scaled by a certain function:

$$\Phi(x, y, z, t) = f(z, h)\phi(x, y, t)$$

where, according to [7], the scaling function f reads

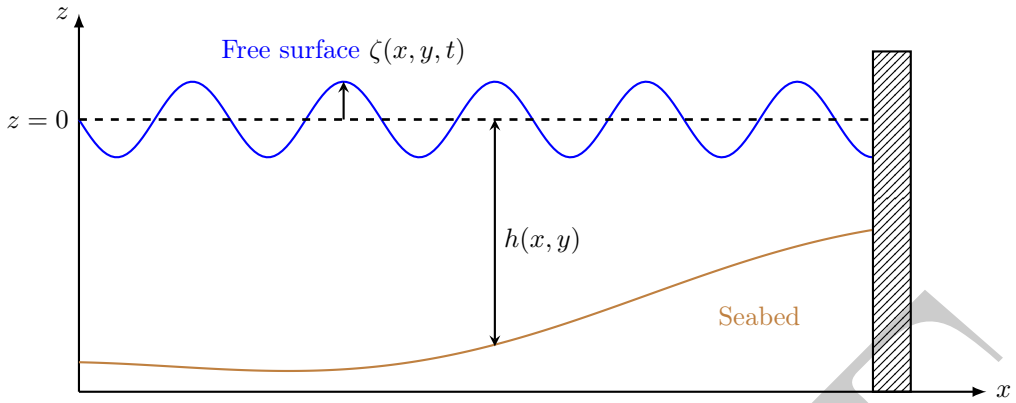


Fig. 1: Sketch of a free surface elevation ζ in the (x, z)-plane.

$$f(z, h) = \frac{\cosh(\kappa(z + h)}{\cosh(\kappa h)}$$

with $\kappa(x, y)$ being the wavenumber which can be retrieved from the linear dispersion relation

$$\omega = g\kappa \tanh(\kappa h)$$

with g the Earth's gravity and $\omega = 2\pi/T$ the constant angular frequency associated with the wave period T. Following the same principle as in [23], that is exploiting the Lagrangian formulation, we obtain the following set of equations

$$\begin{aligned} &g\frac{\partial \zeta}{\partial t} + \boldsymbol{\nabla} \cdot (C_p C_g \boldsymbol{\nabla} \phi) + (\kappa^2 C_p C_g - \omega^2)\phi = 0 \\ &\frac{\partial \phi}{\partial t} + g\zeta = 0 \end{aligned}$$

which can be restated to eliminate the free-surface potential ϕ and thus having the time-dependent mild-slope equation for the free-surface elevation

$$-\frac{\partial^2 \zeta}{\partial t^2} + \boldsymbol{\nabla} \cdot (C_p C_g \boldsymbol{\nabla} \zeta) + (\kappa^2 C_p C_g - \omega^2)\zeta = 0. \tag{1}$$

From here, we can derive the mild-slope model and then the Helmholtz model for simulating wave reflection in a given port geometry.

Mild-slope equation. The mild-slope equation allows us to describe the propagation of the reflected wave above a certain depth $z = -h(x, y)$. To derive it from Eq. (1), we apply the same principle as in [10], namely, we decompose the free-surface elevation into an incident and a reflected part

$$\zeta = \zeta_R + \zeta_I$$

where both parts can be split into their real-valued amplitude and phase

$$\zeta_R(x, y) = u(x, y)e^{-i\omega t} \quad \text{and} \quad \zeta_I(x, y) = v(x, y)e^{-i\omega t}$$

where the incident part is defined through θ, the incident wave angle, and $v_{\max}$, the maximum wave amplitude

$$v(x, y) = v_{\max} e^{-i\boldsymbol{\kappa}\cdot\mathbf{x}} \quad \text{with} \quad \boldsymbol{\kappa} = \kappa(\cos(\theta), \sin(\theta))^\intercal.$$

By injecting the expression of ζ_R in Eq. (1), one can find the steady mild-slope equation for the reflected amplitude

$$\boldsymbol{\nabla} \cdot (C_p C_g \boldsymbol{\nabla} u) + \kappa^2 C_p C_g u = 0. \tag{2}$$

Helmholtz Equation. The Helmholtz equation constitutes a weaker model than the mild-slope equation in the sense that it relies on the hypothesis of constant depth h. Moreover, assuming $C_g = C_p/2$ (as in shallow water approximation) and noting that $C_p = \omega/\kappa$ is constant, one can rewrite the mild-slope equation Eq. (2) as the Helmholtz equation [21]:

$$\Delta u + \kappa^2 u = 0. \tag{3}$$

2.2 Model Problem

Let us consider a polygonal domain $\Omega \subset \mathbb{R}^2$ whose boundary is partionned by mutually disjoint subsets such that $\partial\Omega = \overline{\Gamma_D} \cup \overline{\Gamma_N} \cup \overline{\Gamma_R}$, with $|\Gamma_R| > 0$. The model problem is therefore the following boundary value problem composed of the Helmholtz equation Eq. (3) together with mixed boundary conditions:

$$\begin{cases} \text{Find } u \in H^1(\Omega) \text{ such that} & \\ \Delta u + \kappa^2\, u = f, & \text{in } \Omega \\ u = g_D, & \text{in } \Gamma_D\,, \\ \dfrac{\partial u}{\partial n} = g_N, & \text{in } \Gamma_N\,, \\ \dfrac{\partial u}{\partial n} + i\kappa\, u = g_R, & \text{in } \Gamma_R\,. \end{cases} \tag{4}$$

where $f \in H^{-1}(\Omega)$ is the load term and $g_D \in H^{\frac{1}{2}}(\Gamma_D)$, $g_N \in H^{-\frac{1}{2}}(\Gamma_N)$and $g_R \in H^{-\frac{1}{2}}(\Gamma_R)$ are the functions imposed at the corresponding borders. According to [24], this problem is well-posed under the assumptions made on the domain and its boundary.
Let us define the following spaces:

$$V_0 \coloneqq \{v \in H^1(\Omega) \colon v|_{\Gamma_D} = 0\}$$
$$V_D \coloneqq \{v \in H^1(\Omega) \colon v|_{\Gamma_D} = g_D\}$$

and let $b \colon V_D \times V_0 \to \mathbb{R}$, $m \colon V_D \times V_0 \to \mathbb{R}$ and $r \colon V_D \times V_0 \to \mathbb{R}$ three bilinear forms and $l \colon V_0 \to \mathbb{R}$ a linear form defined as follows

$$b(u,v) = -\int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v$$
$$m(u,v) = \kappa^2 \int_\Omega uv$$
$$r(u,v) = -i\kappa \int_{\Gamma_R} uv$$
$$l(v) = \int_\Omega fv - \int_{\Gamma_N} g_N v - \int_{\Gamma_R} g_R v.$$

The weak formulation of Eq. (4) then writes

$$\begin{cases} \text{Find } u \in V_D \text{ such that} \\ \qquad a(u,v) = l(v), \quad \forall v \in V_0 \end{cases} \tag{5}$$

where for all $u, v \in V_D \times V_0$, $a(u,v) = b(u,v) + m(u,v) + r(u,v)$

3 The Virtual Element Method

In this section, we expose the building steps of the VEM. As mentioned before, the method is similar to the FEM in that it follows the same general construction—in fact, one can easily write a virtual element code starting from a finite element code since the biggest difference lies in the computation of local matrices. The VEM can be synthesized and implemented following four main steps divided into building blocks as depicted in Fig. 2.

Once the domain Ω has been decomposed by a polygonal mesh Ω_h, the first step consists of building the Ciarlet triplet $(T, V_h(T), \Sigma_T)$ where T is an element of the mesh Ω_h, $V_h(T)$ is the local virtual space defined on T and Σ_T is the local set of degrees of freedom attached to T.

Based on this three-block foundation, one can move up to the second block by constructing the global virtual element space V_h by continuously glueing the local virtual spaces over all mesh elements. However, the functions contained in the local virtual spaces are not known explicitly as in finite element but are defined implicitly in such a way that they solve a local PDE on each mesh element—which is why they are referred to as "virtual". Therefore, in order to render those functions computable, one needs to define a projection operator Π onto a polynomial space, such that it can be computed from the degrees of freedom. Thanks to the projection operator Π, the global space can be split into a polynomial part ΠV_h and a non-polynomial part $(I - \Pi)V_h$. In that regard, the virtual space is richer than the finite element space but remains a conforming approximation in the sense that $V_h \subset V_D$.

Now that an approximation space has been defined, one can move to the third block by specifying the discrete variational problem where the discrete bilinear form a_h is composed of two parts inherited from the virtual space projection decomposition: a consistency part computed from Π and a stability part computed from $I - \Pi$.

Finally, the last step corresponds to solving the linear system which is strictly equivalent to the discrete variational problem, it is simply restated in algebraic form.

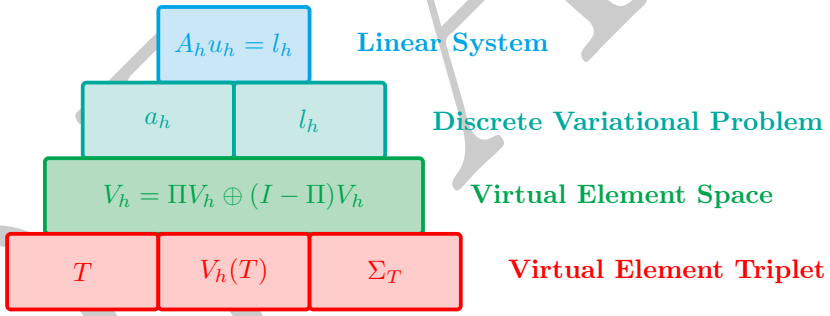


Fig. 2: Fundamental building blocks for the Virtual Element Method

3.1 Preliminaries

Mesh decomposition. As in FEM, the first step for building the method is the discretization of the computational domain Ω. But, contrary to the FEM, the VEM can handle polygonal meshes made out of a wide variety of element shapes—that is, in the 2D case, not only simplices or quadrangles, but also general polygons. This geometrical flexibility allows us to consider non-conforming elements with hanging nodes as well as non-convex elements.

We therefore consider a polygonal decomposition $(\Omega_h)_h$ of the computational domain Ω defined as below.

Definition 3.1 (Polygonal mesh)**.** *A polygonal mesh is a tuple* $\Omega_h = (\mathcal{T}_h, \mathcal{E}_h, \mathcal{V}_h)$ *such that*

1. $\mathcal{T}_h$ *is a finite collection of non-empty open polygons* T *with boundary* ∂T*, centroid* $\boldsymbol{x}_T$ *and diameter*

h_T that forms a partition of Ω, i.e.

$$\overline{\Omega} = \bigcup_{T \in \mathcal{T}_h} \overline{T} \qquad \forall\ T_1, T_2 \in \mathcal{T}_h,\ T_1 \neq T_2,\ T_1 \cap T_2 = 0 \tag{6}$$

2. *$\mathcal{E}_h$ is a finite collection of non-empty open one-dimensional hyperplanes in such a way that for any edge $e \in \mathcal{E}_h$, either there exist $T_1, T_2 \in \mathcal{T}_h$, such that $e \subset \partial T_1 \cap \partial T_2$ (in that case, e is called an internal edge), either there exists $T \in \mathcal{T}_h$, such that $e \subset \partial T \cap \partial\Omega$ (in that case, e is called a boundary edge)*

3. *$\mathcal{V}_h$ is a finite collection of vertices corresponding to the end points of each edge in $\mathcal{E}_h$*

where the index h corresponds to the mesh size, that is the maximal diameter among all the mesh elements.

In the following, N^V, N^E and N^P will respectively denote the total number of vertices, edges and polygons inside the mesh. At the local scale, N_T^V and N_T^E will denote the number of vertices and edges inside the element $T \in \mathcal{T}_h$.
On top of that, for any element $T \in \mathcal{T}_h$, $\mathcal{V}_T$ will denote the set of vertices of T and $\mathcal{E}_T$ the set of edges of T. For further purposes involving boundary conditions, we denote by $\mathcal{E}_h^{\mathrm{i}}$ the collection of internal edges and by $\mathcal{E}_h^{\mathrm{b}}$ the collection of boundary edges, such that $\mathcal{E}_h = \mathcal{E}_h^{\mathrm{i}} \cup \mathcal{E}_h^{\mathrm{b}}$. More precisely, we will distinguish between boundary edges inforced with Dirichlet condition $\mathcal{E}_h^{\mathrm{b,d}}$, Neumann condition $\mathcal{E}_h^{\mathrm{b,n}}$ and Robin conditions $\mathcal{E}_h^{\mathrm{b,r}}$.

In the following, we are interested in meshes made of regular-shaped elements, more specifically isotropic meshes with non-degenerate faces. Isotropic means here that we do not consider elements that become more and more stretched while refining the mesh and non-degenerate faces refer to edges whose diameter is uniformly comparable to the diameter of the element to which it belongs. Then, we must assert the following assumption to avoid badly shaped elements in the mesh.

Assumption 3.1 (Shape-regularity)**.** *A polygonal mesh Ω_h is said to be shape-regular if there exists a real number $\rho \in (0,1)$, independent of h, such that every element $T \in \mathcal{T}_h$ is star-shaped with respect to a ball of radius*

$$r_T \geq \rho h_T \tag{7}$$

where h_T is the diameter of T.

In general, the mesh is assumed to be shape-regular as stated above. However, such an assumption is purely theoretical, and, in practice, this condition can be weakened by considering the mesh to be shape-regular whenever its elements consist of a union of star-shaped subsets [16].

Polynomial spaces. Let $\mathcal{O} \subset \mathbb{R}^2$ be open. In the following, $\mathbb{P}^k(\mathcal{O})$ will denote the space of polynomials of degree less than or equal to k over $\mathcal{O}$, where $\mathcal{O}$, in practice, can be an element $T \in \mathcal{T}_h$ or an edge $e \in \mathcal{E}_h$. Since we are restricted to the two-dimensional case, we define

$$n_k \coloneqq \dim \mathbb{P}^k(T) = \frac{(k+1)(k+2)}{2},$$

the dimension of the local polynomial space. We also consider a multiindex $\boldsymbol{\alpha} = (\alpha_1, \alpha_2) \in \mathbb{N}^2$ with length $|\boldsymbol{\alpha}| = \alpha_1 + \alpha_2$ such that, if $\boldsymbol{x} = (x, y) \in \mathbb{R}^2$, then $\boldsymbol{x}^{\boldsymbol{\alpha}} = x^{\alpha_1} y^{\alpha_2}$. We will work with scaled monomials of the form

$$m_{\boldsymbol{\alpha}} \coloneqq \left(\frac{\boldsymbol{x} - \boldsymbol{x}_{\mathcal{O}}}{h_{\mathcal{O}}} \right)^{\boldsymbol{\alpha}} \tag{8}$$

where $\boldsymbol{x}_{\mathcal{O}}$ is the barycenter of $\mathcal{O}$ and $h_{\mathcal{O}}$ its diameter. The set of all such polynomials of degree less than or equal to k will be denoted by $\mathcal{M}_k(\mathcal{O}) \coloneqq \{m_{\boldsymbol{\alpha}} \colon |\boldsymbol{\alpha}| \leq k\}$, which constitutes a basis of $\mathbb{P}^k(\mathcal{O})$.

For the sake of simplicity, we will associate to each scaled monomial of multi-index $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)$ a scalar index $\alpha = \iota(\boldsymbol{\alpha})$ via the natural pairing

$$
\begin{array}{ccc}
(\alpha_1, \alpha_2) & \longleftrightarrow & \alpha \\
 & & \\
(0,0) & \longleftrightarrow & 1 \\
(1,0) & \longleftrightarrow & 2 \\
(0,1) & \longleftrightarrow & 3 \\
\vdots & & \vdots
\end{array}
$$

which is given by the following formula

$$\iota(\alpha_1, \alpha_2) = \frac{(\alpha_1 + \alpha_2)(\alpha_1 + \alpha_2 + 1)}{2} + \alpha_2 + 1. \tag{9}$$

3.2 Virtual Element Discretization

A virtual element is a particular type of finite element $(T, V_h(T), D_T)$ in the sense of Ciarlet [9, 15], where T is a polygonal element of $\mathcal{T}_h$, $V_h(T)$ is a local space of dimension N_T containing polynomials enriched with non-polynomial functions defined implicitly through a local PDE on T and $D_T = \{D_i\}_{1 \leq i \leq N_T}$ is the set of local degrees of freedom. As a consequence, any local basis $\{\varphi_j\}_{1 \leq j \leq N_T}$ satisfies

$$D_i(\varphi_j) = \delta_i^j, \qquad \forall i, j \in \{1, \dots, N_T\}, \tag{10}$$

with δ_i^j, the usual kronecker symbol. This relation is fundamental and is a consequence of the unisolvence property of the set of degrees of freedom—in other words, it expresses the fact that the operator $\underline{D_T} \colon v \in V_h(T) \to (D_1(v), \dots, D_{N_T}(v)) \in \mathbb{R}^{N_T}$ is bijective [19].

Local Projection Operators. One of the key ingredient for building the VEM are projection operators onto local polynomial spaces. They are essential to the virtual element discretization in order to compute the virtual functions. Traditionally, two kind of projections are considered: the L^2-projection and the H^1-projection—also called the elliptic projection.

Definition 3.2 (H^1-Projection)**.** *Let $T \in \mathcal{T}_h$ and $k \geq 1$ be an integer. We define the H^1-projection $\Pi_T^{1,k} : V_h(T) \to \mathbb{P}^k(T)$ such that, for any $v \in H^1(T)$,*

$$\begin{cases} \displaystyle\int_T \boldsymbol{\nabla} \Pi_T^{1,k} v_h \cdot \boldsymbol{\nabla} p = \int_T \boldsymbol{\nabla} v_h \cdot \boldsymbol{\nabla} p, & \forall p \in \mathbb{P}^k(T)/\mathbb{P}^0(T) \\ \mathrm{P}_0 \left(\Pi_T^{1,k} v_h \right) = \mathrm{P}_0 v_h, \end{cases} \tag{11}$$

where $\mathrm{P}_0 : V_h(T) \to \mathbb{P}^0(T)$ is the projection operator onto constants defined by

$$\mathrm{P}_0 v_h = \begin{cases} \dfrac{1}{N_V} \displaystyle\sum_{i=1}^{N_V} v_h(x_i) & \text{for } k = 1 \\ \dfrac{1}{|T|} \displaystyle\int_T v_h & \text{for } k \geq 2 \end{cases}. \tag{12}$$

This operator is crucial for keeping the system Eq. (11) solvable. Indeed, as one can notice, the first row in Eq. (11) does not hold when $p \in \mathbb{P}^0(T)$. That is why we fix this issue by adding another equality using the projection operator onto constants.

Definition 3.3 (L^2-Projection)**.** *Let $T \in \mathcal{T}_h$ and $k \geq 0$ an integer. We define the L^2-projection $\Pi_T^{0,k} : V_h(T) \to \mathbb{P}^k(T)$ such that, for any $v \in L^2(T)$,*

$$\int_T \Pi_T^{0,k} v_h \; p = \int_T v_h \; p \qquad \forall p \in \mathbb{P}^k(T) \tag{13}$$

Another important projection operator is the L^2-projection of the gradient $\boldsymbol{\Pi}_T^{0,k-1} \colon \boldsymbol{\nabla} V_h(T) \to \mathbb{P}^{k-1}(T)^2$ defined as above as gradients of functions of the virtual space $\boldsymbol{\nabla} v_h$ against vectorial polynomials $p \in \mathbb{P}^{k-1}(T)^2$.

Local Virtual Space. We define the local virtual space for all $T \in \mathcal{T}_h$ by

$$\begin{aligned} V_h(T) = \big\{ v_h \in H^1(\Omega) \cap C^0(\partial T) \colon & (i) \quad v_h|_E \in \mathbb{P}^k(E),\ \forall E \subset \partial T \\ & (ii) \quad \Delta v \in \mathbb{P}^{k-2}(T) \\ & (iii) \quad (v_h - \Pi_T^{1,k} v_h, p) = 0,\ \forall p \in \mathbb{P}^k(T)/\mathbb{P}^{k-2}(T) \big\}. \end{aligned} \tag{14}$$

The first two conditions (i) and (ii) show that this space is composed of functions that are polynomials of degree at most k on the edges E of the element T such that they are globally continuous on the boundary ∂T—which means that there are no discontinuities at the nodes. Moreover, the virtual functions inside the element are required to solve a Poisson problem in the weak sense.
It is possible to require the functions inside the element to verify another kind of local PDE. In fact, a wide range of virtual spaces have been built by considering other local problems, such as the divergence-free virtual space [12].
The last condition (iii) is added to render the L^2-projection computable as firstly proposed in [2]. We are then dealing with an enhanced virtual space which is larger than the classical one initially introduced in [8].

Local Degrees of Freedom. In order to represent our solution onto the mesh, we need to attach to each local virtual space a set of local degrees of freedom (DOF). For a 2D VEM, there are three types of DOF fixed to each geometrical component of the element (as depicted in Fig. 3) such that, for all $T \in \mathcal{T}_h$ and for $v_h \in V_h(T)$, we select

1. the value of v_h at the vertices of T: $\forall V \in \mathcal{V}_T$,

$$D^V(v_h) = v_h(\boldsymbol{x}_V) \tag{15}$$

   where $\boldsymbol{x}_V$ corresponds to the coordinates of vertex V;

2. the value of v_h at the $k-1$ internal points of the $(k+1)$-point Gauss-Lobatto quadrature rule or, equivalently, the moments of v_h against the monomials of degree up to $k-1$ on the edge: $\forall E \in \mathcal{E}_T$,

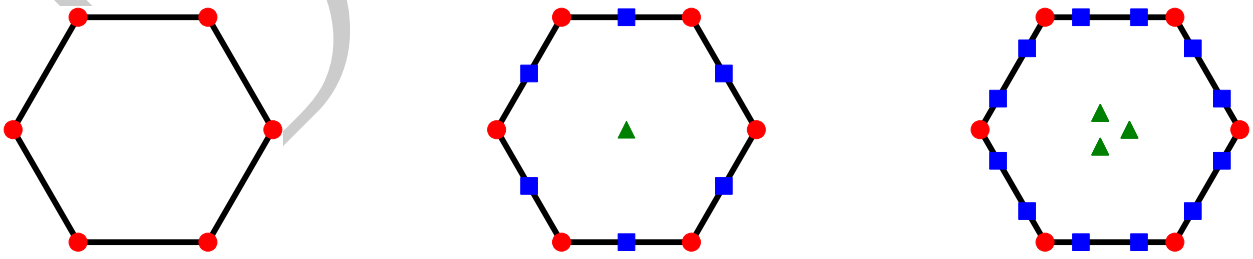
$$D^E(v_h) = \frac{1}{|E|} \int_E v_h m_\alpha \tag{16}$$

   with $m_\alpha \in \mathcal{M}_{k-1}(E)$;

3. the moments of v_h against the monomials of degree up to $k-2$ in the element:

$$D^T(v_h) = \frac{1}{|T|} \int_T v_h m_\alpha \tag{17}$$

   with $m_\alpha \in \mathcal{M}_{k-2}(T)$.



$k = 1$ $k = 2$ $k = 3$

Fig. 3: 2D element with ● : vertex DOFs, ■ : Edge DOFs, ▲ : Cell DOFs.

The set of local degrees of freedom therefore consists of

$$D_T = \{D^V\}_{V\in\mathcal{V}_T} \cup \{D^E\}_{E\in\mathcal{E}_T} \cup \{D^T\}.$$

with

$$\operatorname{card}(D_T) = N_T^V + (k-1)N_T^E + n_{k-2} = kN_T^V + n_{k-2} = \dim(V_h(T)) = N_T.$$

In the future, the set of DOFs will be indexed as $D_T = \{D_i\}_{1\le i\le N_T}$, where the DOFs are being ordered as above—that is vertices for $1 \le i \le N_T^V$, edges for $N_T^V + 1 \le i \le kN_T^V$ and cell for $kN_T^V + 1 \le i \le kN_T^V + n_{k-2}$.

3.3 Discrete Problem

As in the FEM, the discrete problem is written using the Galerkin method which reads

$$\begin{cases} \text{Find } u_h \in V_h \\ a_h(u_h, v_h) = l_h(v_h), \quad \forall v_h \in V_h. \end{cases} \tag{18}$$

The global approximation space V_h is obtained by glueing continuously the local virtual spaces $V_h(T)$ over the all the elements $T \in \mathcal{T}_h$, that is

$$V_h = \{v_h \in H^1(\Omega) \cup C^0(\overline{\Omega})\colon v_h|_T \in V_h(T),\ \forall T \in \mathcal{T}_h\}$$

which dimension is

$$N = N^V + (k-1)N^E + N^T n_{k-2}.$$

The only fundamental difference with the FEM relies on how the bilinear form a_h and the load term l_h are defined. Indeed, the VEM exploits the orthogonal decomposition of the global space V_h with respect to the L^2-projection $\Pi_h^{0,k}$ for the mass term and the elliptic projection $\Pi_h^{1,k}$ for the stiffness. From those decompositions, the bilinear form a_h inherits a consistent term that is exact for polynomials and a stability term that approximates the non-polynomial part of the virtual space.

Bilinear form a_h. Before defining the global discrete bilinear form $a_h\colon V_h \times V_h \to \mathbb{R}$, we discretize each term that makes it up, starting with the stiffness term $b_h\colon V_h \times V_h \to \mathbb{R}$ which, for all $u_h, v_h \in V_h$, reads

$$b_h(u_h, v_h) \coloneqq -\sum_{T\in\mathcal{T}_h} \int_T \boldsymbol{\nabla}\Pi_T^{1,k} u_h \cdot \boldsymbol{\nabla}\Pi_T^{1,k} v_h - \sum_{T\in\mathcal{T}_h} s_h^{1,T}\left((I-\Pi_T^{1,k})u_h, (I-\Pi_T^{1,k})v_h\right) \tag{19}$$

where $s_h^{1,T} : V_h(T) \times V_h(T) \to \mathbb{R}$ is a bilinear form satisfying the stability property: for all $v_h \in V_h(T)$, there exist $\lambda_*, \lambda^* > 0$ such that

$$\lambda_* b(v_h, v_h) \le s_h^{1,T}\left((I-\Pi_T^{1,k})v_h, (I-\Pi_T^{1,k})v_h\right) \le \lambda^* b(v_h, v_h).$$

In the same spirit, we define the mass term $m_h\colon V_h \times V_h \to \mathbb{R}$ for all $u_h, v_h \in V_h$ by

$$m_h(u_h, v_h) \coloneqq \kappa^2 \sum_{T\in\mathcal{T}_h} \int_T \Pi_T^{0,k} u_h \cdot \Pi_T^{0,k} v_h + \kappa^2 \sum_{T\in\mathcal{T}_h} s_h^{0,T}\left((I-\Pi_T^{0,k})u_h, (I-\Pi_T^{0,k})v_h\right) \tag{20}$$

where, as before, $s_h^{0,T}\colon V_h(T) \times V_h(T) \to \mathbb{R}$ is a stable bilinear form, i.e. for all $v_h \in V_h(T)$, there exist $\mu_*, \mu^* > 0$ such that

$$\mu_* m(v_h, v_h) \le s_h^{1,T}\left((I-\Pi_T^{0,k})v_h, (I-\Pi_T^{0,k})v_h\right) \le \mu^* m(v_h, v_h).$$

Remark 3.1. *The stabilisation term is required to hold the well-posedness of the discrete variational problem. There are multiple ways of defining it, the most classical one being the so-called "dofi-dofi" stabilisation term expressed as follows*

$$s_h^{r,T}(u_h, v_h) = \sigma_T^r \sum_{i=1}^{N_T} D_i\left((I - \Pi_T^{r,k})u_h\right) D_i\left((I - \Pi_T^{r,k})v_h\right) \qquad \forall u_h, v_h \in V_h \tag{21}$$

where $r = 0, 1$ enables us to distinguish between the stiffness and mass stabilization case and σ_T^r is a scaling coefficient depending on each case. For $r = 0$, the coefficient usually scales as a gradient, typically $\sigma_T^0 = h_T^2$ or $\sigma_T^0 = |T|$. For $r = 1$, we usually take $\sigma_T^1 = 1$.

Finally, we define the discrete boundary term $r_h \colon V_h \times V_h \to \mathbb{R}$ inherited from the Robin boundary condition, for all $u_h, v_h \in V_h$, by

$$r_h(u_h, v_h) \coloneqq -i\kappa \sum_{E \in \mathcal{E}_h^{\mathrm{b,r}}} \int_E u_h v_h.$$

Remark 3.2. *Note that, in this case, no projection operator is needed before the virtual functions u_h and v_h because, on the boundary of an element, they correspond to polynomials of degree k, which renders the integral over the edge fully computable. However, this imply that we need to integrate a polynomial of degree $2k$ over the edge e, which is a delicate tack that we detail in the dedicated section 4.2*

In the end, the global bilinear form $a_h \colon V_h \times V_h \to \mathbb{R}$ is defined as the sum of all those term, i.e for all $u_h, v_h \in V_h$ we have

$$a_h(u_h, v_h) \coloneqq b_h(u_h, v_h) + m_h(u_h, v_h) + r_h(u_h, v_h).$$

Load term l_h. The load term $l_h \colon V_h \to \mathbb{R}$ writes, for all $v_h \in V_h$

$$l_h(v_h) \coloneqq \sum_{T \in \mathcal{T}_h} \int_T f \Pi_T^{0,k} v_h - \sum_{E \in \mathcal{E}_h^{\mathrm{b,n}}} \int_E g_N v_h - \sum_{E \in \mathcal{E}_h^{\mathrm{b,r}}} \int_E g_R v_h.$$

Linear System. By writing the discrete solution u_h in the basis virtual basis as

$$u_h = \sum_{i=1}^{N} D_i(u_h)\varphi_i,$$

one can rewrite Eq. (18) in its equivalent algebraic form as the system

$$\mathbf{A}_h \mathbf{u}_h = \mathbf{l}_h \tag{22}$$

where $\mathbf{u}_h = (D_1(u_h), \dots, D_N(u_h))^T$ is the discrete unknown, $\mathbf{l}_h = (l_h(\varphi_j))_{1 \le j \le N}$ is the load term and $\mathbf{A}_h = \mathbf{K}_h + \mathbf{M}_h + \mathbf{R}_h$ is the global matrix formed by the sum of the matrices associated with the corresponding terms

$$\begin{aligned}
\mathbf{K}_h &= (b_h(\varphi_i, \varphi_j))_{1 \le i,j \le N} \\
\mathbf{M}_h &= (m_h(\varphi_i, \varphi_j))_{1 \le i,j \le N} \\
\mathbf{R}_h &= (r_h(\varphi_i, \varphi_j))_{1 \le i,j \le N}
\end{aligned}$$

The challenging task now lies in the computation of those matrices. As explained in the next section, it is achieved by applying the integration by part formula and exploiting the properties of the projection operators.

4 Implementation Guideline for the Virtual Element Method

In this section, we outline the implementation of the VEM, taking a closer look at the computation of local projection matrices and the application of boundary conditions, specifically Robin boundary conditions. It is worth mentioning to the interested reader that several implementation guides for the VEM have already been published: the hitchhiker's guide for the classical VEM [5], the guide for the divergence-free VEM for mixed problem [14] and, more recently, more general implementation guides for coding the VEM in Matlab [28, 22].

For the interesting reader that first wants to take the VEM in hand before coding it, we point out that there already exist many open-source virtual element codes written in different programming languages. Here is a non-exhaustive list: two oriented-object C++ libraries called Veamy [26] and Vem++ [13], a Python module called dune-vem [15], a Matlab package called VEMLab, and many more.

Our programming approach follows the classical implementation path illustrated in Fig. 4 which relies on three main steps that are almost self-sufficient in the sense that they can be coded and verified separately and then connected all together to avoid errors. Depending on the programming language preferred, one could adopt an object-oriented point of view and conceive each step as a class that contains the corresponding objects and methods. Otherwise, a partial implementation can be considered by augmenting an already existing finite element code since the structure is the same and the only main difference lies in the computation of local matrices.

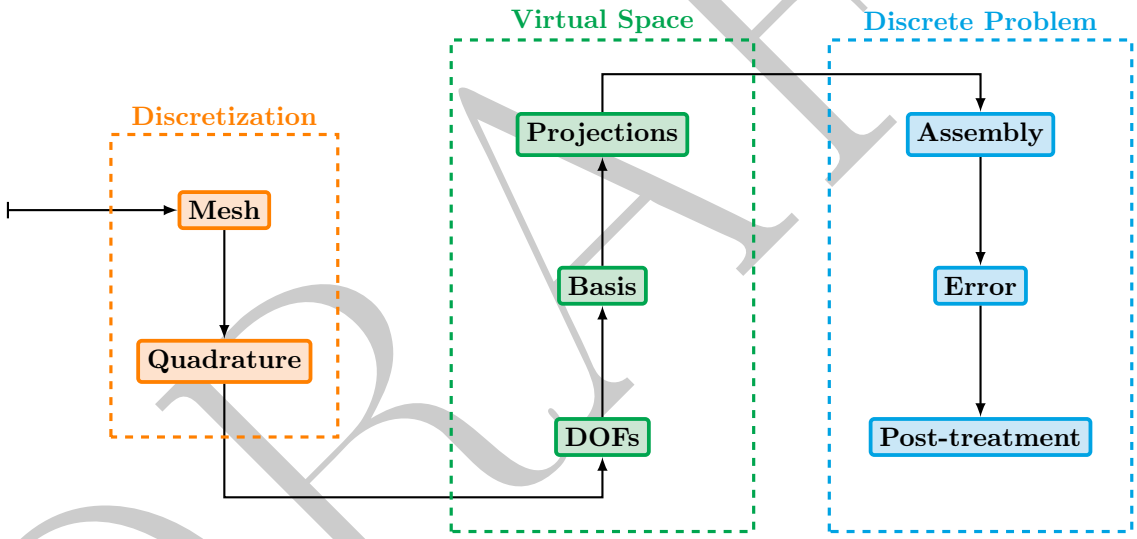


Fig. 4: Implementation path for coding the virtual element method

Discretization. Any code for a polytopal method needs two basic tools: a mesh generator and quadrature rules. The mesh generation can be achieved by using open-source software such as Gmsh [18] and PyPolyMesher [1, 29] which can also handle polygonal and polyhedral decompositions. More generally, an appropriate mesh structure should possess three main fields `vertex`, `edge` and `polygon` which contain the indices and the important geometrical characteristics (coordinates, measure, centroid, etc.) of the corresponding objects and their subordinate objects (e.g. the `polygon` field should contain the indices of all the polygons in the mesh, their geometrical characteristics such as the area, the barycenter and so on, and, for each polygon, it must return to the indices of their respective edges in the field `edge`).

The other crucial point is the quadrature rule for integrating functions, specifically polynomials. In general, considering an open subset $\mathcal{O} \subset \mathbb{R}^2$—which, in practice, can correspond to an element or an edge—, one should equip the code with a quadrature rule $\mathcal{Q}(\mathcal{O}) = ((\boldsymbol{x}_k, w_k))_{1 \le k \le n_{\mathcal{Q}}}$ over $\mathcal{O}$ such that, for any function $f \colon \mathcal{O} \to \mathbb{R}$, we have

$$\int_{\mathcal{O}} f(\boldsymbol{x})d\boldsymbol{x} \simeq \sum_{k=1}^{n_{\mathcal{Q}}} f(\boldsymbol{x}_k)w_k. \tag{23}$$

As stated above, the preferred quadrature rule for an edge is the Gauss-Lobatto one. For a polygon, one can consider the quadrature rule composed of the ones on each triangle diving the polygon. Another recent technique [4] makes use of the Stokes formula to express any integral over an element as a combination of integrals over lower-dimensional objects (edges or vertices) as explained in [22].

Virtual Space. The core of the code is the computation of the local matrices. To perform such calculations, three ingredients are required: a numbering of degrees of freedom, a way of evaluating the monomial basis and, finally, a routine for computing local projections.
The numbering of degrees of freedom should be done globally and can be based on the indices given by the mesh structure. It is usually done by ordering the DOFs in the following way: `vertex DOFs`, `edge DOFs` and `cell DOFs`—just as globally as locally. The main difference at the local level is that the DOFs are, by convention, arranged clockwise around the element.
Regarding the local basis, it is preferable to pre-compute the monomials at the points of interest (quadrature points for instance) to save computational time. One can even pre-compute the entire monomial basis by considering a matrix of the form $(m_i(\boldsymbol{x}_j))_{ij}$ where the $\boldsymbol{x}_j$ are the points of interest.
All those features are essential for computing the local projections as detailed below in section 4.1 where the expression of the projection matrices, which is obtained by solving a system locally on the element, is then used to compute the local stiffness and mass matrices.

Discrete Problem. Finally, the discrete problem is managed as in the H^1-conforming FEM. Indeed, the assembly is performed by addding up the local matrices to the global matrix. A sparse assembly with a triplet containing the row index, the column index and the corresponding matrix value is preferred since it is less time consuming. The solution obtained by resolution of the global linear system can then be used for the error analysis and other possible post-treatment.

4.1 Computing the Local Projection Matrices

Local Stiffness Matrix. As explained above, one can write the local stiffness matrix $\mathbf{K}_T$ for all $T \in \mathcal{T}_h$ by

$$\mathbf{K}_T = \int_T \boldsymbol{\nabla}\varphi_i \cdot \boldsymbol{\nabla}\varphi_j = \int_T \boldsymbol{\nabla}{\Pi_T}^{1,k}\varphi_i \cdot \boldsymbol{\nabla}{\Pi_T}^{1,k}\varphi_j + s_h^{1,T}\left(\boldsymbol{\nabla}(I - {\Pi_T}^{1,k})\varphi_i, \boldsymbol{\nabla}(I - {\Pi_T}^{1,k})\varphi_j\right) \tag{24}$$

Therefore, to compute the stiffness matrix, it is sufficient to calculate the matrix $\mathbf{\Pi^1}$ associated with the projection operator $\Pi_T^{1,k}$. To do so, we exploit the relation Eq. (11) that characterizes the elliptic projection for a certain virtual basis function φ_j, writing $\Pi_T^{1,k}\varphi_j$ in the monomial basis

$$\Pi_T^{1,k}\varphi_j = \sum_{i=1}^{n_k} s_j^i m_i, \tag{25}$$

which gives us the following system

$$\sum_{i=1}^{n_k} s_j^i \int_T \boldsymbol{\nabla}m_i \cdot \boldsymbol{\nabla}m_\alpha = \int_T \boldsymbol{\nabla}\varphi_j \cdot \boldsymbol{\nabla}m_\alpha \qquad \text{for } 2 \le \alpha \le n_k \tag{26.a}$$

$$\sum_{i=1}^{n_k} s_j^i \, \mathrm{P}_0 m_i = \mathrm{P}_0\varphi_j \qquad \text{for } \alpha = 1 \tag{26.b}$$

which writes equivalently in matrix form

$$\mathbf{G}\mathbf{s}_j = \mathbf{B}_j \qquad \text{for } 1 \le j \le N_T \text{ fixed} \tag{27}$$

Matrix G. We denote by $\mathbf{G}$ the $n_k \times n_k$ matrix

$$\mathbf{G} := \begin{bmatrix} \mathrm{P}_0 m_1 & \mathrm{P}_0 m_2 & \cdots & \mathrm{P}_0 m_{n_k} \\ 0 & (\boldsymbol{\nabla} m_2, \boldsymbol{\nabla} m_2)_{0,T} & \cdots & (\boldsymbol{\nabla} m_2, \boldsymbol{\nabla} m_{n_k})_{0,T} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & (\boldsymbol{\nabla} m_{n_k}, \boldsymbol{\nabla} m_2)_{0,T} & \cdots & (\boldsymbol{\nabla} m_{n_k}, \boldsymbol{\nabla} m_{n_k})_{0,T} \end{bmatrix}. \tag{28}$$

This matrix can easily be computed by integrating the monomials over the element T, using a quadrature rule on T as in Eq. (23).

Once $\mathbf{G}$ has been computed, the system Eq. (27) can be extended by varying $1 \le j \le N_T$ instead of keeping it fixed. Hence, the right-hand side is not a vector anymore but a matrix.

Matrix B. We denote by $\mathbf{B}$ the $n_k \times N_T$ matrix

$$\mathbf{B} := \begin{bmatrix} \mathrm{P}_0 \varphi_1 & \cdots & \mathrm{P}_0 \varphi_{N_T} \\ (\boldsymbol{\nabla} m_2, \boldsymbol{\nabla} \varphi_1)_{0,T} & \cdots & (\boldsymbol{\nabla} m_2, \boldsymbol{\nabla} \varphi_{N_T})_{0,T} \\ \vdots & \ddots & \vdots \\ (\boldsymbol{\nabla} m_{n_l}, \boldsymbol{\nabla} \varphi_1)_{0,T} & \cdots & (\boldsymbol{\nabla} m_{n_l}, \boldsymbol{\nabla} \varphi_{N_T})_{0,T} \end{bmatrix}. \tag{29}$$

The first line of the matrix is easy to compute. Indeed, by using the expression of projection operator onto constants P_0 given by Eq. (12) and by recalling that φ_j satisfies Eq. (10), one has

$$\mathbf{B}_{1j} = \begin{cases} 1 & \text{for } k = 1 \\ \delta_j^{kN_T^V + 1} & \text{for } k \ge 2 \end{cases} \qquad \text{for } 1 \le j \le N_T$$

For $\alpha \ge 2$, we perform an integration by part

$$\mathbf{B}_{\alpha j} = \int_T \boldsymbol{\nabla} m_\alpha \cdot \boldsymbol{\nabla} \varphi_j = -\int_T \Delta m_\alpha \varphi_j + \int_{\partial T} (\boldsymbol{\nabla} m_\alpha \cdot \boldsymbol{n}) \varphi_j.$$

The matrix can then be exactly split into two blocks corresponding to each integral of the right-hand side. Let us start with the boundary integral, we have for $1 \le j \le kN_T^V$

$$\mathbf{B}_{\alpha j} = \int_{\partial T} (\boldsymbol{\nabla} m_\alpha \cdot \boldsymbol{n}) \varphi_j = \sum_{E \in \mathcal{E}_T} \int_E (\boldsymbol{\nabla} m_\alpha \cdot \boldsymbol{n}_E) \varphi_j$$

with $\boldsymbol{n}_E$ the outward normal of the edge E. Since the basis function φ_j takes the value 1 at the point $\boldsymbol{x}_j$ and 0 everywhere else, the sum over the edges then reduces into one or two terms, depending if the point $\boldsymbol{x}_j$ is attached to a vertex V_j or an edge E_j

$$\mathbf{B}_{\alpha j} = \begin{cases} (\boldsymbol{\nabla} m_\alpha(\boldsymbol{x}_j) \cdot \boldsymbol{n}_{E_j} + \boldsymbol{\nabla} m_\alpha(\boldsymbol{x}_j) \cdot \boldsymbol{n}_{E_{j+1}}) w_j & \text{if attached to } V_j \\ (\boldsymbol{\nabla} m_\alpha(\boldsymbol{x}_j) \cdot \boldsymbol{n}_{E_j}) w_j & \text{if attached to } E_j \end{cases}$$

where w_j is the Gauss-Lobatto weight associated to the point $\boldsymbol{x}_j$.

On the other hand, for the integral over the element, by computing the Laplacian of m_α (using Eq. (8)), we get for $kN_T^V + 1 \le j \le kN_T^V + 1 + n_{k-2}$

$$\mathbf{B}_{\alpha j} = -\int_T \Delta m_\alpha \varphi_j = -\frac{\alpha_1(\alpha_1 - 1)}{h_T^2} \int_T m_\beta \varphi_j - \frac{\alpha_1(\alpha_1 - 1)}{h_T^2} \int_T m_\gamma \varphi_j$$

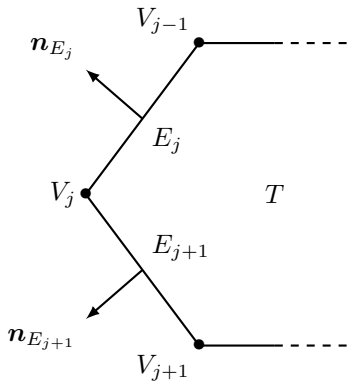


Fig. 5: Sketch of integration over the edges of the element T.

where $\beta = \iota(\alpha_1 - 2, \alpha_2)$ and $\gamma = \iota(\alpha_1, \alpha_2 - 2)$ are the corresponding indices which can be computed through Eq. (9) for $\alpha_1, \alpha_2 \geq 1$. The first integral on the right-hand side corresponds to an internal degree of freedom if and only if $m_\beta \in \mathcal{M}_{k-2}(T)$, i.e. $1 \leq \beta \leq n_{k-2}$, otherwise it reduces to zero. By the same reasoning on the second internal, we get that

$$\mathbf{B}_{\alpha j} = \begin{cases} -\dfrac{\alpha_1(\alpha_1 - 1)|T|}{h_T^2} & \text{if } 1 \leq \beta \leq n_{k-2} \\ -\dfrac{\alpha_2(\alpha_1 - 2)|T|}{h_T^2} & \text{if } 1 \leq \gamma \leq n_{k-2} \end{cases}.$$

We can now solve the system and compute the matrix associated with the elliptic projection $\mathbf{\Pi}_*^1 = \mathbf{G}^{-1}\mathbf{B}$. However $\mathbf{\Pi}_*^1 \colon \mathbb{R}^{n_k} \to \mathbb{R}^{N_T}$ represents the elliptic projection within the monomial basis $\mathcal{M}_k(T)$. Therefore, we need to compute an additional matrix that will allow us to write the matrix $\mathbf{\Pi}^1 \colon \mathbb{R}^{N_T} \to \mathbb{R}^{N_T}$ which represents the elliptic projection within the virtual basis.

Matrix D. We denote by $\mathbf{D}$ the $N_T \times n_k$ matrix given by

$$\mathbf{D} = \begin{bmatrix} D_1(m_1) & D_1(m_2) & \cdots & D_1(m_{n_k}) \\ D_2(m_1) & D_2(m_2) & \cdots & D_2(m_{n_k}) \\ \vdots & \vdots & \ddots & \vdots \\ D_{N_T}(m_1) & D_{N_T}(m_2) & \cdots & D_{N_T}(m_{n_k}) \end{bmatrix}. \tag{30}$$

or, equivalently

$$\mathbf{D}_{i\alpha} = D_i(m_\alpha), \quad \text{for } 1 \leq i \leq N_T \quad \text{and } 1 \leq \alpha \leq n_k,$$

This matrix can be easily computed from the expression of degrees of freedom. For the DOFs on the boundary of the element (vertices and edges), it corresponds to the evaluation of the monomials at the corresponding points $\boldsymbol{x}_i$—the vertex points in one case and the $k-1$ internal Gauss-Lobatto points in the other. For the DOFs inside the cell, it suffices to compute the moment of every monomial against the monomials $m_j \in \mathcal{M}_{k-2}(T)$. In short, we have

$$\mathbf{D}_{i\alpha} = \begin{cases} m_\alpha(\boldsymbol{x}_i) & \text{for } 1 \leq i \leq kN_T^V \\ \dfrac{1}{|T|}\displaystyle\int_T m_\alpha m_j \quad \text{for } 1 \leq j \leq n_{k-2}, \text{ and } kN_T^V + 1 \leq i \leq N_T \end{cases}.$$

Consequently, the matrix $\mathbf{D} \colon \mathbb{R}^{n_k} \to \mathbb{R}^{N_T}$ can be understood as a change-of-basis matrix from the basis of monomials $\mathcal{M}_k(T)$ to the virtual basis $(\varphi_j)_{1 \leq j \leq N_T}$ of $V_h(T)$.

H^1-Projection $\mathbf{\Pi}^1$. We denote by $\mathbf{\Pi}^1$ the matrix defined by

$$\mathbf{\Pi}^1 = \mathbf{D}\mathbf{\Pi}^1_* = \mathbf{D}(\mathbf{G}^{-1}\mathbf{B}). \tag{31}$$

A practical way of checking that $\mathbf{\Pi}^1$ is well-computed is to check that

$$\mathbf{G} = \mathbf{B}\mathbf{D}.$$

Finally, the local stiffness matrix $\mathbf{K}_T$ reads

$$\mathbf{K}_T = (\mathbf{\Pi}^1_*)^{\mathrm{T}}\widetilde{\mathbf{G}}(\mathbf{\Pi}^1_*) + \sigma^1_T(\mathbf{I} - \mathbf{\Pi}^1)^{\mathrm{T}}(\mathbf{I} - \mathbf{\Pi}^1) \tag{32}$$

where $\widetilde{\mathbf{G}}$ is equal to $\mathbf{G}$ except for the first row which is set to zero, and the stabilization parameter $\sigma^1_T = 1$ as explained in remark 3.1.

Local L^2-Projection. For all $T \in \mathcal{T}_h$, the local mass matrix $\mathbf{M}_T$ can be written in the following form:

$$\mathbf{M}_T = \int_T \varphi_i\varphi_j = \int_T \Pi^{0,k}_T\varphi_i\Pi^{0,k}_T\varphi_j + s^{0,T}_h\left((I - \Pi^{0,k}_T)\varphi_i, (I - \Pi^{0,k}_T)\varphi_j\right).$$

To compute the local mass matrix, it is then sufficient to calculate the matrix $\mathbf{\Pi}^0$ associated with L^2-projection operator $\Pi^{0,k}_T$. We consider the decomposition of $\Pi^{0,k}_T\varphi_j$ in the monomial basis

$$\Pi^{0,k}_T\varphi = \sum_{i=1}^{n_k} t^i_j m_i$$

which, injected in Eq. (13), gives us the following system

$$\sum_{i=1}^{n_k} t^i_j \int_T m_i m_\alpha = \int_T \varphi_j m_\alpha \qquad \text{for } 1 \le \alpha \le N_T$$

which can also be written

$$\mathbf{H}\mathbf{t}_j = \mathbf{C}_j.$$

Matrix H. We denote by $\mathbf{H}$ the $n_k \times n_k$ matrix given by

$$\mathbf{H} \coloneqq \begin{bmatrix} (m_1, m_1)_{0,T} & \cdots & (m_1, m_{n_k})_{0,T} \\ \vdots & \ddots & \vdots \\ (m_{n_k}, m_2)_{0,T} & \cdots & (m_{n_k}, m_{n_k})_{0,T} \end{bmatrix}, \tag{33}$$

or, equivalently

$$\mathbf{H}_{\alpha\beta} = (m_\alpha, m_\beta)_{0,T}, \quad \text{for } 1 \le \alpha, \beta \le n_k,$$

This matrix can easily be computed by integrating the monomials over the element T, using the quadrature rule on T as in Eq. (23).

Matrix C. We denote by $\mathbf{C}$ the $n_k \times N_T$ matrix given by

$$\mathbf{C}_{ij} \coloneqq (m_i, \varphi_j)_{0,T}, \quad \text{for } 1 \le i \le n_k \quad \text{and } 1 \le j \le N_T \tag{34}$$

This matrix can be treated into two different parts. First, we observe that, when $1 \le i \le n_{k-2}$, $\mathbf{C}_{ij}$ can be rewritten as a degree of freedom

$$\mathbf{C}_{ij} = |T|D_i(\varphi_j) = |T|\delta^j_i$$

which gives us an identity matrix scaled by the area of the polygon $|T|$ for the first block $1 \leq i, j \leq n_{k-2}$. Then, when $n_{k-2}+1 \leq i \leq n_k$, one recognizes the enhancing condition of the virtual space defined above Eq. (14) and we can hence write

$$\mathbf{C}_{ij} = (m_i, \Pi_T^{1,k}\varphi_j)_{0,T} = (\mathbf{H}\mathbf{G}^{-1}\mathbf{B})_{ij}.$$

L^2-Projection $\mathbf{\Pi}^0$. We denote by $\mathbf{\Pi}^0$ the matrix defined by

$$\mathbf{\Pi}^0 = \mathbf{D}\mathbf{\Pi}_*^0 = \mathbf{D}(\mathbf{H}^{-1}\mathbf{C}) \tag{35}$$

A practical way of checking that $\mathbf{\Pi}^0$ is well-computed is to check that

$$\mathbf{H} = \mathbf{C}\mathbf{D}.$$

Finally, the local mass matrix $\mathbf{M}_T$ reads

$$\mathbf{M}_T = \mathbf{C}^T\mathbf{H}^{-1}\mathbf{C} + \sigma_T^0(\mathbf{I}-\mathbf{\Pi}^0)^{\mathrm{T}}(\mathbf{I}-\mathbf{\Pi}^0) \tag{36}$$

where the stabilization parameter $\sigma_T^0 = |T|$ as explained in remark 3.1.

Local load term. The load term easily writes

$$\mathbf{l}_T = (\mathbf{\Pi}_*^0)^T F$$

where F is the vector formed by the moments of f against the monomials of $\mathcal{M}_k(T)$, i.e. $F = \left(\int_T f m_1, \ldots, \int_T f m_{n_k}\right)^T$.

Global Assembly. Below, in the algorithm 1 algorithm, we propose a pseudo-code implementation of the virtual element method and the algorithm 2 takes into account Dirichlet and Robin boundary conditions.

Algorithm 1 Global assembly of the linear system

1: **for** $T \in \mathcal{T}_h$ **do**
2: Compute B, D, G, C, H
3: Compute Π_*^1, Π^1, Π_*^0, Π^0
4: Compute the local stiffness matrix: $\mathbf{K}_T = (\mathbf{\Pi}_*^1)^{\mathrm{T}}\widetilde{\mathbf{G}}(\mathbf{\Pi}_*^1) + (\mathbf{I}-\mathbf{\Pi}^1)^{\mathrm{T}}(\mathbf{I}-\mathbf{\Pi}^1)$
5: Compute the local mass matrix: $\mathbf{M}_T = \mathbf{C}^{\mathrm{T}}\mathbf{H}^{-1}\mathbf{C} + |T|(\mathbf{I}-\mathbf{\Pi}^0)^{\mathrm{T}}(\mathbf{I}-\mathbf{\Pi}^0)$
6: $A_h \leftarrow A_h + (-\mathbf{K}_T + \kappa^2\mathbf{M}_T)$ ▷ Adds global contributions
7: **end for**

4.2 Imposing Boundary conditions

In this section, we focus on the implementation of mixed boundary conditions: a mixed Neumann and Dirichlet condition with a particular emphasis on calculating the Robin term $r_h(u_h, v_h)$ in our variational formulation.
Unlike the mass matrix $\mathbf{M}_h$ and the stiffness matrix $\mathbf{K}_h$, which are calculated using the virtual element formalism, the Robin matrix $\mathbf{R}_h$ is calculated in a manner analogous to Lagrange high-order finite elements, with the difference that the degrees of freedom are not placed in the same locations on the edge.
We can express the global matrix $\mathbf{R}_h$ associated to the formulation r_h in a basis of classical shape function, thus

$$(\mathbf{R}_h)_{ij} = \left(\int_{\Gamma_R} \alpha(x,y)\Phi_j(x,y), \Phi_i(x,y)\right)_{ij},$$

with $\alpha\colon \mathbb{R}^2 \to \mathbb{R}$. The boundary Γ_R can be decomposed into a sum of 1D elements that can be characterized by the segment $[\xi_0, \xi_0+\lambda]$ between two vertices V_0 and V_1. These elements are segments joining 2 consecutive

Algorithm 2 Imposing boundary conditions in the linear system

for $E \in \mathcal{E}_h^{\mathrm{b}}$ **do**
  if $E \in \mathcal{E}_h^{\mathrm{b,r}}$ **then**
    Compute $\mathbf{R}_E$
    $A_h \leftarrow A_h + \mathbf{R}_E$ ▷ Adds global contributions
  else if $e \in \mathcal{E}_h^{\mathrm{b,d}}$ **then**
    $A_h \leftarrow 1$ ▷ Set the line to 0 and add a 1 to the correct DDL position
    $l_h \leftarrow g_D$ ▷ Set the value of the DC on this DDL
  end if
end for

points of the edge. The Φ_i basis function attached to the i vertex of the edge, restricted to the edge element, is a polynomial of degree k on the edge. These characteristic 1D elements are shown in Fig. 6 with the degrees of freedom corresponding to the Gauss-Lobatto quadrature points on $[0,\lambda]$. Consequently, the higher the order, the more points there will be on the segment.

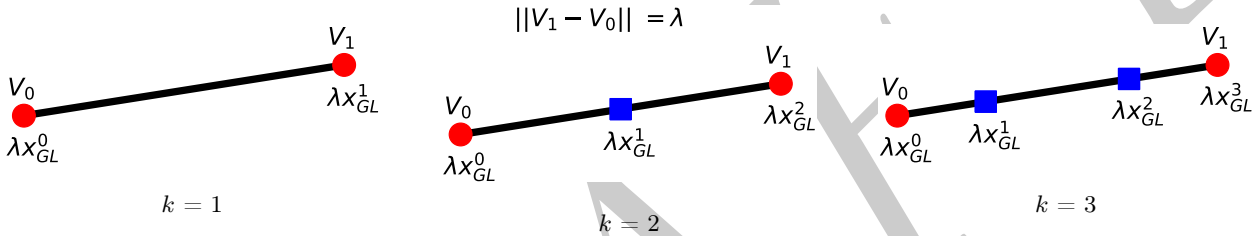


Fig. 6: 1D element $[\xi_0, \xi_0 + \lambda]$ representation for different orders k, with ● : Summits dofs, ■ : Edges dofs.

In this way, we can express the local edge matrix for all $E \in \mathcal{E}_h^{\mathrm{b,r}}$

$$\mathbf{R}_E = \left(\int_0^{\lambda} \alpha_*(\xi_0 + \xi) l_i(\xi) l_j(\xi)\, d\xi \right)_{0 \le i,j \le k} \tag{37}$$

with l_i, l_j polynomial test functions of order k, $\alpha_*(\xi_0 + \xi) = \alpha(V_0 + \xi \vec{t})$ the 1D restriction of α and $\vec{t}$ the tangential unit vector (from V_0 to V_1). We can easily deduce the explicit form of l_i because we have $\forall\, i, j \in [\![0, k]\!]$,

$$l_i(\lambda\, x_{GL}^j) = \delta_j^i, \tag{38}$$

with x_{GL}^j the $j-$th Gauss-Lobatto quadrature point on [0,1] (see Fig. 6). We can therefore deduce from the Lagrange polynomials:

$$l_i(\xi) = \sum_{j=0}^{k} \delta_j^i \left(\prod_{m=0, m\neq j}^{k} \frac{\xi - \lambda x_{\mathrm{GL}}^m}{\lambda x_{\mathrm{GL}}^j - \lambda x_{\mathrm{GL}}^m} \right) = \frac{1}{\lambda^k} \prod_{m=0, m\neq i}^{k} \frac{\xi - \lambda x_{\mathrm{GL}}^m}{x_{\mathrm{GL}}^i - x_{\mathrm{GL}}^m}.$$

For example, for $k = 1$, $l_0(\xi) = \frac{\lambda - \xi}{\lambda}$ and $l_1(\xi) = \frac{\xi}{\lambda}$.

Now we can compute the local matrix $\mathbf{R}_E^h$ of size $(k+1,\ k+1)$ through Eq. (37). All we need to do is calculate the integrals using a quadrature method, such as Gauss-Lobatto, which was introduced in the section on virtual elements, section 3.
After that, $\mathbf{R}_h$ is assembled in the same way as conventional finite element assemblies. Except that here, instead of iterating over all the DOFs of all the elements in the mesh, we only iterate among the DOFs of the edges belonging to Γ_R.

Remark 4.1. *Let us give the a few more useful points:*

- *In the case where α is a constant function (which is the case for the Helmholtz equation), we can simplify the Eq. (37) using a change of variable to obtain the following local matrix equation Eq. (39) (see calculation in appendix A for more details).*

$$\mathbf{R}_E = \left(\int_0^{\lambda} \alpha_*(\xi_0 + \xi) l_i(\xi) l_j(\xi)\, d\xi \right)_{0 \le i,j \le k} = \alpha\, \lambda \left(\int_0^1 \tilde{l_i}(\xi) \tilde{l_j}(\xi)\, d\xi \right)_{0 \le i,j \le k}, \tag{39}$$

with $\tilde{l_i}$ the polynomials for a unit element $[\xi_0, \xi_0 + 1]$. So all we need to do is evaluate the integral equation Eq. (39) once to obtain all the local matrices of the boundary segments. Moreover, this integral deals with a polynomial of degree $2k$, which can be evaluated exactly using a quadrature method with $k + 2$ Gauss-Lobatto points.

- *In the case where the Robin boundary condition is inhomogeneous, the approach is similar. We express the matrix analogously to the matrix in* (37).

$$\mathbf{L} = \int_{\Gamma_R} \beta(x, y) \Phi_i(x, y). \tag{40}$$

with $\beta \colon \mathbb{R}^2 \to \mathbb{R}$. Then we express the local matrix always on the segments characterized by $[\xi_0, \xi_0 + \delta]$,

$$\mathbf{L}_E = \left(\int_0^{\lambda} \beta_*(\xi_0 + \xi) l_i(\xi)\, d\xi \right)_{1 \le i \le k+1}, \tag{41}$$

with l_i a polynomial test function of order k, $\beta_*(\xi_0 + \xi) = \beta(V_0 + \xi\, \vec{t})$ the 1D restriction of β on Γ_R and $\vec{t}$ the tangential unit vector (from V_0 to V_1). Here, we calculate and assemble this matrix as before. Moreover, we can always simplify the calculation of this matrix if β is a constant function, then

$$\mathbf{L}_E = \lambda \beta \left(\int_0^1 \tilde{l_i}(\xi)\, d\xi \right)_{1 \le i \le k+1}. \tag{42}$$

5 Results and Applications

In this section, we will present the numerical results obtained thanks to the virtual element schemes. First, we performed a validation of the scheme on problems with manufactured solutions. Then a more realistic test case is performed.

5.1 Test case with analytical solutions

Let us considered a domain $\Omega := [0, 1] \times [0, 1]$ where we solve the Eq. (43) with Dirichlet and Robin's boundary conditions Eq. (43).

$$\begin{cases} \Delta u + \kappa^2\, u = f(x, y) & , \quad \text{in } \Omega\,, \\ u = u_{\text{exact}} & , \quad \text{on } \Gamma_2 \cup \Gamma_3 \cup \Gamma_4\,, \\ \dfrac{\partial u}{\partial n} + i\, \kappa\, u = g(x, y) & , \quad \text{on } \Gamma_1, \end{cases} \qquad \begin{array}{ccc} & \Gamma_3 & \\ \Gamma_4 & \Omega & \Gamma_2 \\ & \Gamma_1 & \end{array} \tag{43}$$

with the manufactured solution given by,

$$\begin{aligned} u_{\text{exact}}(x, y) &= (x + y) \cdot (1 + xi) + \exp(x^2 + i\, y^2), \\ f(x, y) &= -((2x)^2 + (2\, i\, y)^2 + 2(1 + i)) \cdot \exp(x^2 + i\, y^2) + \kappa^2 \cdot u_{\text{exact}}(x, y), \\ g(x, y) &= (1 + i) + (2\, i\, y) \cdot \exp(x^2 + i\, y^2) + i\, \kappa \cdot u_{\text{exact}}(x, y). \end{aligned} \tag{44}$$

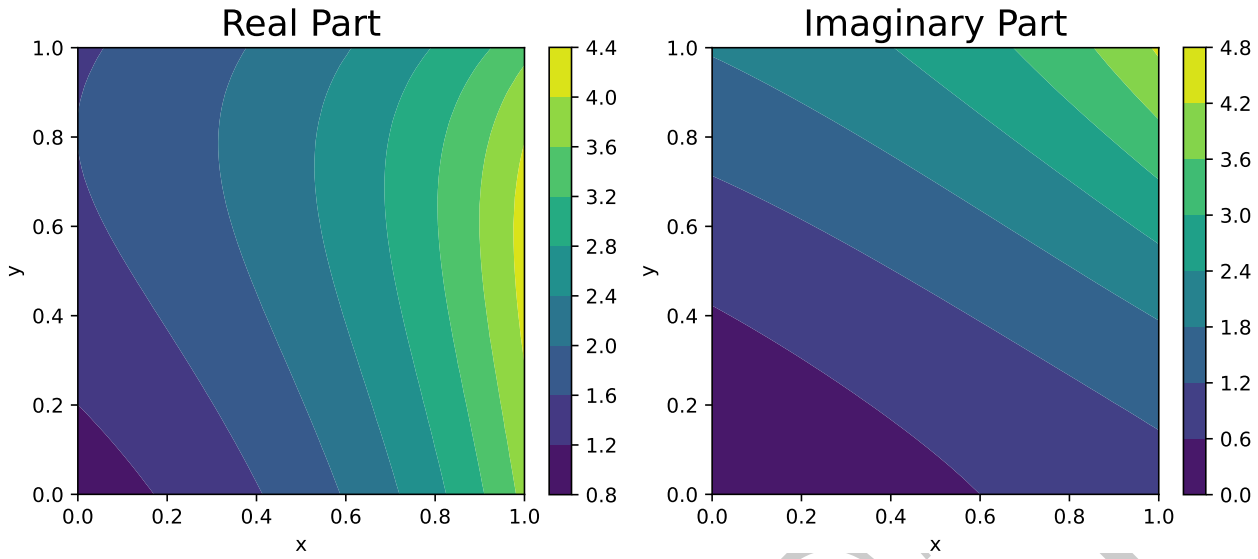


Fig. 7: Real and Imaginary part of u_{exact}.

and represented by the Fig. 7.
For this analytical case, we take the geometry of a unit square and link it with regular triangles, irregular triangles, irregular quadrilaterals and polygons. To generate these meshes, we use the Gmsh [18] and PyPolyMesher [1, 29]. We perform calculations from order 1 to order 5 on maximum cell diameters h from 0.05 to 0.7 m. We then compute the L^2 error for each calculation. The results are shown in Fig. 8.
We find the expected rate of convergence $\mathcal{O}(h^{k+1})$.

5.2 Relevance of Robin boundary conditions

To show the interest of a Robin boundary condition in our problem, we look at the problem represented by the Fig. 9. In this problem, we want to calculate the amplitude of reflected waves around an island. Thus, we have an incident wave arriving at 0 ° with an amplitude of 2 m and a period of 20 s. This wave is reflected on the boundary island Γ_D. On the other hand, the wave must be able to leave the domain freely via the boundary Γ_{inf}.
The reflected wave is calculated by the following [21] equation and the wave leaving condition at infinity Γ_{inf} will be studied for a Robin (left) and zero Neumann (right) condition.

$$\begin{cases} \Delta u + \kappa^2\, u = 0 & , \quad \text{in } \Omega\,, \\ u = -u_{\text{inc}} & , \quad \text{on } \Gamma_{\text{D}}\,, \\ \dfrac{\partial u}{\partial n} + i\,\kappa\, u = 0 & , \quad \text{on } \Gamma_{\text{Inf}}. \end{cases} \quad \text{or} \quad \begin{cases} \Delta u + \kappa^2\, u = 0 & , \quad \text{in } \Omega\,, \\ u = -u_{\text{inc}} & , \quad \text{on } \Gamma_{\text{D}}\,, \\ \dfrac{\partial u}{\partial n} = 0 & , \quad \text{on } \Gamma_{\text{Inf}}. \end{cases}$$

The results of this study are shown in the Fig. 10.
The reflected fields in the Fig. 10 are significantly different depending on the two different boundary conditions.

5.3 Application Case: Wave Field Calculation in Port of Cherbourg

In this section, we apply the solution of the Helmholtz and Berkhoff equations to a coastal engineering problem. We take the case of the port of Cherbourg in France and calculate the associated wave fields under certain conditions. First, we select our study site, as shown in the Fig. 11 (left). Next, we break down the contour into 3 different boundaries (Fig. 11 (center)): $\Gamma_{\mathbf{in}}$ the harbour entrance, $\Gamma_{\mathbf{out}}$ the harbour exit and $\Gamma_{\mathbf{D}}$ the port walls. Finally, we assign the correct boundary condition to these edges (Fig. 11 (right)).

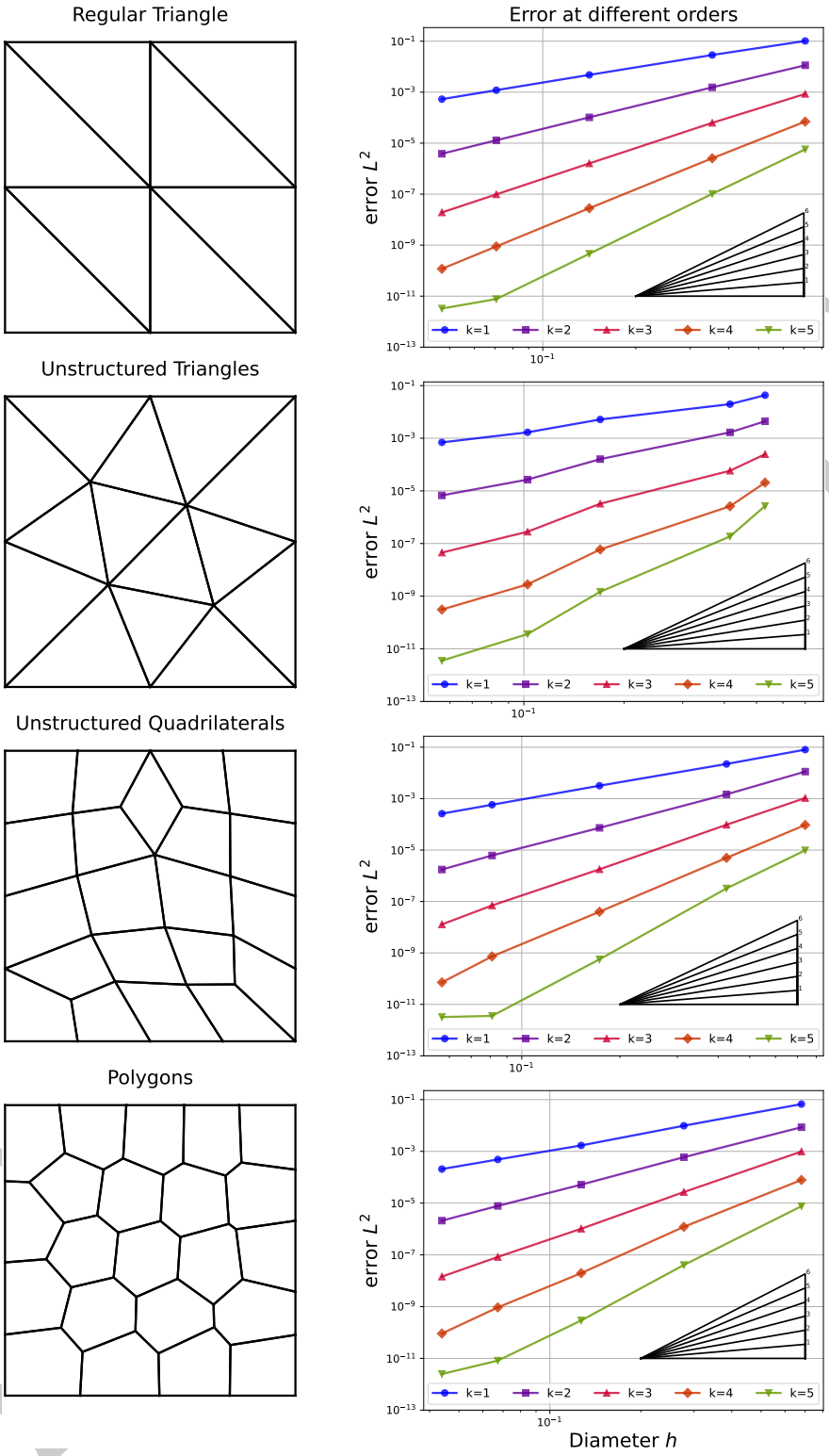


Fig. 8: Convergence of order $\mathcal{O}(h^{k+1})$.

The $\Gamma_{\mathbf{in}}$ boundary condition is modeled by an inhomogeneous Dirichlet condition taking the incident field as argument. The $\Gamma_{\mathbf{out}}$ boundary condition is modeled by a Robin condition allowing the wave to exit without disturbing other wave fields. More information on this condition in section 5.2. The $\Gamma_{\mathbf{D}}$ boundary condition is modeled by an inhomogeneous Dirichlet condition with a reflection coefficient γ. First, we'll look at the

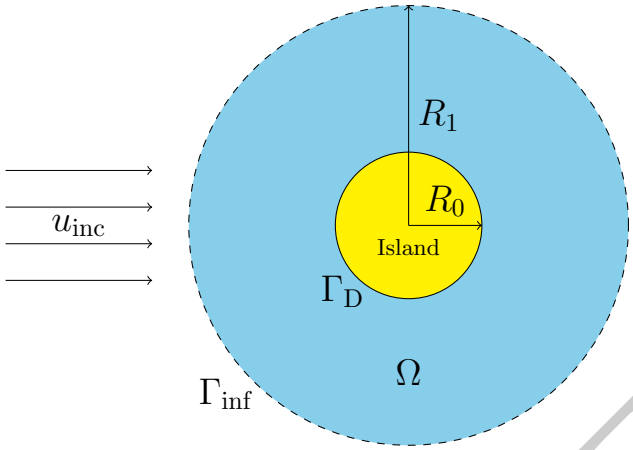


Fig. 9: Sketch of the island reflection problem.

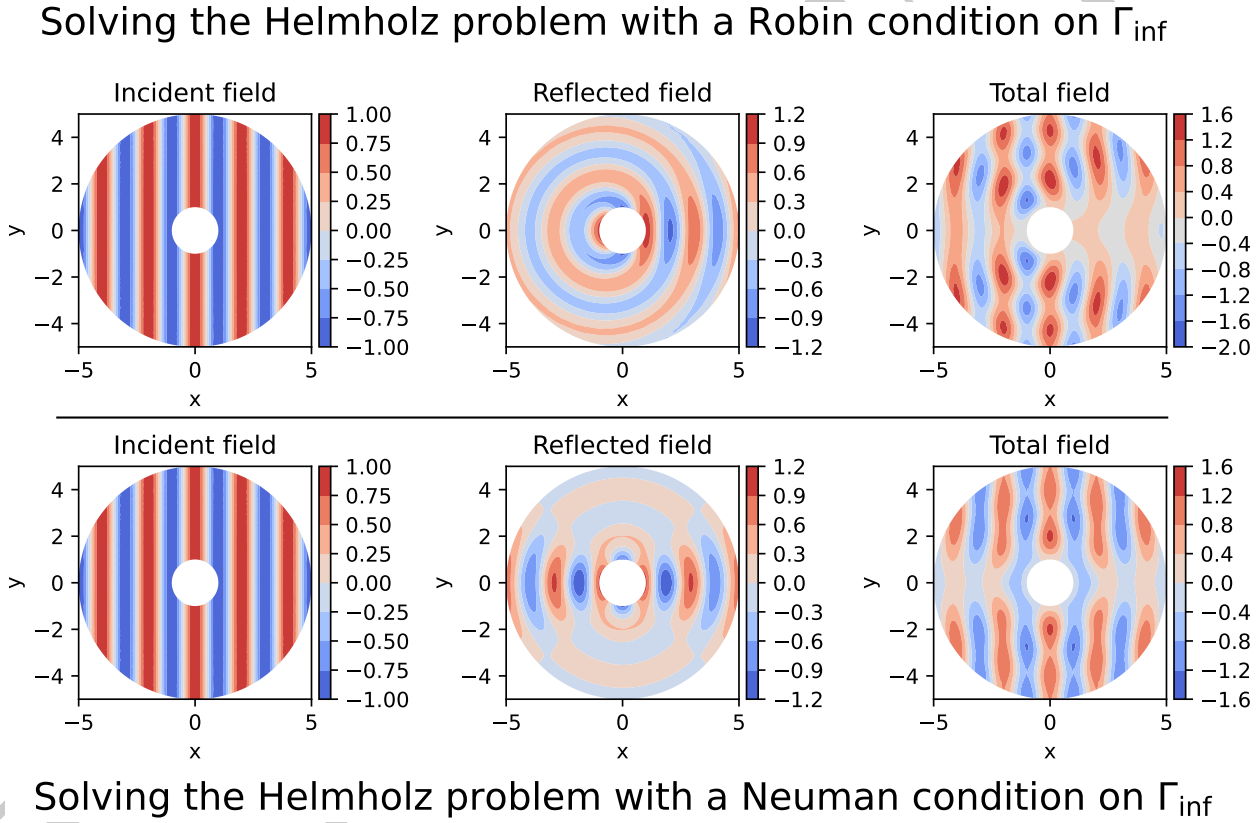


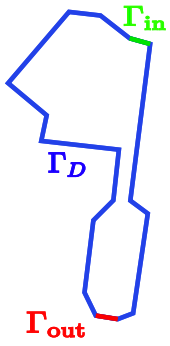
Fig. 10: Comparison of the results obtained on the island problem by solving the Helmholtz equation with a Robin condition (top) and a zero Neumann condition (bottom).

importance of this reflection coefficient in the section 5.3.1. Finally, we will compare the results with different orders of the virtual element method, in section 5.3.2.

5.3.1 Sensitivity of the γ reflection coefficient

In this section, we look at the influence of the harbor wall reflection coefficient γ on wave fields. We compare reflected and total wave fields for two different reflection coefficients, $\gamma = 1$ (Fig. 12 (top)) and $\gamma = 0.5$ (Fig. 12 (bottom)). For this study, we generate an incident wave field entering the harbour at 280 ° with a maximum amplitude $u_{\max} = 1$ m and a wave period $T_0 = 8$ s. This incident field can be seen in the Fig.

Port location



Port boundary

The Helmholtz equation:

$$\begin{cases} \Delta u + \kappa^2 u = 0, & \text{in } \Omega\,, \\ u = u_I, & \text{in } \Gamma_{in}\,, \\ \dfrac{\partial u}{\partial n} + i\kappa\, u = 0, & \text{in } \Gamma_{out}\,, \\ u = -\gamma\, u_I, & \text{in } \Gamma_D\,. \end{cases}$$

Fig. 11: Configuration of our study of the port of Cherbourg

12 (left). The results of this study are shown in the Fig. 12 with i) on the left, the incident field ii) in the middle, the reflected field (solution of the Helmholtz equation) iii) on the right, the total field.

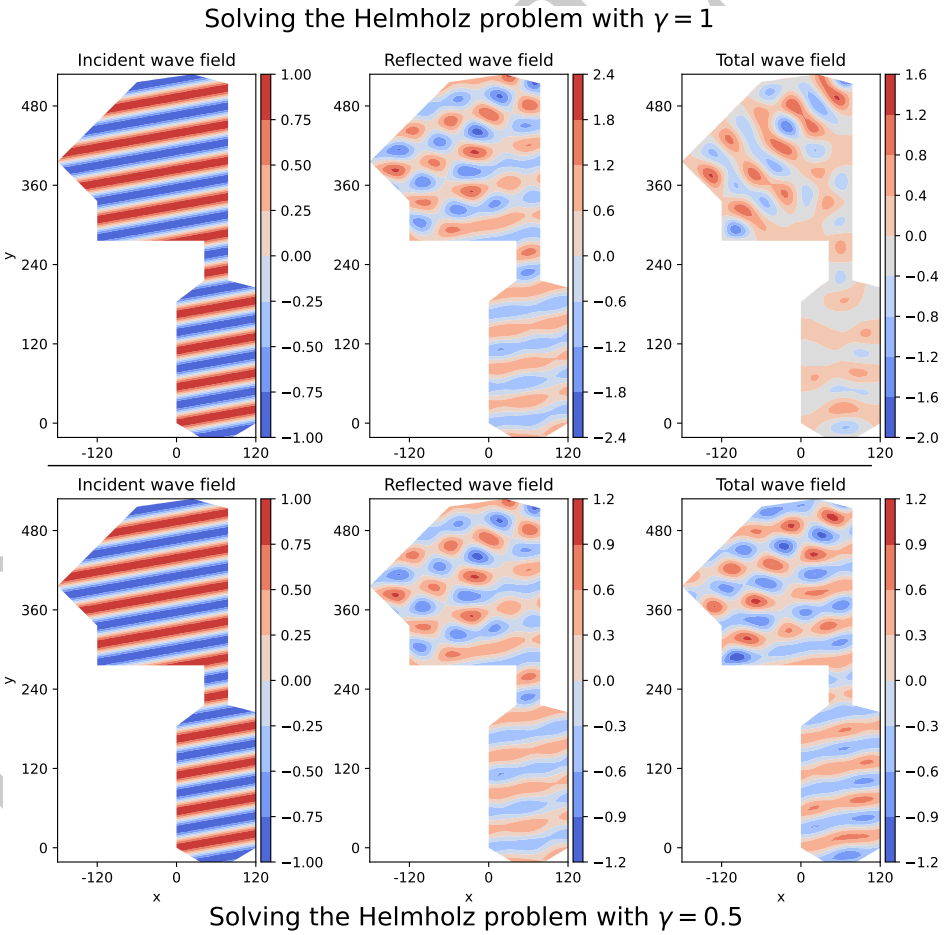


Fig. 12: Comparison of wave fields for the two reflection coefficients $\gamma = 1$ (top) and $\gamma = 0.5$ (bottom). Problem condition: $\alpha = 280$ °, $u_{\max} = 1$ m and $T_0 = 8$ s.

The results in the Fig. 12 show that by halving the reflection coefficient γ, the reflected wave field is also halved.

5.3.2 Sensitivity to order of resolution

In this section, we look at the influence of the order of solution of the virtual element method on the solution of the Helmholtz problem. We compare the reflected and total wave fields for two orders of resolution with fairly coarse mesh (Fig. 13), order 1 (Fig. 13 top) with 81 degrees of freedom and order 5 (Fig. 13 bottom) with 881 degrees of freedom. For this study, we generate an incident wave field entering the harbour at 250 ° with a maximum amplitude $u_{\max} = 1$ m and a wave period $T_0 = 8$ s. The results of this study are shown in Fig. 13 with i) on the left, the incident field ii) in the middle, the reflected field (solution of the Helmholtz equation) iii) on the right, the total field.

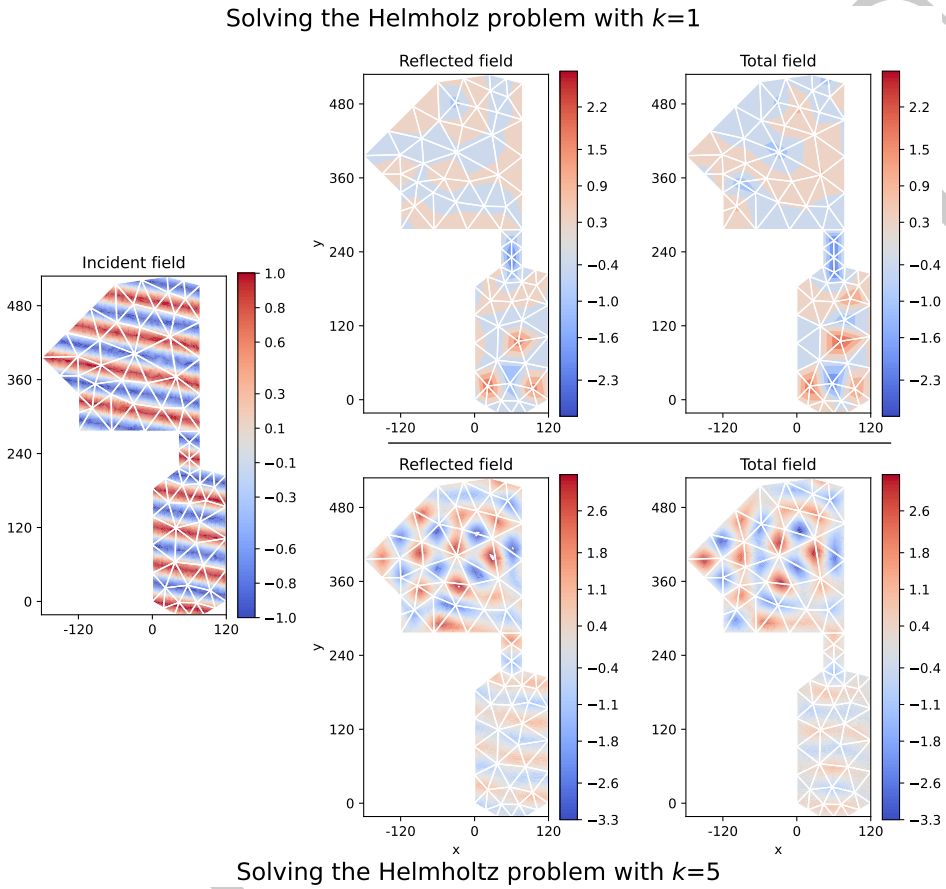


Fig. 13: Comparison of wave fields for the two order of resolution $k = 1$ (top) and $k = 5$ (bottom). Problem condition: $\alpha = 250$ °, $u_{\max} = 1$ m and $T_0 = 8$ s.

Unsurprisingly, the results in the Fig. 13 show that results in order 5 are more accurate than those in order 1.

6 Discussion

Convergence tests of our model on the Helmholtz equation showed its superconvergence (Fig. 8). We modeled the port of Cherbourg, taking into account boundary conditions. The choice of a Robin condition proved relevant for the outflow condition, as shown in the Fig. 10 where it's clear that the wave reflected under Robin's condition (top) can leave freely, while the wave reflected under Neumann's condition (bottom) seems to be disturbed under this condition. A robin edge condition offers a clear advantage in terms of computation time, compared with perfectly matched layer (PML) conditions [27], which also allow the wave to pass, but require an additional computation on an extension of the domain.
Next, we have seen that the reflection coefficient of the walls plays a major role in the model results. Indeed,

the results in the Fig. 12 showed a totally different total wave field (incident + reflected), so that the harbour's eigenmodes are no longer located in exactly the same places for the two configurations. It is therefore very important to calibrate this reflection condition correctly.
Finally, we have seen that order can also play a major role in the accuracy of results, as shown by the Fig. 13. Indeed, we note that with order 1, it's very difficult to capture the port's eigenmodes, whereas with order 5, the port's eigenmodes are distinguishable.
Many precautions need to be taken to model the port as accurately as possible [10]. It is also possible to go further in the modelling by taking into account a variable bottom (which is not the case for the Helmholtz model). To do this, at lower cost, we can solve the following Mild-Slope equation Eq. (B1), taking into account bottom variability.

$$\begin{cases} \nabla(C_p C_g \nabla u) + \kappa^2 C_p C_g \, u = 0, & \text{in } \Omega \,, \\ u = u_I, & \text{in } \Gamma_{\text{in}} \,, \\ \dfrac{\partial u}{\partial n} + i\kappa \, u = 0, & \text{in } \Gamma_{\text{out}} \,, \\ u = -\gamma \, u_I & \text{in } \Gamma_{\text{D}} \,. \end{cases} \tag{45}$$

with

$$C_p = \frac{\omega}{\kappa} \quad \text{and} \quad C_g = \frac{1}{2} C_p \left[1 + \kappa h \frac{1 - \tanh^2(\kappa h)}{\tanh(\kappa h)} \right]. \tag{46}$$

The choice of boundary conditions has been explained in the application section 5.
To approximate this equation easily, simply consider $C_p C_g$ and $\kappa^2 C_p C_g$ constants per cell. Details of the variational formulation are given in appendix B.
The influence of a variable bottom on the calculation of wave fields can be seen directly in the following example. We compare a simulation with a flat bottom at a depth of 5 m (Fig. 14 top left) using the Helmholtz model, with a linear bottom (Fig. 14 bottom left) using the Berkhoff model. For this study, we generate an incident wave field entering the harbour at 280 ° with a maximum amplitude $u_{\max} = 2$ m and a wave period $T_0 = 8$ s. To make the modelling more realistic, [25] breaking wave criterion is added. This decreases wave amplitude linearly with depth. This incident field can be seen in figure 14. The results of this study are shown in the Fig. 14 from left to right: i) the depth ii) the incident field iii) the reflected field iv) the total field.
The results in the Fig. 14 show that the lack of depth limits the formation of eigenmodes. In fact, in the flat-bottom simulation (top), eigenmodes are formed in the upper and lower parts of the harbor; whereas in the linear-bottom simulation (bottom), eigenmodes are no longer formed where there is almost no water: in the lower part of the harbor.

Remark 6.1. *For this model, the accuracy will be less good than Helmholtz's due to the approximation made by cells.*

7 Conclusion

In this article, we have addressed a coastal engineering problem using the virtual element formalism. This study has highlighted a formalism that is more precise than traditional formalisms, and above all, can handle complex meshes, enabling eigenmodes to be targeted on a given geometry. Thanks to a description of the implementation of the virtual elements, we have been able to provide a guide line for dealing with these wave problems. In addition, the implementation of a high-order Robin condition is something that is rarely mentioned in the literature. The results of the application on the port of Cherbourg showed that it is essential to choose the right parameters, such as the reflection coefficient of the walls and the order of calculation of the virtual element method. This study could be useful to coastal engineers interested in numerical mathematics, wishing to tackle development problems on coastal structures as was done in [10].

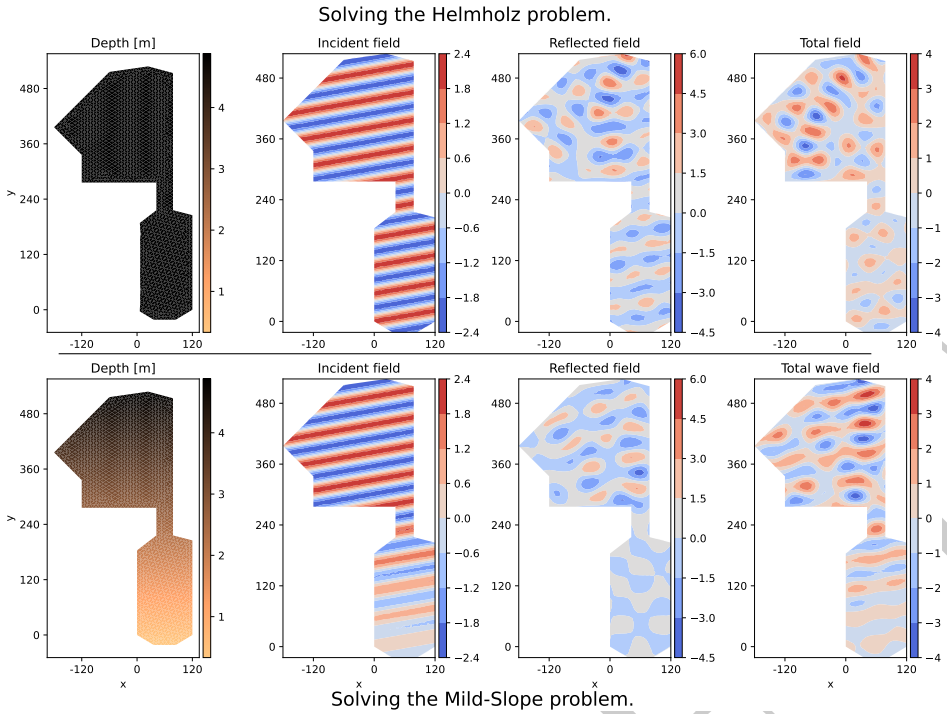


Fig. 14: Comparison of wave fields for two different sea bottom: a flat bottom (top) and a linear bottom (bottom). Problem condition: $\alpha = 280$ °, $u_{\max} = 2$ m and $T_0 = 8$ s.

CRediT authorship contribution statement

R. Dupont: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Conceptualization. **M. Dauphin**: Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation. **R. Mottier**: Writing – review & editing, Validation, Supervision, Methodology.

Declaration of competing interest

The authors declare that they have no conflict of interest.

Data availability

All data, models, and code generated or used during the study are available on request.

Acknowledgements

This work was conducted as part as M. Dupont's PhD studies which is funded by the CNRS with the MITI grant. We gratefully acknowledge funding from CNRS, OPTIBEACH projects and FEDER Europe.

Appendix

A Calculation details of Eq. (39)

The following provides the detailed calculation for the expression derived in Eq. (39).

$$
\begin{aligned}
\mathbf{R}_e^h =& (\int_0^\lambda \alpha_*(\xi_0+\xi) l_i(\xi) l_j(\xi)\, d\xi)_{0\le i,j\le k}\,, \\
\underset{\alpha_*=\alpha=cte}{=}& (\frac{\alpha}{\lambda^{2k}} \int_0^\lambda \left[\prod_{m=0,m\neq j}^{k} \frac{\xi-\lambda x_{\mathrm{GL}}^m}{x_{\mathrm{GL}}^i - x_{\mathrm{GL}}^m} \prod_{m=0,m\neq j}^{k} \frac{\xi-\lambda x_{\mathrm{GL}}^m}{x_{\mathrm{GL}}^j - x_{\mathrm{GL}}^m}\right] d\xi)_{0\le i,j\le k}\,, \\
\underset{\substack{\xi=\lambda\xi' \\ d\xi=\lambda d\xi'}}{=}& (\frac{\alpha}{\lambda^{2k}} \int_0^1 \left[\prod_{m=0,m\neq j}^{k} \frac{\lambda\xi'-\lambda x_{\mathrm{GL}}^m}{x_{\mathrm{GL}}^i - x_{\mathrm{GL}}^m} \prod_{m=0,m\neq j}^{k} \frac{\lambda\xi'-\lambda x_{\mathrm{GL}}^m}{x_{\mathrm{GL}}^j - x_{\mathrm{GL}}^m}\right] \lambda d\xi')_{0\le i,j\le k}\,, \\
=& (\frac{\alpha}{\lambda^{2k}} \int_0^1 \left[\prod_{m=0,m\neq j}^{k} \lambda\frac{\xi'-x_{\mathrm{GL}}^m}{x_{\mathrm{GL}}^i - x_{\mathrm{GL}}^m} \prod_{m=0,m\neq j}^{k} \lambda\frac{\xi'-x_{\mathrm{GL}}^m}{x_{\mathrm{GL}}^j - x_{\mathrm{GL}}^m}\right] \lambda d\xi')_{0\le i,j\le k}\,, \\
=& (\alpha\lambda \int_0^1 \left[\prod_{m=0,m\neq j}^{k} \frac{\xi'-x_{\mathrm{GL}}^m}{x_{\mathrm{GL}}^i - x_{\mathrm{GL}}^m} \prod_{m=0,m\neq j}^{k} \frac{\xi'-x_{\mathrm{GL}}^m}{x_{\mathrm{GL}}^j - x_{\mathrm{GL}}^m}\right] d\xi)_{0\le i,j\le k}\,, \\
=& \alpha\,\lambda (\int_0^1 \tilde{l}_j(\xi)\tilde{l}_i(\xi)\, d\xi)_{0\le i,j\le k}\,.
\end{aligned}
\tag{A1}
$$

B Solving the Mild-Slope equation

The amplitude of the reflected wave u_R can also be obtained by solving the Berkhoff equation [6], in the case of a variable bottom,

$$
\begin{cases}
\nabla(C_p C_g \nabla u) + \kappa^2 C_p C_g\, u = 0, & \text{in } \Omega\,, \\
u = u_I, & \text{in } \Gamma_{\text{in}}\,, \\
\dfrac{\partial u}{\partial n} + i\kappa\, u = 0, & \text{in } \Gamma_{\text{out}}\,, \\
u = -\gamma\, u_I & \text{in } \Gamma_{\text{D}}\,.
\end{cases}
\tag{B1}
$$

with

$$
C_p = \frac{\omega}{\kappa} \quad \text{and} \quad C_g = \frac{1}{2}\, C_p \left[1 + \kappa h\, \frac{1-\tanh^2(\kappa h)}{\tanh(\kappa h)}\right]. \tag{B2}
$$

The choice of boundary conditions has been explained in the application section 5.

Remark: In practice, κ is obtained simply by using the [20] approximation.

Now, we consider the [6] equation Eq. (B1) and thus the following variational formulation:

$$
\begin{cases}
\text{find } v \in V = H_0^1(\Omega) \text{ such that} \\
\qquad A(v,w) \;=\; 0 \quad \forall w \in V,
\end{cases}
\tag{B3}
$$

where,

$$
A(u,v) = \int_\Omega \nabla(C_p C_g \nabla v\, w) + \int_\Omega \kappa^2 C_p C_g\, v\, w \quad . \tag{B4}
$$

Discrete Mild-Slope Problem.

The discrete problem read as follow. Find $u_h \in V_h \subset V$ such that

$$\text{find } u_h \in V_h \subset V \text{ such that} A_h(u_h, v_h) = 0 \quad \forall v_h \in V_h, \tag{B5}$$

where $V_h \subset V$ is a finite dimensional space and $A_h(\,\cdot\,,\,\cdot\,) : V_h \times V_h \to \mathbb{R}$ is a discrete bilinear form approximating the continuous form $A(\,\cdot\,,\,\cdot\,)$.
We thus have the discrete form:

$$\begin{aligned}
A_h(v_h, w_h) &= \sum_{T\in\mathcal{T}_h} \left[\int_T \nabla(C_p C_g \nabla v_h w_h) + \int_T k^2 C_p C_g v_h w_h \right], \\
&\underset{\substack{1/|T|\int_T C_p C_g = \mathcal{A}_T \\ 1/|T|\int_T \kappa^2 C_p C_g = \mathcal{B}_E}}{\approx} \sum_{T\in\mathcal{T}_h} \left[\mathcal{A}_T \int_T (\Delta v_h\, w_h) + \mathcal{B}_T \int_T v_h w_h \right], \\
&\underset{\substack{\text{green} \\ \partial v/\partial n = -iku}}{=} \sum_{T\in\Omega_h} \left[-\mathcal{A}_T \int_T \nabla v_h \nabla w_h + \mathcal{B}_T \int_T v_h w_h - \mathbb{1}_{\Gamma_{\text{out}}\subset T}\, i\mathcal{A}_T \int_{\Gamma_{\text{out}}} \kappa\, v_h w_h \right], \\
&= a_h(v_h, w_h) + r_h(v_h, w_h)\,.
\end{aligned} \tag{B6}$$

with a_h and r_h the discrete forms of a and r: defined in the same way as above, $\mathbb{1}_{\Gamma_{\text{out}}\subset E}$ the indicator function and $u = u + u_D$ and u_D is the lifting of $-\gamma u_I$ or u_I (depending on the border).

Remark: Unlike the discrete formulation of the homogeneous Helmholtz equation [21] (see section 3.3), the discrete formulation of the Berkhoff equation Eq. (B6) assumes that κ^2 and $C_p C_g$ are constant for each cell in the mesh.

References

[1] S. S. Abedi-Shahri. pypolymesher: Generation of polygonal mesh, 2024.

[2] B. Ahmad, A. Alsaedi, F. Brezzi, L. D. Marini, and A. Russo. Equivalent projectors for virtual element methods. *Computers & Mathematics with Applications*, 66(3):376–391, 2013.

[3] G. B. Airy. *Tides and waves*. B. Fellowes, 1845.

[4] P. F. Antonietti, P. Houston, and G. Pennesi. Fast numerical integration on polytopic meshes with applications to discontinuous galerkin finite element methods. *Journal of Scientific Computing*, 77(3):1339–1370, 2018.

[5] L. Beirão da Veiga, F. Brezzi, L. D. Marini, and A. Russo. The hitchhiker's guide to the virtual element method. *Mathematical Models and Methods in Applied Sciences*, 24(08):1541–1573, 2014.

[6] J. Berkhoff. COMPUTATION OF COMBINED REFRACTION - DIFFRACTION. *Coastal Engineering Proceedings*, 1(13):23, Jan. 1972.

[7] N. Booij. A note on the accuracy of the mild-slope equation. *Coastal Engineering*, 7(3):191–203, 1983.

[8] F. Brezzi, A. Cangiani, G. Manzini, L. Marini, A. Russo, et al. Basic principles of virtual element methods. *Mathematical Models and Methods in Applied Sciences*, 23(1):199–214, 2013.

[9] P. G. Ciarlet. *The finite element method for elliptic problems*. SIAM, 2002.

[10] M. Cook, F. Bouchette, B. Mohammadi, L. Sprunck, and N. Fraysse. Optimal Port Design Minimizing Standing Waves with A Posteriori Long Term Shoreline Sustainability Analysis. *China Ocean Engineering*, 35(6):802–813, Dec. 2021.

[11] L. B. da Veiga, K. Lipnikov, and G. Manzini. *The mimetic finite difference method for elliptic problems*, volume 11. Springer, 2014.

[12] L. B. da Veiga, C. Lovadina, and G. Vacca. Divergence free virtual elements for the stokes problem on polygonal meshes. *ESAIM: Mathematical Modelling and Numerical Analysis*, 51(2):509–535, 2017.

[13] F. Dassi. Vem++, a c++ library to handle and play with the virtual element method. *arXiv preprint arXiv:2310.05748*, 2023.

[14] F. Dassi and G. Vacca. Bricks for the mixed high-order virtual element method: Projectors and differential operators. *Applied Numerical Mathematics*, 155:140–159, 2020.

[15] A. Dedner and A. Hodson. A framework for implementing general virtual element spaces. *SIAM Journal on Scientific Computing*, 46(3):B229–B253, 2024.

[16] D. A. Di Pietro and J. Droniou. *The Hybrid High-Order Method for Polytopal Meshes*. Modeling, Simulation and Applications series. Springer International Publishing, Mar. 2020.

[17] A. Feldmeier. *Theoretical fluid dynamics*. Springer, 2019.

[18] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11):1309–1331, 2009.

[19] J.-L. Guermond and A. Ern. Theory and practice of finite elements. *Applied Mathematical Sciences*, 159, 2004.

[20] J. Guo. Simple and explicit solution of wave dispersion equation. *Coastal Engineering*, 45(2):71–74, 2002.

[21] P. Helmholtz. Xliii. on discontinuous movements of fluids. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 36(244):337–346, 1868.

[22] C. Herrera, R. Corrales-Barquero, J. Arroyo-Esquivel, and J. G. Calvo. A numerical implementation for the high-order 2d virtual element method in matlab. *Numerical Algorithms*, 92(3):1707–1721, 2023.

[23] J. T. Kirby and S. K. Misra. *A note on the modified mild-slope equation*. University of Delaware, Ocean Engineering Laboratory, Center for Applied Coastal Research, 1998.

[24] L. Mascotto, I. Perugia, and A. Pichler. A nonconforming trefftz virtual element method for the helmholtz problem. *Mathematical Models and Methods in Applied Sciences*, 29(09):1619–1656, 2019.

[25] W. Munk. The solitary wave theory and its application to surf problems. *Annals of the New York Academy of Sciences*, 51:376 – 424, 12 1949.

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

[27] I. Singer and E. Turkel. A perfectly matched layer for the helmholtz equation in a semi-infinite strip. *Journal of Computational Physics*, 201(2):439–465, 2004.

[28] O. J. Sutton. The virtual element method in 50 lines of matlab. *Numerical Algorithms*, 75:1141–1159, 2017.

[29] C. Talischi, G. H. Paulino, A. Pereira, and I. F. Menezes. Polymesher: a general-purpose mesh generator for polygonal elements written in matlab. *Structural and Multidisciplinary Optimization*, 45:309–328, 2012.