HAL
open science

High-order discrete fourier transform for the solution of the Poisson equation

Federica Caforio, Sébastien Imperiale

To cite this version:

Federica Caforio, Sébastien Imperiale. High-order discrete fourier transform for the solution of the Poisson equation. SIAM Journal on Scientific Computing, 2019, 41 (5), pp.A2747-A2771. 10.1137/18M1225410 . hal-01914257v2

HAL Id: hal-01914257

https://inria.hal.science/hal-01914257v2

Submitted on 11 Sep 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A HIGH-ORDER SPECTRAL ELEMENT FAST FOURIER TRANSFORM FOR THE POISSON EQUATION *

F. CAFORIO † AND S. IMPERIALE ‡

Abstract. The aim of this work is to propose a novel, fast solver for the Poisson problem discretised with High-Order Spectral Element Methods (HO-SEM) in canonical geometries (rectangle in 2D, rectangular parallelepiped in 3D). This method is based on the use of the Discrete Fourier Transform to reduce the problem to the inversion of the symbol of the operator in the frequency space. The proposed solver is endowed with several properties. First, it preserves the efficiency of the standard FFT algorithm; then, the matrix storage is drastically reduced (in particular, it is independent of the space dimension); a pseudo-explicit Singular Value Decomposition (SVD) is used for the inversion of the symbols; finally, it can be extended to non-periodic boundary conditions. Furthermore, due to the underlying HO-SEM discretisation, the multi-dimensional symbol of the operator can be efficiently computed from the one-dimensional symbol by tensorisation.

Key word. Fast Fourier transform, high-order finite elements, Poisson's equation

AMS subject classifications. 65T99, 65N30

1. Introduction. The objective of this work is to provide a fast method to solve the Poisson partial differential equation (PDE) in bounded domains. This topic has already been largely addressed in the literature (see [14, 21] or more recently [3, 17]). However, to our best knowledge, there is no efficient algorithm that is adequate for the specific applications that we target. In more detail, we require a fast method that is compatible (in a sense that will be specified in what follows) with high-order Finite Element (FE) discretisations. The main application is the treatment of the incompressibility condition for transient elastodynamic (see [9]) or Navier-Stokes PDEs (see [9, 13]). In these PDEs, although parameters may be heterogeneous and/or anisotropic, the treatment of the incompressibility constraint can be reduced to the solution of a Poisson equation for a scalar unknown, i.e. the pressure, thanks to penalisation techniques.

Advanced methods involve the solution of these PDEs (Stokes or elastodynamics) with High-Order (HO) FE methods. However, the convergence properties of these formulations are governed by stability considerations involving ellipticity requirements and the famous Ladyzenskaya - Babŭska - Brezzi (LBB) inf-sup condition [5]. If this stability condition is not satisfied, a loss of accuracy can be observed (we refer to [6, 7] for more detail). For this reason, it can be shown that the pressure must be solved in an suitable HO - FE space.

A well-known FE method for Stokes flow and transient elastodynamic problems is the so-called Spectral Element Method (SEM) (see [18] for Stokes equation and [12, 16] for elastodynamics). Two main assets of this type of FE methods are the optimal rate of convergence achieved and mass lumping. Therefore, they represent a privileged choice for the explicit discretisation of transient problems. Furthermore, they are constructed in one-dimension from Lagrangian basis functions supported on Gauss-Lobatto points and their extension to two- and three-dimensions is straightforward, since it is based on tensorisation.

*Submitted to the editors September 11, 2019.

†federica.caforio@inria.fr. Inria, Université Paris-Saclay, France. LMS, Ecole Polytechnique, CNRS, Université Paris-Saclay, France.

‡sebastien.imperiale@inria.fr. Inria, Université Paris-Saclay, France. LMS, Ecole Polytechnique, CNRS, Université Paris-Saclay, France.

Due to the aforementioned inf-sup condition, the choice of space discretisation for the main unknown restricts the choice of the auxiliary variable, i.e. the pressure. In more detail, it is shown in [6] that if $(R+1)$-order continuous SEM are used for the main unknown (see also [2] for a related analysis), and a regular cartesian mesh is considered, it is possible to use $R-$order continuous SEM for the pressure, still keeping the optimal order of convergence. In a recent paper [9] we have proposed a novel numerical scheme that is adapted to the approximation of elastic wave propagation in incompressible media, for application to elastography imaging of soft tissues [20]. In particular, the method relies on high-order spectral finite elements for space discretisation and an implicit/explicit, second-order, energy-preserving time discretisation. The proposed algorithm requires at each time step the resolution of a Poisson problem to account for the incompressibility constraint, that is imposed by penalisation techniques, and few matrix-vector multiplications for the explicit part of the scheme. In this paper we construct a fast solver for the Poisson problem discretised with HO-SEM on uniform meshes. The main peculiarity of this method is that it corresponds to the explicit computation of the inverse of the discrete Laplace operator (computed with HO-SEM), hence accuracy and convergence properties of the solver are directly inherited from those of the SEM.

The proposed method relies on the use of the Fast Fourier Transform (FFT) to reduce the problem to the inversion of the – frequency-dependent – symbol of the operator. When first-order finite elements are adopted, the symbol is scalar, its inversion is trivial and the solution is recovered using inverse FFT. This is the classical fast method presented in numerous textbooks, see for instance [14, 15]. However, when high-order FE are used, the symbol is not scalar anymore, and its inversion is no longer trivial. The definition of this symbol has been first introduced in [1, 11] for the discrete analysis of wave propagation phenomena in periodic domains but, surprisingly, it has never been used as an actual solver for the Poisson equation. The solver that we construct has the following algorithmic properties: first, it preserves the efficiency of standard FFT algorithm; then, the storage cost is independent of the dimension of the problem; moreover, a pseudo-explicit Singular Value Decomposition (SVD) is used for the inversion of the symbols; finally, it is extended to the treatment of Neumann and Dirichlet boundary conditions.

Note that, since the underlying space discretisation is based on SEM, the proposed method is suitable for application to canonical geometries, i.e. regular quadrilaterals or parallelepipeds. However, in the specific application that we target, this is not a restriction if the computational domain under study can be smoothly mapped to a uniform mesh.

This paper is organised as follows:

- Section 2 deals with the statement of the problem under study and some classical results for first-order discretisation. In more detail:
  - Section 2.1 contains the main features of the Poisson-like partial differential equation that we aim to solve and its numerical approximation by finite element methods;
  - in Section 2.2 we recall the standard fast method that can be used when first-order discretisation is considered, based on the classical FFT. We only detail the one-dimensional case for the sake of brevity.
- In Section 3 we describe a novel, fast method for the solution of the one-dimensional problem with high-order space approximation, named HOFFT. This method is derived as a generalisation of the standard FFT algorithm. In particular:
  - Section 3.1 deals with the derivation and the practical implementation of the

method to solve the problem when periodic boundary conditions (BCs) are imposed;
  - in Section 3.2 we detail the generalisation of the HOFFT solver to other BCs, e.g. Dirichlet and Neumann. This is based on a periodic and symmetric extension of the source term, that is efficiently performed taking into account the symmetry properties of the Discrete Fourier Transform.
- Section 4 is devoted to the extension of the HOFFT solver to multiple dimensions, based on tensorisation from the one-dimensional case. In particular, Section 4.1 concerns the resolution of the multi-dimensional problem with periodic BCs, whereas Section 4.2 deals with Dirichlet or Neumann boundary conditions.
- Numerical results are shown in Section 5, along with some convergence analysis results. Furthermore, we provide some remarks on the complexity of the solver – that is shown to be in $O(R^{d+1}\, N^d \log N)$ in d dimensions on meshes with N^d elements – and its parallelisability.
- Finally, in Section 6 we provide complementary elements on the applicability of the proposed solver.

2. Statement of the problem and standard results.

2.1. Variational formulation for the Poisson problem. Let us consider a generalisation of the Poisson equation such that

$$\rho\, u - \Delta u = f \quad \text{in} \quad \Omega = \prod_{i=1}^{d} [0, L_i] \ \subset \mathbb{R}^d, \tag{2.1}$$

with u and f scalar functions with values in $\mathbb{C}$, and $\rho \in \mathbb{C}$ a constant with non-zero imaginary part or non-negative real part. We do not specify the conditions imposed on the boundary of Ω for the moment. Given an admissible subspace $\mathcal{V} \subset H^1(\Omega)$, we want to solve the following variational formulation related to Eq. (2.1):
Let $f \in L^2(\Omega)$, find $u \in \mathcal{V}$ such that

$$\rho\, m(u, v) + a(u, v) = \ell(v), \quad \forall v \in \mathcal{V}, \tag{2.2}$$

where

$$m(u, v) = \int_\Omega u\, \overline{v}\, \mathrm{d}\Omega, \quad a(u, v) = \int_\Omega \underline{\nabla} u \cdot \overline{\underline{\nabla} v}\, \mathrm{d}\Omega, \quad \ell(v) = \int_\Omega f\, \overline{v}\, \mathrm{d}\Omega.$$

For spatial discretisation, we introduce a finite-dimensional subspace $\mathcal{V}_h \subset \mathcal{V}$ of continuous functions. The discrete problem reads: *find $u_h \in \mathcal{V}_h$ such that*

$$\rho\, m_h(u_h, v_h) + a_h(u_h, v_h) = \ell_h(v_h), \quad \forall\, v_h \in \mathcal{V}_h, \tag{2.3}$$

where

$$m_h(u_h, v_h) = \int_\Omega^Q u_h\, \overline{v_h}\, \mathrm{d}\Omega, \quad a_h(u_h, v_h) = \int_\Omega^Q \underline{\nabla} u_h \cdot \overline{\underline{\nabla} v_h}\, \mathrm{d}\Omega, \tag{2.4}$$

and where we define $f_h \in \mathcal{V}_h$ such that $\ell_h(v_h) = m_h(f_h, v_h)$. Note that the superscript Q on the integral signs denotes the approximation by quadrature rule. We recall that in one dimension, given a function g, an $(R_Q + 1)-$points quadrature rule is based on a suitable set of points $\{\eta_k\}$ in $[0, 1]$ and weights $\{\omega_k\}$ such that

$$\int_{[0,h]} g(x)\, \mathrm{d}x \approx \int_{[0,h]}^Q g(x)\, \mathrm{d}x := h \sum_{k=0}^{R_Q} g(\eta_k\, h)\, \omega_k. \tag{2.5}$$

We recall that, if Gauss-Lobatto nodes are used, this formula is exact for polynomials up to degree $2R_Q - 1$ (see for instance [19] for more detail). Fast methods can be used to solve (2.3), see for example the hierarchical matrix (H-matrix) method [4], that is an efficient algorithm to compute approximate inverse matrices based on data-sparse approximations of non-sparse matrices. Within the family of fast methods, the most efficient one relies on the use of the Fast Fourier Transform (FFT). The use of this method is, however, restricted to some assumptions on the computational domain – Ω is a rectangle in 2D, or a parallelepiped in 3D – and to the use of $\mathcal{Q}^1$ finite elements for constructing $\mathcal{V}_h$, i.e. the set of linear polynomials in each variable of space (see for example [14, 15]). The next section is devoted to the recall of this solver (in one dimension). The solver that we propose in this article represents a generalisation of the aforementioned algorithm for a high-order discretisation.

2.2. A fast solver based on FFT in one-dimension. To briefly recall the standard Fast Fourier Transform (FFT), let us consider the one-dimensional case. We assume that the computational domain is $[0, L]$ and define the grid points

$$x_n = n\,h, \quad n \in \mathcal{N}, \quad N\,h = L,$$

where we have defined $\mathcal{N} = \big\{0, 1, .., N-1\big\}$. We introduce the $\mathcal{P}^1$-Lagrange shape functions $\{\varphi_n\}$ as the set of functions that are continuous and periodic in $[0, L]$, i.e. they belong to $C^0_\sharp([0, L])$ and they are affine in each interval $[x_n, x_{n+1}]$ with $n \in \mathcal{N}$ and such that $\varphi_n(x_m) = \delta_{nm}$ for all $n, m \in \mathcal{N}$. We have

$$u_h(x) = \sum_{n\in\mathcal{N}} u_n\varphi_n(x), \quad v_h(x) = \sum_{n\in\mathcal{N}} v_n\varphi_n(x), \quad f_h(x) = \sum_{n\in\mathcal{N}} f_n\varphi_n(x), \tag{2.6}$$

where the coefficients u_n, v_n and f_n represent the values of u_h, v_h, f_h, respectively, at the points x_n. Finally, we assume that the source term f_h belongs to $C^0_\sharp([0, L])$ and its values $f_n \coloneqq f_h(x_n)$ are given. Based on these known values, we can construct an interpolation function

$$\tilde{f}_h(x) = \frac{1}{N}\sum_{k\in\mathcal{N}} \hat{f}_k\, e^{\frac{i2\pi xk}{N}}, \tag{2.7}$$

that should agree with f_h at the grid points. This is guaranteed by the use of the Discrete Fourier Transform (DFT) of the function f_h and its inverse, called Inverse Discrete Fourier Transform (IDFT). It is possible to prove the proposition ([14]) below.

Proposition 2.1. *Let $f_h \in C^0_\sharp([0, L])$ and $\tilde{f}_h \in C^0_\sharp([0, L])$ be defined by (2.7). Then,*

$$\begin{aligned} f_h(x_n) = \tilde{f}_h(x_n),\ \forall n \in \mathcal{N} \iff \hat{f}_k &= \sum_{n\in\mathcal{N}} f_h(x_n)\, e^{-\frac{i2\pi nhk}{L}},\ \forall k \in \mathcal{N} \\ \iff f_h(x_n) &= \frac{1}{N}\sum_{k\in\mathcal{N}} \hat{f}_k\, e^{\frac{i2\pi nhk}{L}},\ \forall n \in \mathcal{N}. \end{aligned}$$

The basic ingredient to prove this proposition is the following orthogonality property:

$$\sum_{\ell\in\mathcal{N}} e^{-\frac{i2\pi nh\ell}{L}}\, e^{\frac{i2\pi nhk}{L}} = \delta_{k,\ell}, \quad \forall k, n \in \mathcal{N}. \tag{2.8}$$

Derivation of the symbol of the operator. If we adopt the trapezoidal rule for the computation of each integral in (2.4), i.e.

$$\int_{[x_n,x_{n+1}]}^{Q} g(x)\,dx = \frac{h}{2}\big(g(x_n)+g(x_{n+1})\big), \tag{2.9}$$

for all $n \in \mathcal{N}$, then we retrieve the finite difference scheme (with the convention $u_{-1} = u_{N-1}$)

$$\rho\, u_n\, h - \frac{1}{h}\Big(u_{n+1} + u_{n-1} - 2\,u_n\Big) = f_n\, h \quad n \in \mathcal{N}. \tag{2.10}$$

Analogously, we can construct an interpolation function for the solution $u_h(x)$, that reads

$$\tilde{u}_h(x) = \frac{1}{N}\sum_{k\in\mathcal{N}} \hat{u}_k\, e^{\frac{i2\pi xk}{N}} \quad \text{and} \quad \hat{u}_k = \sum_{n\in\mathcal{N}} u_n\, e^{-\frac{i2\pi nhk}{L}}. \tag{2.11}$$

Therefore, thanks to Proposition 2.1, $\tilde{u}_n := \tilde{u}_h(x_n) = u_n$, for all $n \in \mathcal{N}$. Denoting again $\tilde{u}_{-1} = \tilde{u}_{N-1}$, we obtain that

$$\rho\, \tilde{u}_n\, h - \frac{1}{h}\Big(\tilde{u}_{n+1} + \tilde{u}_{n-1} - 2\,\tilde{u}_n\Big) = \tilde{f}_n\, h \quad n \in \mathcal{N}. \tag{2.12}$$

By definition of the interpolation functions in Eqs. (2.7) and (2.11) and due to the orthogonality property (2.8), we get the following equations

$$\mathcal{S}_k \hat{u}_k = \hat{f}_k, \quad \mathcal{S}_k := \rho - \frac{2}{h^2}\big(\cos(2\pi hk/L) - 1\big),\ \forall k \in \mathcal{N}, \tag{2.13}$$

where $\mathcal{S}_k$ is called the symbol of the operator associated with Eq. (2.10). Note that (2.10) is the finite difference discretisation of the 1D equation (2.1). In fact, it is known that first-order finite elements, with the choice of quadrature (2.9), are equivalent to finite differences on regular meshes. Hence, the symbol $\mathcal{S}_k$ is similar to the finite difference symbol that is usually encountered in the literature. A similar remark holds for higher dimensions in space.

The algorithm. The complete algorithm for the solution of Problem (2.10) can be deduced from (2.13) and resumed in three main steps:

1. Perform a DFT of the discrete source term

$$\hat{f}_k = \sum_{n\in\mathcal{N}} f_n\, e^{-\frac{i2\pi nhk}{L}},\ \forall k \in \mathcal{N};$$

2. Solve for each frequency

$$\mathcal{S}_k \hat{u}_k = \hat{f}_k, \quad k \in \mathcal{N}; \tag{2.14}$$

3. Perform an inverse DFT of the solution for each frequency

$$u_n = \frac{1}{N}\sum_{k\in\mathcal{N}} \hat{u}_k\, e^{\frac{i2\pi nhk}{L}},\ \forall n \in \mathcal{N}.$$

This algorithm has several advantages. First, no storage of finite element matrices is required (as for any finite difference scheme). Furthermore, the problem for each frequency k is decoupled. Therefore, operations can be performed in parallel on the frequencies. At last, the first and third steps can be performed by a Fast Fourier Transform (FFT) algorithm. This algorithm is efficient: the computation of a FFT of N points only requires $O(N \log N)$ arithmetical operations.

Remark If $\rho = 0$, then $\mathcal{S}_0 = 0$. In that case, Eq. (2.14) reads $\hat{f}_0 = 0$. This implies that the source term must satisfy the compatibility condition

$$\sum_{n\in\mathcal{N}} f_n = 0,$$

that corresponds to the standard property that f should have a zero mean value when solving $-\Delta u = f$ in a periodic domain. In that case, the solution u is defined up to a constant value and one can set $\hat{u}_k = 0$ to recover the solution with zero mean value.

3. The High-Order Spectral Element FFT (HOFFT) solver in one-dimension.

3.1. Periodic boundary conditions. If higher-order approximation in space is considered for the numerical solution of Eq. (2.3), the Fourier-based method described above is not suitable and needs to be generalised. First, we introduce the framework for higher-order spatial approximation of Eq. (2.3). This formulation is inspired by [1] and [10]. For the sake of clarity, we consider a one-dimensional domain of size $[0, L]$ and impose periodic boundary conditions (BCs) at first. We refer the reader to Sections 3.2 and 4 for the extension to Neumann and Dirichlet BCs and to multiple dimensions, respectively.
Let $\mathcal{V}_{h,R}$ be the approximation space for continuous, periodic and piecewise R-order polynomials in $[0, L]$. We introduce a basis of shape functions $\{\varphi_{n,j}\}_{n\in\mathcal{N},\, j\in\mathcal{R}}$, where we have defined $\mathcal{N} := \{0, 1, .., N-1\}$ and $\mathcal{R} := \{0, 1, .., R-1\}$. These functions satisfy

$$\begin{aligned}
&\mathrm{Supp}(\varphi_{n,0}) = [(n-1)h, (n+1)h], \quad && \forall n \in \mathcal{N}^*,\\
&\mathrm{Supp}(\varphi_{n,j}) = [nh, (n+1)h], && \forall n \in \mathcal{N},\ \forall j \in \mathcal{R}^*,
\end{aligned}$$

with $\mathcal{N}^* := \mathcal{N} \setminus \{0\}$ and $\mathcal{R}^* := \mathcal{R} \setminus \{0\}$. In order to take periodicity into account, we also have

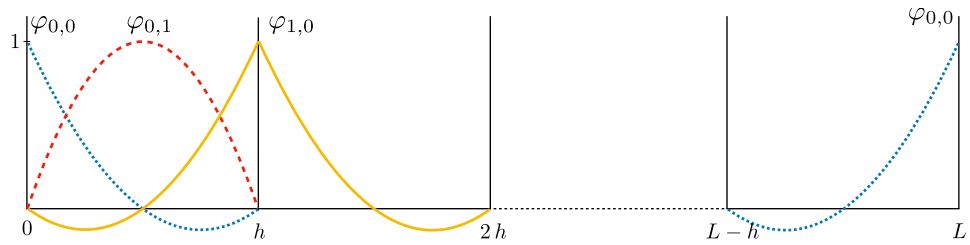
$$\mathrm{Supp}(\varphi_{0,0}) = [0, h] \cup [L-h, L].$$

Moreover, we assume that the shape functions are obtained by translation, namely

(3.1)
$$\forall x \in [0,h],\ \forall n \in \mathcal{N}^*,\ \forall j \in \mathcal{R}^*, \quad \begin{cases} \varphi_{n,0}(x + n\,h) = \varphi_{0,0}(x) & \in \mathcal{P}^R([0,h]),\\ \varphi_{n,0}\big(x + (n-1)\,h\big) = \varphi_{1,0}(x) & \in \mathcal{P}^R([0,h]),\\ \varphi_{n,j}(x + n\,h) = \varphi_{0,j}(x) & \in \mathcal{P}^R([0,h]), \end{cases}$$

where $\mathcal{P}^R([0,h])$ denotes the set of polynomials of degree R on $[0,h]$, and where, by periodicity again, $\varphi_{0,0}(x + L - h) = \varphi_{1,0}(x)$, for $x \in [0,h]$. We introduce the set of Gauss-Lobatto points $\{\xi_j\}_{j=0}^R$, s.t. $\xi_j \in [0,1]$ for all $j \in \mathcal{R}$ and, by definition of shape

Figure 1 Shape functions for quadratic Lagrange polynomial interpolation on Gauss-Lobatto points in $[0, L]$.



functions, we have

$$\varphi_{m,i}\big((\xi_j+n)h\big) = \delta_{m\,n}\,\delta_{i\,j},\ \forall i,j\in\mathcal{R},\ \forall m,n\in\mathcal{N}.$$

By way of illustration, Lagrange basis functions for $R=2$ are depicted in Figure 1. Then, f_h and u_h belong to $\mathcal{V}_{h,R}$ and can be rewritten as

$$f_h(x) = \sum_{n\in\mathcal{N}}\sum_{j\in\mathcal{R}} f_{n,j}\,\varphi_{n,j}(x), \quad u_h(x) = \sum_{n\in\mathcal{N}}\sum_{j\in\mathcal{R}} u_{n,j}\,\varphi_{n,j}(x),$$

with coefficients

$$f_{n,j} \coloneqq f_h\big((\xi_j+n)h\big), \quad u_{n,j} \coloneqq u_h\big((\xi_j+n)h\big) \quad \forall j\in\mathcal{R}, \forall n\in\mathcal{N},$$

and an analogous decomposition holds for $v_h\in\mathcal{V}_{h,R}$. We can introduce an interpolation function of f_h defined as

$$\tilde{f}_h(x) = \frac{1}{N}\sum_{k\in\mathcal{N}}\Big(\sum_{j\in\mathcal{R}}\hat{f}_{k,j}\,e^{\frac{i2\pi(x-h\xi_j)k}{L}}\sum_{n\in\mathcal{N}}\varphi_{n,j}(x)\Big). \tag{3.2}$$

As a consequence of our definition, we obtain

$$\tilde{f}_h\big((\xi_j+n)h\big) = \frac{1}{N}\sum_{k\in\mathcal{N}}\hat{f}_{k,j}\,e^{\frac{i2\pi nk}{N}},\ \forall j\in\mathcal{R}, \tag{3.3}$$

since $h\,N = L$. Eq. (3.3) represents a generalisation of Eq. (2.7) for higher-order spatial approximation. Therefore, the coefficients $\hat{f}_{k,j}$, for all $j\in\mathcal{R}$, can be computed by means of a classical DFT. In particular, the following Proposition holds.

Proposition 3.1. *Let the functions f_h and $\tilde{f}_h$ belong to $C^0_\sharp([0,L])$ and let $\tilde{f}_h$ be defined by Eq. (3.2). Defining*

$$\tilde{f}_{n,j} \coloneqq \tilde{f}_h\big((\xi_j+n)h\big) \quad \forall j\in\mathcal{R},\ \forall n\in\mathcal{N},$$

then we have, for all j in $\mathcal{R}$,

$$\begin{aligned} f_{n,j} = \tilde{f}_{n,j} \iff \hat{f}_{k,j} &= \sum_{n\in\mathcal{N}} f_{n,j}\,e^{-\frac{i2\pi nk}{N}},\ \forall k\in\mathcal{N},\\ \iff f_{n,j} &= \frac{1}{N}\sum_{k\in\mathcal{N}}\hat{f}_{k,j}\,e^{\frac{i2\pi nk}{N}},\ \forall n\in\mathcal{N}. \end{aligned} \tag{3.4}$$

Proposition 3.1 is a straightforward consequence of Proposition 2.1 and Eq. (3.3). Following the same approach as in Section 2.2, we introduce the interpolation function of u_h,

$$\tilde{u}_h(x) = \frac{1}{N} \sum_{k\in\mathcal{N}} \Big(\sum_{j\in\mathcal{R}} \hat{u}_{k,j}\, e^{\frac{i2\pi(x-h\xi_j)k}{L}} \sum_{n\in\mathcal{N}} \varphi_{n,j}(x) \Big).$$

where the coefficients $\hat{u}_{k,j}$ are given by

$$\hat{u}_{k,j} = \sum_{n\in\mathcal{N}} u_{n,j}\, e^{-\frac{i2\pi nk}{N}},\ \forall k \in \mathcal{N},\ \forall j \in \mathcal{R}. \tag{3.5}$$

We have the property

$$\tilde{u}_{n,j} := \tilde{u}_h\big((\xi_j + n)h\big) = \frac{1}{N} \sum_{k\in\mathcal{N}} \hat{u}_{k,j}\, e^{\frac{i2\pi nk}{N}},\ \forall j \in \mathcal{R}, \tag{3.6}$$

and, as a consequence of Eqs. (3.6) and (3.5) and Proposition 3.1, we derive

$$u_{n,j} = \tilde{u}_{n,j}, \quad \forall n \in \mathcal{N},\ \forall j \in \mathcal{R}. \tag{3.7}$$

We now manipulate the bilinear forms in Eq. (2.3). If the same quadrature formula is used for each segment of size h, the bilinear form m_h in Eq. (2.3) can be rewritten

$$m_h(u_h, v_h) = \sum_{n\in\mathcal{N}} \int_{[nh,(n+1)h]}^{Q} u_h(x)\, \overline{v_h(x)}\, \mathrm{d}x = \sum_{n\in\mathcal{N}} \int_{[0,h]}^{Q} u_h(x+nh)\, \overline{v_h(x+nh)}\, \mathrm{d}x.$$

Consequently, by quadrature rule (Eq. (2.5)) and the definition of shape functions, we obtain

$$\int_{[0,h]}^{Q} u_h(x+nh)\, \overline{v_h(x+nh)}\, \mathrm{d}x = h \sum_{k=0}^{R_Q} \sum_{i,j=0}^{R} \omega_k\, \varphi_{0,i}(\eta_k\, h)\, \varphi_{0,j}(\eta_k\, h)\, u_{n,j}\, \overline{v_{n,i}},$$

where, for the sake of conciseness, we use the notations $\varphi_{0,R}(x) = \varphi_{1,0}(x)$ and $u_{n,R} = u_{n+1,0}$ and, by periodicity, $u_{N-1,R} = u_{N,0} = u_{0,0}$ (a same notation is used for $v_{n,i}$ and $f_{n,i}$). Finally, one can show that

$$m_h(u_h, v_h) = \sum_{n\in\mathcal{N}} \sum_{i,j=0}^{R} \hat{m}_{i,j}\, u_{n,j}\, \overline{v_{n,i}}, \quad \text{with} \quad \hat{m}_{i,j} = h \sum_{k=0}^{R_Q} \omega_k\, \varphi_{0,i}(\eta_k\, h)\, \varphi_{0,j}(\eta_k\, h),$$

i.e. $\hat{m}_{i,j}$ denote the coefficients of the mass matrix in the reference element $[0, h]$. A similar treatment of the bilinear form a_h gives

$$a_h(u_h, v_h) = \sum_{n\in\mathcal{N}} \sum_{i,j=0}^{R} \hat{a}_{i,j} u_{n,j}\, \overline{v_{n,i}}, \quad \text{with} \quad \hat{a}_{i,j} = h \sum_{k=0}^{R_Q} \omega_k\, \underline{\nabla}\, \varphi_{0,i}(\eta_k\, h) \cdot \underline{\nabla}\, \varphi_{0,j}(\eta_k\, h).$$

Note that the coefficients $\hat{a}_{i,j}$ and $\hat{m}_{i,j}$ do not depend on the index n, as a consequence of the invariance of the discrete bilinear forms with respect to translations. This property is fundamental for the derivation of the symbol of the operator. Furthermore, due to the use of Gauss-Lobatto nodes both for the interpolation and the quadrature rule (i.e. $\{\eta_k\} = \{\xi_k\}$), the mass matrix is diagonal. This property, named mass lumping,

is a fundamental feature of Spectral Element Methods (SEM), first proposed by Maday and Patera [18]. We make this choice for the rest of the article. Consequently, we obtain

$$m_h(u_h, v_h) = \sum_{n\in\mathcal{N}} \sum_{i=0}^{R} \hat{m}_{i,i} u_{n,i}\, \overline{v_{n,i}},$$

and the algebraic system to solve is

$$\sum_{n\in\mathcal{N}} \sum_{i=0}^{R} \hat{m}_{i,i} u_{n,i}\, \overline{v_{n,i}} + \sum_{n\in\mathcal{N}} \sum_{i,j=0}^{R} \hat{a}_{i,j} u_{n,j}\, \overline{v_{n,i}} = \sum_{n\in\mathcal{N}} \sum_{i=0}^{R} \hat{m}_{i,i} f_{n,i}\, \overline{v_{n,i}}. \tag{3.8}$$

Since Eq. (2.3) must be true for any $v_h \in \mathcal{V}_h$, it must hold true for any choice of $v_{n,i}$, $n \in \mathcal{N},\ i \in \{0, 1, .., R\}$. In particular, for $v_{n,i} = \delta_{n\,\ell}\,\delta_{i\,q}$, with $q \in \mathcal{R}^*$ and $\ell \in \mathcal{N}$, we obtain

$$\hat{m}_{q,q}\, u_{\ell,q} + \sum_{j\in\mathcal{R}^*} \hat{a}_{q,j}\, u_{\ell,j} + \hat{a}_{q,0}\, u_{\ell,0} + \hat{a}_{q,R}\, u_{\ell+1,0} = \hat{m}_{q,q}\, f_{\ell,q}, \tag{3.9}$$

where we have used the property that $u_{\ell,R} = u_{\ell+1,0}$ by definition. Then, in Eq. (3.8) we set, for all $n \in \mathcal{N}$, $v_{n,0} = v_{n-1,R} = \delta_{n\,\ell}$ and $v_{n,i} = 0$ for $i \in \mathcal{R}^*$. We find

$$\begin{aligned}(\hat{m}_{0,0} + \hat{m}_{R,R})\, u_{\ell,0} &+ \sum_{j\in\mathcal{R}^*} \hat{a}_{0,j}\, u_{\ell,j} + \hat{a}_{0,0}\, u_{\ell,0} + \hat{a}_{0,R}\, u_{\ell+1,0} \\ &+ \sum_{j\in\mathcal{R}^*} \hat{a}_{R,j}\, u_{\ell-1,j} + \hat{a}_{R,0}\, u_{\ell-1,0} + \hat{a}_{R,R}\, u_{\ell,0} = (\hat{m}_{0,0} + \hat{m}_{R,R})\, f_{\ell,0},\end{aligned} \tag{3.10}$$

where, again, we have used the property that $f_{\ell,0} = f_{\ell-1,R}$. Note that Eq. (3.9) is still true when we replace $u_{\ell,j}$ by $\tilde{u}_{\ell,j}$ and $f_{\ell,j}$ by $\tilde{f}_{\ell,j}$, due to Eq. (3.7). Following the strategy of Section 2.2, we further replace $\tilde{u}_{\ell,j}$ by their expression in terms of $\hat{u}_{k,j}$ (Eq. (3.6)) and $\tilde{f}_{\ell,j}$ by their expression in terms of $\hat{f}_{k,j}$ (Eq. (3.3)), for all $j \in \mathcal{R}$. Therefore, Eq. (3.9) becomes, $\forall q \in \mathcal{R}^*$,

$$\begin{aligned}\sum_{k\in\mathcal{N}} \Bigg(\hat{m}_{q,q}\, \hat{u}_{k,q}\, e^{\frac{i2\pi k\ell}{N}} + \sum_{j\in\mathcal{R}^*} \hat{a}_{q,j}\, \hat{u}_{k,j}\, e^{\frac{i2\pi k\ell}{N}} + \big(\hat{a}_{q,0} + \hat{a}_{q,R}\, e^{\frac{i2\pi k}{N}}\big) \hat{u}_{k,0}\, e^{\frac{i2\pi k\ell}{N}} \Bigg) \\ = \sum_{k\in\mathcal{N}} \hat{m}_{q,q}\, \hat{f}_{k,q}\, e^{\frac{i2\pi k\ell}{N}}.\end{aligned} \tag{3.11}$$

since Eq (3.5) implies that $\hat{u}_{\ell+1,0} = e^{\frac{i2\pi k}{N}}\, \hat{u}_{\ell,0}$. Moreover, with the same strategy, we deduce from Eq. (3.10) the following equation:

$$\begin{aligned}&\sum_{k\in\mathcal{N}} \Bigg((\hat{m}_{0,0} + \hat{m}_{R,R})\, \hat{u}_{k,0}\, e^{\frac{i2\pi k\ell}{N}} \Bigg) \\ &\quad + \sum_{k\in\mathcal{N}} \Bigg(\sum_{j\in\mathcal{R}^*} \hat{a}_{0,j}\, \hat{u}_{k,j}\, e^{\frac{i2\pi k\ell}{N}} + \big(\hat{a}_{0,0} + \hat{a}_{0,R}\, e^{\frac{i2\pi k}{N}}\big) \hat{u}_{k,0}\, e^{\frac{i2\pi k\ell}{N}} \Bigg) \\ &\quad + \sum_{k\in\mathcal{N}} \Bigg(\sum_{j\in\mathcal{R}^*} \hat{a}_{R,j}\, e^{\frac{-i2\pi k}{N}}\, \hat{u}_{k,j}\, e^{\frac{i2\pi k\ell}{N}} + \big(\hat{a}_{R,0}\, e^{\frac{-i2\pi k}{N}} + \hat{a}_{R,R}\big) \hat{u}_{k,0}\, e^{\frac{i2\pi k\ell}{N}} \Bigg) \\ &\qquad = \sum_{k\in\mathcal{N}} (\hat{m}_{0,0} + \hat{m}_{R,R})\, \hat{f}_{k,0}\, e^{\frac{i2\pi k\ell}{N}}.\end{aligned} \tag{3.12}$$

Let us now define the vectors $\underline{\hat{U}}_k, \underline{\hat{F}}_k \in \mathbb{C}^R$, such that

$$\underline{\hat{U}}_k = (\hat{u}_{k,0}\,, \hat{u}_{k,1}, .., \hat{u}_{k,R-1})^T, \quad \underline{\hat{F}}_k = (\hat{f}_{k,0}\,, \hat{f}_{k,1}, .., \hat{f}_{k,R-1})^T,$$

and the matrices $\boldsymbol{\mathcal{A}}_k, \boldsymbol{\mathcal{M}} \in \mathbb{C}^{R\times R}$ such that

$$\boldsymbol{\mathcal{A}}_k = \begin{bmatrix} \mathrm{a}_k & \underline{\mathcal{B}}_k \\ \underline{\mathcal{B}}_k^* & \mathring{\boldsymbol{\mathcal{A}}} \end{bmatrix}, \quad \boldsymbol{\mathcal{M}} = \begin{bmatrix} \mathcal{M} & 0 \\ 0 & \mathring{\boldsymbol{\mathcal{M}}} \end{bmatrix}, \tag{3.13}$$

where a_k and $\mathcal{M}$ belong to $\mathbb{R}$, the vector $\underline{\mathcal{B}}_k$ belongs to $\mathbb{C}^{R-1}$ and the matrices $\mathring{\boldsymbol{\mathcal{A}}}$ and $\mathring{\boldsymbol{\mathcal{M}}}$ belong to $\mathbb{R}^{(R-1)\times(R-1)}$. In particular, they are given by

$$\mathrm{a}_k = \hat{a}_{0,0} + \hat{a}_{R,R} + 2\,\hat{a}_{0,R}\,\cos(2\pi k/N), \quad \mathcal{M} = \hat{m}_{0,0} + \hat{m}_{R,R},$$

and

$$\mathcal{B}_{k,q} = \hat{a}_{0,q} + \hat{a}_{R,q}\,e^{i2\pi k/N}, \quad \mathring{\boldsymbol{\mathcal{M}}}_{q\,q^*} = \hat{m}_{q,q}\,\delta_{q\,q^*}, \quad \mathring{\boldsymbol{\mathcal{A}}}_{q\,q^*} = \hat{a}_{q,q^*},\ q, q^* \in \mathcal{R}^*.$$

Thereupon, Eqs. (3.11) and (3.12) can be rewritten in compact form as

$$\sum_{k\in\mathcal{N}} e^{\frac{i2\pi k\ell}{N}}\,(\rho\,\boldsymbol{\mathcal{M}} + \boldsymbol{\mathcal{A}}_k)\,\underline{\hat{U}}_k = \sum_{k\in\mathcal{N}} e^{\frac{i2\pi k\ell}{N}}\,\boldsymbol{\mathcal{M}}\,\underline{\hat{F}}_k,\ \forall \ell \in \mathcal{N}. \tag{3.14}$$

Due to the orthogonality of the factors $e^{\frac{i2\pi k\ell}{N}}$ (Eq. (2.8)), we can further simplify Eq. (3.14), retrieving

$$(\rho\,\boldsymbol{\mathcal{M}} + \boldsymbol{\mathcal{A}}_k)\,\underline{\hat{U}}_k = \boldsymbol{\mathcal{M}}\,\underline{\hat{F}}_k,\ \forall k \in \mathcal{N}.$$

Ultimately, the solution $\underline{\hat{U}}_k$, for every frequency $k \in \mathcal{N}$, is computed by solving

$$\boldsymbol{\mathcal{S}}_k \underline{\hat{U}}_k = \underline{\hat{F}}_k,\ \forall k \in \mathcal{N}, \tag{3.15}$$

where the symbol of the operator $\boldsymbol{\mathcal{S}}_k$ reads

$$\boldsymbol{\mathcal{S}}_k = (\rho\,\mathbf{I} + \boldsymbol{\mathcal{M}}^{-1}\,\boldsymbol{\mathcal{A}}_k).$$

The algorithm. In order to solve (3.8), we can derive an algorithm based on Eqs. (3.4), (3.15) and (3.5). The main steps of this algorithm read:

1. Perform a DFT of the discrete source term for each j in $\mathcal{R}$

$$\hat{f}_{k,j} = \sum_{n\in\mathcal{N}} f_{n,j}\,e^{-\frac{i2\pi nk}{N}},\ \forall k \in \mathcal{N};$$

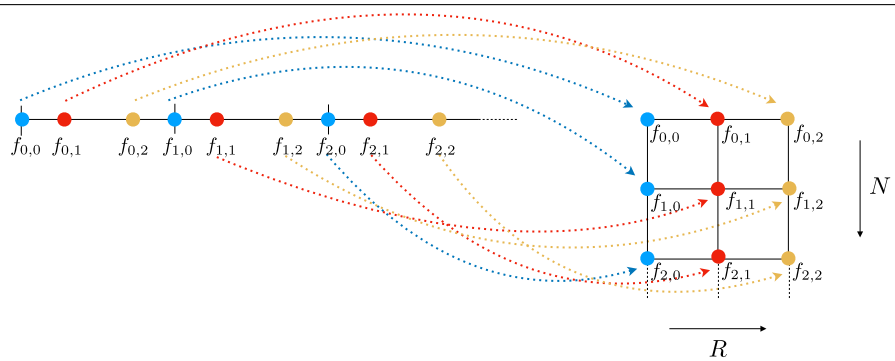
2. Solve for each frequency

$$\boldsymbol{\mathcal{S}}_k \underline{\hat{U}}_k = \underline{\hat{F}}_k,\ \forall k \in \mathcal{N}; \tag{3.16}$$

3. Perform an inverse DFT for each j in $\mathcal{R}$

$$u_{n,j} = \frac{1}{N}\sum_{k\in\mathcal{N}} \hat{u}_{k,j}\,e^{\frac{i2\pi nk}{N}},\ \forall n \in \mathcal{N}.$$

Figure 2 Redistribution of the source term in matrix form. Row index corresponds to the element, column index refers to the Gauss-Lobatto node in the reference element.



A crucial aspect of this method concerns the definition of an efficient algorithm to compute the coefficients $f_{n,j}$ and $\hat{f}_{k,j}$. To this end, we re-organise the source term vector in matrix form, in which the row index corresponds to the element considered, whereas the column index refers to the Gauss-Lobatto point in the reference element of size h. Then, the terms $\hat{f}_{k,j}$ are obtained by DFT performed column by column (using the FFT algorithm). See Figure 2 for a schematic illustration of the procedure.

Inversion of the symbol by SVD. In this section we anticipate the difficulties that will be encountered in multiple dimensions, namely the inversion of the symbol of the operator $\boldsymbol{\mathcal{S}}_k$ in Eq. (3.16). We now derive a pseudo-explicit inversion of this matrix by means of the Singular Value Decomposition (SVD). For this purpose, let us denote by $\left\{\lambda_{k,i}, \underline{V}_{k,i}\right\}_{i\in\mathcal{R}}$ the set of eigenvalues and eigenvectors of $\boldsymbol{\mathcal{S}}_k$. They satisfy the eigenvalue problem $\boldsymbol{\mathcal{S}}_k\,\underline{V}_{k,i} = \lambda_{k,i}\,\underline{V}_{k,i}$ for all $i \in \mathcal{R}$. Since $\boldsymbol{\mathcal{M}}\,\boldsymbol{\mathcal{S}}_k$ inherits the properties of the bilinear forms m_h and a_h, one can show that, for $\rho \neq 0$, it is a hermitian and positive-definite matrix. Hence, all the eigenvalues $\lambda_{k,i}$ of $\boldsymbol{\mathcal{S}}_k$ are positive and real, and all eigenvectors $\underline{V}_{k,i}$ are orthonormal with respect to the scalar product by $\boldsymbol{\mathcal{M}}$, i.e.

$$\underline{V}^*_{k,j}\,\boldsymbol{\mathcal{M}}\,\underline{V}_{k,i} = \delta_{ij},\ \forall i,j \in \mathcal{R} \quad \text{and} \quad \sum_{i\in\mathcal{R}} \underline{V}_{k,i}\,\underline{V}^*_{k,i} = \boldsymbol{\mathcal{M}}^{-1}.$$

After some standard algebra, we obtain

$$\boldsymbol{\mathcal{S}}_k = \sum_{i\in\mathcal{R}} \lambda_{k,i}\,\underline{V}_{k,i}\,\underline{V}^*_{k,i}\,\boldsymbol{\mathcal{M}} \implies \boldsymbol{\mathcal{S}}_k^{-1} = \sum_{i\in\mathcal{R}} \lambda_{k,i}^{-1}\,\underline{V}_{k,i}\,\underline{V}^*_{k,i}\,\boldsymbol{\mathcal{M}}.$$

Therefore, Eq. (3.16) can be rewritten as

$$\underline{\hat{U}}_k = \sum_{i\in\mathcal{R}} \lambda_{k,i}^{-1}\,\underline{V}_{k,i}\,\underline{V}^*_{k,i}\,\boldsymbol{\mathcal{M}}\,\underline{\hat{F}}_k. \tag{3.17}$$

At this stage, Eq. (3.17) requires the same computational cost as Eq. (3.16), since the computation of the eigenvalues and eigenvectors of a matrix implies the same number of operations as the multiplication by an inverse matrix. Nevertheless, we will show in Section 4 that the generalisation to higher dimensions becomes very efficient, due to tensorisation. Note that, if $\rho = 0$, then $\boldsymbol{\mathcal{S}}_0$ has not full rank and Eq. (3.16) yields

a compatibility condition for the source term. It can be proved that such condition reads (see the remark below for further detail)

$$\sum_{n\in\mathcal{N}}\sum_{i=0}^{R}\hat{m}_{i,i}f_{n,i}=0 \tag{3.18}$$

and corresponds to the standard zero mean condition. Such property can be imposed on the solution by the following procedure: we introduce the Moore–Penrose pseudo-inverse λ^+ of a scalar λ defined as

$$\lambda^+=\begin{cases}0, & \lambda=0,\\ \lambda^{-1}, & \lambda\neq 0.\end{cases}$$

Then, we define

$$\boldsymbol{\mathcal{S}}_k^+:=\sum_{i\in\mathcal{R}}\lambda_{k,i}^+\,\underline{V}_{k,i}\,\underline{V}_{k,i}^*\,\boldsymbol{\mathcal{M}} \tag{3.19}$$

and solve (3.16) by computing $\underline{\hat{U}}_k=\boldsymbol{\mathcal{S}}_k^+\,\underline{\hat{F}}_k$.

Remark The compatibility condition (3.18) is obtained by observing that the constant function belongs to $\mathcal{V}_{h,R}$ and, therefore, $a_h(u_h,1)=0$ (see Eq. (2.4)). Hence, considering Eq. (2.3), one can see that if $\rho=0$ one must guarantee that $\ell_h(1)=m_h(f_h,1)=0$. Eq. (3.18) is then retrieved from Eq. (3.8).

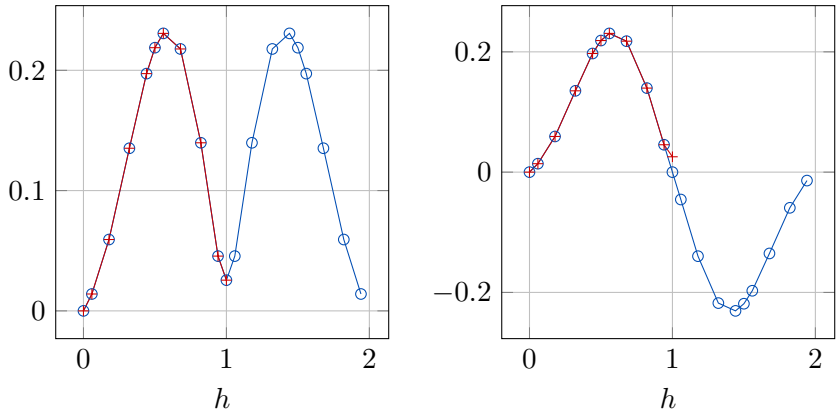
3.2. Neumann and Dirichlet Boundary conditions. When linear interpolation is adopted, it is standard to use, in the definition of the interpolation function (2.7), the Discrete Sine Transform (DST) for Dirichlet BCs, or Discrete Cosine Transform (DCT) for Neumann BCs. If a high-order approximation is considered, it is not possible to use DCT or DST. Consequently, we explicitly "double" the computational domain and the discrete source term f_h in order to extend it into a periodic one and perform DFT. According to the BC imposed, we perform an even (for Neumann BCs) or odd (for Dirichlet BCs) extension of our source term (that is set to 0 at the end-points of the interval for antisymmetric extensions). See Figure 3 for a graphical illustration of this procedure in one-dimension. Therefore, we need to take into account $2N$ frequencies, instead of N frequencies for the periodic case. However, thanks to the symmetry properties of the DFT, we are able to restrict most operations to $N+1$ frequencies, as it is shown in what follows. Note that, after doubling the computational domain and performing an even (or odd) extension of the source term, depending on the BCs considered, the sequence $\left\{f_{n,R-j}\right\}_{n\in\mathcal{N},j\in\mathcal{R}}$ satisfies, for any given $n\in\mathcal{N}$,

$$\begin{cases}f_{n,R-j}=f_{2N-n-1,j}, & \text{if } j\in\mathcal{R}^*,\\ f_{n,0}=f_{2N-n,0},\end{cases} \tag{3.20}$$

for an even extension, and

$$\begin{cases}f_{n,R-j}=-f_{2N-n-1,j}, & \text{if } j\in\mathcal{R}^*,\\ f_{n,0}=-f_{2N-n,0},\end{cases} \tag{3.21}$$

Figure 3 Illustration of a periodic extension of a discrete, one-dimensional function at the Gauss-Lobatto nodes. Left: comparison between the original function and its even expansion (for Neumann BCs); right: comparison between the original function and its odd expansion (for Dirichlet BCs). Gauss-Lobatto nodes for N=2 elements, 5th-order Lagrange polynomials.



for an odd extension. Henceforth, we use the symbol $\pm$ for the sake of conciseness, where $+$ is associated with Neumann conditions, whereas $-$ corresponds to Dirichlet conditions. We define the coefficients $\hat{f}_{k,j}$ following Eq. (3.4). Given $j \in \mathcal{R}$, they read

$$\hat{f}_{k,j} = \sum_{n=0}^{2N-1} f_{n,j}\, e^{-\frac{i\pi n k}{N}}, \quad \forall k \in \{0, 1, .., 2N-1\}.$$

From the properties of the scalars $\{f_{n,j}\}_{n\in\mathcal{N}, j\in\mathcal{R}}$ we can deduce some symmetry properties of the coefficients $\hat{f}_{k,j}$, as it is stated in the following lemma. These properties allow to reduce the number of frequencies considered for the construction of the symbol of the operator. Due to the use of high-order polynomial interpolation, this symmetry is not trivial.

Lemma 3.2. *Assume that $\{f_{n,j}\}_{n\in\mathcal{N}, j\in\mathcal{R}}$ satisfies Eq. (3.20) or Eq. (3.21). Then, the coefficients $\hat{f}_{k,j}$ read, $\forall k \in \mathcal{N}^*$*

$$\begin{cases} \hat{f}_{2N-k,j} = \pm e^{\frac{-i\pi k}{N}} \hat{f}_{k,R-j}, & \text{if } \; j \in \mathcal{R}^*, \\ \hat{f}_{2N-k,0} = \pm \hat{f}_{k,0}, \end{cases} \tag{3.22}$$

with the established convention for the plus and minus signs.

Proof. Let us consider an even extension of the discrete source term, and fix $j \in \mathcal{R}^*$. On the one hand, due to Eq. (3.20), Eq. (3.21) and Lemma 3.2, we are able

to retrieve that, for $k \in \mathcal{N} \cup \{N\}$,

$$\hat{f}_{k,R-j} = \sum_{n=0}^{2N-1} f_{n,R-j}\, e^{-\frac{i\pi nk}{N}} = \sum_{n=0}^{2N-1} f_{2N-n-1,j}\, e^{-\frac{i\pi nk}{N}}$$
$$= \sum_{m=0}^{2N-1} f_{m,j}\, e^{\frac{-i2N\pi k}{N}} e^{\frac{i\pi mk}{N}} e^{\frac{i\pi k}{N}} = e^{\frac{i\pi k}{N}} \sum_{m=0}^{2N-1} f_{m,j}\, e^{\frac{i\pi mk}{N}}.$$

On the other hand, due to a standard property of the DFT, we have that

$$\hat{f}_{2N-k,j} = \sum_{m=0}^{2N-1} f_{m,j}\, e^{\frac{-i\pi m(2N-k)}{N}} = \sum_{m=0}^{2N-1} f_{m,j}\, e^{\frac{i\pi mk}{N}},\ \forall j \in \mathcal{R}.$$

Hence, we deduce that

$$\hat{f}_{k,R-j} = e^{\frac{i\pi k}{N}} \hat{f}_{2N-k,j}. \tag{3.23}$$

In addition, for $j = 0$ and $k \in \mathcal{N} \cup \{N\}$, we obtain

$$\begin{aligned} \hat{f}_{k,0} &= \sum_{n=0}^{2N-1} f_{n,0}\, e^{-\frac{i\pi nk}{N}} = \sum_{n=0}^{2N-1} f_{2N-n,0}\, e^{-\frac{i\pi nk}{N}} \\ &= \sum_{m=1}^{2N} f_{m,0}\, e^{\frac{-i2N\pi k}{N}} e^{\frac{i\pi mk}{N}} = \sum_{m=0}^{2N-1} f_{m,0}\, e^{\frac{i\pi mk}{N}} \\ &= \hat{f}_{2N-k,0}, \end{aligned} \tag{3.24}$$

since, by periodicity, $f_{2N,0} = f_{0,0}$. The proof for the odd extension of the discrete source term is analogous. ∎

Furthermore, we state the following proposition:

PROPOSITION 3.3. *Let us consider Problem (2.2), with $\Omega = [0, 2L]$ in the periodic setting. Then, if the source term f is even (odd), the solution u is endowed with the same properties, i.e. it is even (odd).*

Note that Proposition 3.3 also holds true for the discrete problem (3.8), and f_h, u_h. From Lemma 3.2 and Proposition 3.3, we deduce that the discrete solution u_h satisfies Eq. (3.22). For this reason, we can construct an efficient algorithm for the solution of a Poisson problem by HOFFT that consists in three steps:

1. Perform a DFT of the discrete source term for each j in $\mathcal{R}^*$

$$\hat{f}_{k,j} = \sum_{n=0}^{2N-1} f_{n,j}\, e^{-\frac{i\pi nk}{N}},\ \forall k \in \mathcal{N} \cup \{N\};$$

   with $f_{n,j}$ given by (3.21) for $n \geqslant N$.
2. Apply the pseudo-inverse of the symbol for each frequency

$$\hat{\underline{U}}_k = \boldsymbol{\mathcal{S}}_k^+\, \hat{\underline{F}}_k,\ \forall k \in \mathcal{N} \cup \{N\};$$

3. Perform an inverse DFT for each j in $\mathcal{R}^*$

$$u_{n,j} = \frac{1}{2N} \sum_{k=0}^{2N-1} \hat{u}_{k,j}\, e^{\frac{i\pi n k}{N}}, \ \forall n \in \mathcal{N} \cup \{N\}.$$

with $\hat{u}_{k,j}$ given by (3.22) for $k \geqslant N+1$.

This algorithm relies on the evaluation of the symbol of the operator of the first $N+1$ frequencies only. Note, however, that the standard implementation of the DFT (or its inverse) by the FFT algorithm implies the evaluation of all 2N frequencies. However, in Step 2, only $N+1$ inversions of the symbol are required. We will show in what follows that in multiple dimensions it is possible to optimise Steps 1 and 3, by considering an *ad hoc* extension of the sequence "dimension by dimension" when the FFT (or its inverse) is performed.

4. Extension to higher dimensions. The generalisation to two (or higher) dimensions is performed by tensorisation from the one-dimensional case. Therefore, most of the properties are directly inherited from the one-dimensional case, and we do not provide the proofs, for the sake of conciseness. Henceforth, we denote by $d > 1$ the dimension of the computational domain and for simplicity of exposure we set $\Omega = [0, L]^d$.

4.1. Periodic boundary conditions. First, let us define $\mathcal{N}^d = \{0, 1, .., N-1\}^d$ and $\mathcal{R}^d = \{0, 1, .., R-1\}^d$. We introduce the set of points in the reference element $[0, h]^d$ as

$$\underline{\xi}_{\mathbf{j}} = [\xi_{j_1}, \xi_{j_2}, .., \xi_{j_r}],$$

where ξ_i corresponds to a 1D Gauss-Lobatto point for all $i \in \mathcal{R}$, and $\mathbf{j}$ is a multi-index in $\mathcal{R}^d$. Let us introduce the multi-index $\mathbf{n} = [n_1, n_2, .., n_d] \in \mathcal{N}^d$. Then, the d-dimensional shape functions defined on any $\underline{x} = [x_1, x_2, .., x_d] \in [0, L]^d$ read

$$\Phi_{\mathbf{n},\mathbf{j}}(\underline{x}) = \prod_{r=1}^{d} \varphi_{n_r, j_r}(x_r), \tag{4.1}$$

where $\varphi_{n_r, j_r}(x_r)$ are shape function values defined as in Section (3.1). Therefore, the shape functions $\Phi_{\mathbf{n},\mathbf{j}}$ satisfy, for all $\mathbf{p} \in \mathcal{R}^d$ and all $\mathbf{m} \in \mathcal{N}^d$,

$$\begin{aligned}\Phi_{\mathbf{m},\mathbf{p}}\big((\underline{\xi}_{\mathbf{j}} + \mathbf{n})h\big) &= \Phi_{\mathbf{m},\mathbf{p}}\big((\xi_{j_1} + n_1)h, (\xi_{j_2} + n_2)h, .., (\xi_{j_d} + n_d)h\big) \\ &= \prod_{r=1}^{d} \delta_{m_r\, n_r}\, \varphi_{n_r, p_r}(\xi_{j_r}) = \prod_{r=1}^{d} \delta_{m_r\, n_r}\, \delta_{p_r\, j_r}.\end{aligned}$$

For simplicity of notation, we denote

$$\sum_{\mathbf{n}} := \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} \cdots \sum_{n_d=0}^{N-1}, \quad \sum_{\mathbf{j}} := \sum_{j_1=0}^{R-1} \sum_{j_2=0}^{R-1} \cdots \sum_{j_d=0}^{R-1}, \quad \sum_{\mathbf{n},\mathbf{j}} := \sum_{\mathbf{n}} \sum_{\mathbf{j}}.$$

Moreover, for any vector $V \in \mathbb{C}^{R^d}$, $V(\mathbf{i})$ is its $\mathbf{i}$-th element, without assuming a specific one-dimensional re-ordering of the degrees of freedom with respect to the multi-indices. A similar strategy is used for any matrix $\boldsymbol{V} \in \mathbb{C}^{R^d \times R^d}$: its element on

the $\mathbf{i}$-th line and $\mathbf{j}$-th column is denoted $\boldsymbol{V}(\mathbf{i},\mathbf{j})$. From Eq. (4.1) we derive the following decomposition of the discrete source term f_h and the discrete solution u_h:

$$f_h(\underline{x}) = \sum_{\mathbf{n},\mathbf{j}} f_{\mathbf{n},\mathbf{j}}\, \Phi_{\mathbf{n},\mathbf{j}}(\underline{x}), \quad u_h(\underline{x}) = \sum_{\mathbf{n},\mathbf{j}} u_{\mathbf{n},\mathbf{j}}\, \Phi_{\mathbf{n},\mathbf{j}}(\underline{x}),$$

with

$$f_{\mathbf{n},\mathbf{j}} \coloneqq f_h\big((\underline{\xi}_{\mathbf{j}} + \mathbf{n})h\big) \quad u_{\mathbf{n},\mathbf{j}} \coloneqq u_h\big((\underline{\xi}_{\mathbf{j}} + \mathbf{n})h\big), \quad \forall \mathbf{n} \in \mathcal{N}^d,\ \forall \mathbf{j} \in \mathcal{R}^d.$$

With these notations, the algebraic problem that must be solved reads:
find $\{u_{\mathbf{n},\mathbf{i}}\}$ *such that*

$$\sum_{\mathbf{n}} \sum_{\mathbf{j}} \hat{m}_{\mathbf{i},\mathbf{i}} u_{\mathbf{n},\mathbf{i}}\, \overline{v_{\mathbf{n},\mathbf{i}}} + \sum_{\mathbf{n}} \sum_{\mathbf{i}} \sum_{\mathbf{j}} \hat{a}_{\mathbf{i},\mathbf{j}} u_{\mathbf{n},\mathbf{j}}\, \overline{v_{\mathbf{n},\mathbf{i}}} = \sum_{\mathbf{n}} \sum_{\mathbf{i}} \hat{m}_{\mathbf{i},\mathbf{i}} f_{\mathbf{n},\mathbf{i}}\, \overline{v_{\mathbf{n},\mathbf{i}}}, \tag{4.2}$$

for all $\{v_{\mathbf{n},\mathbf{i}}\}$, where $\{\hat{m}_{\mathbf{i},\mathbf{i}}\}$ and $\{\hat{a}_{\mathbf{i},\mathbf{j}}\}$ correspond to the elements of the mass and stiffness matrices, respectively, computed on the multidimensional reference element $[0,h]^d$, and where u_h, v_h and f_h are periodic in Ω. Now, let $\mathbf{k} \in \mathcal{N}^d$. Since the source term is periodic in $[0,L]^d$, we can generalise the definition of an interpolation function (3.2) for higher dimensions as

$$\tilde{f}_h(\underline{x}) = \frac{1}{N^d} \sum_{\mathbf{k}} \Bigg(\sum_{\mathbf{j}} \hat{f}_{\mathbf{k},\mathbf{j}}\, e^{\frac{i2\pi(\underline{x} - h\underline{\xi}_{\mathbf{j}})\cdot \mathbf{k}}{L}} \sum_{\mathbf{n}} \varphi_{\mathbf{n},\mathbf{j}}(\underline{x}) \Bigg), \tag{4.3}$$

with $h\,N = L$ and

$$e^{\frac{i2\pi(\underline{x} - h\underline{\xi}_{\mathbf{j}})\cdot \mathbf{k}}{L}} = e^{\frac{i2\pi(x_1 - h\xi_{j_1})\,k_1}{L}}\, e^{\frac{i2\pi(x_2 - h\xi_{j_2})\,k_2}{L}} \cdots e^{\frac{i2\pi(x_d - h\xi_{j_d})\,k_d}{L}}.$$

Note that this formula can be easily extended to more general, regular meshes by considering a different length L depending on the direction. Eq. (4.3) is a generalisation of Eq. (3.2) to the multi-dimensional case. Therefore, the coefficients $\hat{f}_{k,j}$, $\forall j \in \mathcal{R}$, can be computed by DFT, as stated in the following proposition.

PROPOSITION 4.1. *Let* $f_h \in C^0_\sharp([0,L]^d)$ *and* $\tilde{f}_h \in C^0_\sharp([0,L]^d)$ *defined by Eq.* (4.3). *Let us denote*

$$\tilde{f}_{\mathbf{n},\mathbf{j}} \coloneqq \tilde{f}_h\big((\underline{\xi}_{\mathbf{j}} + \mathbf{n})h\big) \quad \forall \mathbf{j} \in \mathcal{R}^d,\ \forall \mathbf{n} \in \mathcal{N}^d.$$

Then, for all $\mathbf{j}$ *fixed in* $\mathcal{R}^d$,

$$\begin{aligned} f_{\mathbf{n},\mathbf{j}} = \tilde{f}_{\mathbf{n},\mathbf{j}} \iff \hat{f}_{\mathbf{k},\mathbf{j}} &= \sum_{\mathbf{n}} f_{\mathbf{n},\mathbf{j}}\, e^{\frac{-i2\pi\mathbf{n}\cdot\mathbf{k}}{N}},\ \forall \mathbf{k} \in \mathcal{N}^d, \\ \iff f_{\mathbf{n},\mathbf{j}} &= \frac{1}{N^d} \sum_{\mathbf{k}} \hat{f}_{\mathbf{k},\mathbf{j}}\, e^{\frac{i2\pi\mathbf{n}\cdot\mathbf{k}}{N}},\ \forall \mathbf{n} \in \mathcal{N}^d. \end{aligned} \tag{4.4}$$

Similarly, we introduce the interpolation function of the discrete solution u_h such that

$$\tilde{u}_h(\underline{x}) = \frac{1}{N^d} \sum_{\mathbf{k}} \Bigg(\sum_{\mathbf{j}} \hat{u}_{\mathbf{k},\mathbf{j}}\, e^{\frac{i2\pi(\underline{x} - h\underline{\xi}_{\mathbf{j}})\cdot \mathbf{k}}{L}} \sum_{\mathbf{n}} \varphi_{\mathbf{n},\mathbf{j}}(\underline{x}) \Bigg),$$

where

$$\hat{u}_{\mathbf{k},\mathbf{j}} = \sum_{\mathbf{n}} u_{\mathbf{n},\mathbf{j}}\, e^{-\frac{i2\pi\mathbf{n}\cdot\mathbf{k}}{N}},\ \forall \mathbf{k} \in \mathcal{N}^d. \tag{4.5}$$

For all $\mathbf{j} \in \mathcal{R}^d$ we have

$$\tilde{u}_{\mathbf{n},\mathbf{j}} := \tilde{u}_h\big((\xi_{\mathbf{j}} + \mathbf{n})h\big) = \frac{1}{N^d} \sum_{\mathbf{k}} \hat{u}_{\mathbf{k},\mathbf{j}}\, e^{\frac{i2\pi\mathbf{n}\cdot\mathbf{k}}{N}},\ \forall \mathbf{n} \in \mathcal{N}^d, \tag{4.6}$$

and, due to Eqs. (4.6) and (4.5), as well as Property 4.1, we retrieve

$$u_{\mathbf{n},\mathbf{j}} = \tilde{u}_{\mathbf{n},\mathbf{j}}, \quad \forall \mathbf{n} \in \mathcal{N}^d, \forall \mathbf{j} \in \mathcal{R}^d.$$

Then, if we use a similar approach to the one proposed for the one-dimensional case, we can show that Eq. (4.2) is equivalent to solve

$$\boldsymbol{\mathcal{S}}_{\mathbf{k}}\, \underline{\hat{U}}_{\mathbf{k}} = \underline{\hat{F}}_{\mathbf{k}}, \qquad \forall \mathbf{k} \in \mathcal{N}^d,$$

where $\underline{\hat{U}}_{\mathbf{k}}(\mathbf{j}) = \hat{u}_{\mathbf{k},\mathbf{j}}$ and $\underline{\hat{F}}_{\mathbf{k}}(\mathbf{j}) = \hat{f}_{\mathbf{k},\mathbf{j}}$, and where the matrix $\boldsymbol{\mathcal{S}}_{\mathbf{k}}$ is given by Proposition 4.2 below. As a straightforward generalisation of the one-dimensional case, the main steps of the algorithm to solve (4.2) read:

1. Perform a DFT of the discrete source term for each $\mathbf{j}$ in $\mathcal{R}^d$

$$\hat{f}_{\mathbf{k},\mathbf{j}} = \sum_{\mathbf{n}} f_{\mathbf{n},\mathbf{j}}\, e^{\frac{-i2\pi\mathbf{n}\cdot\mathbf{k}}{N}},\ \forall \mathbf{k} \in \mathcal{N}^d;$$

2. Solve for each frequency $\mathbf{k} \in \mathcal{N}^d$

$$\boldsymbol{\mathcal{S}}_{\mathbf{k}}\, \underline{\hat{U}}_{\mathbf{k}} = \underline{\hat{F}}_{\mathbf{k}}; \tag{4.7}$$

3. Perform an inverse DFT for each $\mathbf{j}$ in $\mathcal{R}^d$

$$u_{\mathbf{n},\mathbf{j}} = \frac{1}{N^d} \sum_{\mathbf{k}} \hat{u}_{\mathbf{k},\mathbf{j}}\, e^{\frac{i2\pi\mathbf{n}\cdot\mathbf{k}}{N}},\ \forall \mathbf{n} \in \mathcal{N}^d.$$

We emphasise that the multi-dimensional DFT is implemented recursively from the one-dimensional DFT. Furthermore, Step 2 must be implemented with care. In fact, one possibility is to compute the pseudo-inverse of the symbol $\boldsymbol{\mathcal{S}}_{\mathbf{k}}$ for each frequency, and then perform a matrix-vector multiplication to obtain $\underline{\hat{U}}_{\mathbf{k}}$. Note, however, that such an algorithm would imply a substantial computational cost. To give an idea, in 2D the matrix inversion would require $O(R^6)$ operations ($O(R^{3d})$ in general), whereas a matrix-vector multiplication requires $O(R^4)$ operations. In what follows, we propose a method that does not require matrix inversions and drastically reduces the complexity of the algorithm, based on SVD and tensorisation.

For the sake of conciseness, we denote $\boldsymbol{\mathcal{N}}_k = \boldsymbol{\mathcal{M}}^{-1}\boldsymbol{\mathcal{A}}_k$, with $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{A}}_k$ associated with the 1D case, and defined in Eq. (3.13). We can also rewrite the one-dimensional symbol $\boldsymbol{\mathcal{S}}_k := \rho\,\mathbf{I} + \boldsymbol{\mathcal{N}}_k$, with $\mathbf{I}$ the identity matrix of size $R \times R$. The result below has been proved in [11].

Proposition 4.2.

$$\boldsymbol{\mathcal{S}}_{\mathbf{k}}(\boldsymbol{\ell}, \mathbf{m}) = \rho \prod_{r=1}^{d} \delta_{l_r\, m_r} + \sum_{r=1}^{d} \boldsymbol{\mathcal{N}}_{k_r}(\ell_r, m_r) \prod_{q \neq r} \delta_{\ell_q\, m_q}. \tag{4.8}$$

For example, when $d = 2$, the symbol reads

$$\boldsymbol{\mathcal{S}}_{\mathbf{k}}(\boldsymbol{\ell}, \mathbf{m}) = \rho\, \delta_{l_1\, m_1}\, \delta_{l_2\, m_2} + \boldsymbol{\mathcal{N}}_{k_1}(\ell_1, m_1)\, \delta_{\ell_2\, m_2} + \boldsymbol{\mathcal{N}}_{k_2}(\ell_2, m_2)\, \delta_{\ell_1\, m_1}.$$

This result is a direct consequence of the use of the Gauss-Lobatto nodes for quadrature formulae. Due to Proposition 4.2, it is possible to compute the eigenvalues and eigenvectors of the symbol $\boldsymbol{\mathcal{S}}_{\mathbf{k}}$ starting from the eigenvalues and eigenvectors of the matrix $\boldsymbol{\mathcal{N}}_k$. Given $\mathbf{k} \in \mathcal{N}^d$, we denote by $\left\{\lambda_{\mathbf{k},\mathbf{i}}, \underline{V}_{\mathbf{k},\mathbf{i}}\right\}_{\mathbf{i}\in\mathcal{R}^d}$ the set of eigenvalues and eigenvectors of $\boldsymbol{\mathcal{S}}_{\mathbf{k}}$ such that

$$\boldsymbol{\mathcal{S}}_{\mathbf{k}}\, \underline{V}_{\mathbf{k},\mathbf{i}} = \lambda_{\mathbf{k},\mathbf{i}}\, \underline{V}_{\mathbf{k},\mathbf{i}}, \quad \forall \mathbf{k} \in \mathcal{N}^d,\ \forall \mathbf{i} \in \mathcal{R}^d.$$

When $d = 2$, they are given in [11] and read

$$\begin{cases} \lambda_{\mathbf{k},\mathbf{i}} = \rho + \nu_{k_1,i_1} + \nu_{k_2,i_2}, \\ \underline{V}_{\mathbf{k},\mathbf{i}} = \underline{W}_{k_1,i_1} \otimes \underline{W}_{k_2,i_2}, \end{cases}$$

where $\otimes$ denotes the tensor product and $\{\nu_{k,i}, \underline{W}_{k,i}\}$ are the eigenvalues and corresponding eigenvectors of $\boldsymbol{\mathcal{N}}_{k_r}$. This tensorial structure is in fact general, as it is stated in the following Corollary (whose proof is very algebraic but simple, therefore it is omitted).

Corollary 4.3. *Let $\left\{\nu_{k_r,i_r}, \underline{W}_{k_r,i_r}\right\}_{i_r\in\mathcal{R}}$ be the eigenvalues and eigenvectors of $\boldsymbol{\mathcal{N}}_{k_r}$, $r \in \{1, 2, .., d\}$. Then,*

$$\begin{cases} \lambda_{\mathbf{k},\mathbf{i}} = \rho + \displaystyle\sum_{r=1}^{d} \nu_{k_r,i_r}, \quad \forall \mathbf{i} \in \mathcal{R}^d, \\ V_{\mathbf{k},\mathbf{i}}(\mathbf{m}) = \displaystyle\prod_{r=1}^{d} W_{k_r,i_r}(m_r), \quad \forall \mathbf{i} \in \mathcal{R}^d, \end{cases}$$

are the eigenvalues and corresponding orthogonal eigenvectors of $\boldsymbol{\mathcal{S}}_{\mathbf{k}}$.

As a result of Eq. (4.3), the solution $\hat{\underline{U}}_{\mathbf{k}}$ in the frequency domain can be computed by means of an optimised algorithm that we give henceforth. It is based on tensorisation and on the use of internal variables.

Inversion of the symbol by SVD. As for the one-dimensional case, Eq. (4.7) can be rewritten, for any frequency $\mathbf{k} \in \mathcal{N}^d$, as

$$\hat{\underline{U}}_{\mathbf{k}} = \boldsymbol{\mathcal{S}}_{\mathbf{k}}^{+}\, \hat{\underline{F}}_{\mathbf{k}} := \sum_{\mathbf{p}} \lambda_{\mathbf{k},\mathbf{p}}^{+}\, \underline{V}_{\mathbf{k},\mathbf{p}}\, \underline{V}_{\mathbf{k},\mathbf{p}}^{*}\, \boldsymbol{\mathcal{M}}_d\, \hat{\underline{F}}_{\mathbf{k}}, \tag{4.9}$$

where $\boldsymbol{\mathcal{M}}_d$ denotes the mass matrix in d-dimensions and it can be rewritten as

$$\boldsymbol{\mathcal{M}}_d(\mathbf{i}, \mathbf{m}) = \prod_{r=1}^{d} \boldsymbol{\mathcal{M}}(i_r, m_r).$$

Equation (4.9) can be rewritten as

$$\begin{aligned}\hat{U}_{\mathbf{k}}(\mathbf{j}) &= \sum_{\mathbf{p}} \sum_{\mathbf{r}} \lambda^{+}_{\mathbf{k},\mathbf{p}} V_{\mathbf{k},\mathbf{p}}(\mathbf{j})\, \overline{V_{\mathbf{k},\mathbf{p}}(\mathbf{r})}\, \boldsymbol{\mathcal{M}}_d(\mathbf{r}) \hat{F}_{\mathbf{k}}(\mathbf{r}) \\ &= \sum_{\mathbf{p}} \lambda^{+}_{\mathbf{k},\mathbf{p}} V_{\mathbf{k},\mathbf{p}}(\mathbf{j}) \sum_{\mathbf{r}} \overline{V_{\mathbf{k},\mathbf{p}}(\mathbf{r})}\, \boldsymbol{\mathcal{M}}_d(\mathbf{r}) \hat{F}_{\mathbf{k}}(\mathbf{r}).\end{aligned} \tag{4.10}$$

From Eq. (4.10) we see that the total number of operations required to compute all the components of $\underline{\hat{U}}_{\mathbf{k}}$ is $O(R^{3d})$, that is comparable to the cost of a symbol inversion by standard algorithms (e.g. by Gaussian elimination). For example, in 3 dimensions this inversion would require $O(R^9)$ operations and, considering that typical values of R are larger than 3 in our applications, this prevents the method to be used naively. Opportunely, we have shown in Corollary 4.3 that the eigenvalues and eigenvectors in higher dimensions can be directly derived by tensorisation from those in 1D. This will be used to define an optimised algorithm. First, we introduce two intermediate variables, and we split Eq. (4.10) into three main operations:

- We define $\alpha_{\mathbf{k}}(\mathbf{p})$ as

$$\alpha_{\mathbf{k}}(\mathbf{p}) = \sum_{\mathbf{r}} \overline{V_{\mathbf{k},\mathbf{p}}(\mathbf{r})}\, \boldsymbol{\mathcal{M}}_d(\mathbf{r}) \hat{F}_{\mathbf{k}}(\mathbf{r}); \tag{4.11}$$

- Then, we compute the scalar coefficients $\gamma_{\mathbf{k}}(\mathbf{p})$ as

$$\gamma_{\mathbf{k}}(\mathbf{p}) \coloneqq \lambda^{+}_{\mathbf{k},\mathbf{p}}\, \alpha_{\mathbf{k}}(\mathbf{p});$$

- Finally, the components $\hat{U}_{\mathbf{k}}(\mathbf{j})$ are retrieved as

$$\hat{U}_{\mathbf{k}}(\mathbf{j}) = \sum_{\mathbf{p}} \gamma_{\mathbf{k}}(\mathbf{p})\, V_{\mathbf{k},\mathbf{p}}(\mathbf{j}). \tag{4.12}$$

Now, we can take advantage of the properties of the eigenvectors $V_{\mathbf{k},\mathbf{p}}(\mathbf{j})$. Due to Eq.(4.3), we can reduce the two multi-dimensional sums in Eq. (4.11) and Eq. (4.12) in successions of one-dimensional sums. In order to avoid cumbersome expressions, we consider $d = 2$ henceforth. Note, however, that the extension of our analysis for $d > 2$ is straightforward. The intermediate coefficient $\alpha_{\mathbf{k}}(\mathbf{p})$ is computed, for each $\mathbf{p}$, in $O(R)$ operations by

$$\begin{aligned}\alpha_{\mathbf{k}}(\mathbf{p}) = \alpha_{\mathbf{k}}(p_1, p_2) &\coloneqq \sum_{r_1, r_2 \in \mathcal{R}} \boldsymbol{\mathcal{M}}_d(r_1, r_2) \hat{F}_{\mathbf{k}}(r_1, r_2)\, \overline{V_{k_1,p_1}(r_1)}\, \overline{V_{k_2,p_2}(r_2)} \\ &= \sum_{r_2 \in \mathcal{R}} \overline{V_{k_2,p_2}(r_2)} \sum_{r_1 \in \mathcal{R}} \boldsymbol{\mathcal{M}}_d(r_1, r_2) \hat{F}_{\mathbf{k}}(r_1, r_2)\, \overline{V_{k_1,p_1}(r_1)} \\ &= \sum_{r_2 \in \mathcal{R}} \overline{V_{k_2,p_2}(r_2)}\, \beta_{\mathbf{k}}(p_1, r_2),\end{aligned} \tag{4.13}$$

where $\beta_{\mathbf{k}}(\mathbf{p})$ is computed in $O(R)$ operations and is given by

$$\beta_{\mathbf{k}}(\mathbf{p}) = \beta_{\mathbf{k}}(p_1, p_2) \coloneqq \sum_{r_1 \in \mathcal{R}} \boldsymbol{\mathcal{M}}_d(r_1, p_2)\, \hat{F}_{\mathbf{k}}(r_1, p_2) \overline{V_{k_1,p_1}(r_1)}.$$

Analogously, $\hat{U}_{\mathbf{k}}(\mathbf{j})$, for any given $\mathbf{j}$, is obtained in $O(R)$ operations as follows:

$$\begin{aligned}\hat{U}_{\mathbf{k}}(\mathbf{j}) = \hat{u}_{\mathbf{k}}(j_1, j_2) &= \sum_{p_1,p_2\in\mathcal{R}} \gamma_{\mathbf{k}}(p_1,p_2)\, V_{k_1,p_1}(j_1)\, V_{k_2,p_2}(j_2) \\ &= \sum_{p_2\in\mathcal{R}} V_{k_2,p_2}(j_2) \sum_{p_1\in\mathcal{R}} \gamma_{\mathbf{k}}(p_1,p_2)\, V_{k_1,p_1}(j_1) \\ &= \sum_{p_2\in\mathcal{R}} V_{k_2,p_2}(j_2)\, \mu_{\mathbf{k}}(j_1,p_2),\end{aligned} \tag{4.14}$$

with $\mu_{\mathbf{k}}(\mathbf{j})$ computed in $O(R)$ operations again, by

$$\mu_{\mathbf{k}}(\mathbf{j}) = \mu_{\mathbf{k}}(j_1, j_2) := \sum_{p_1\in\mathcal{R}} \gamma_{\mathbf{k}}(p_1, j_2)\, V_{k_1,p_1}(j_1).$$

Consequently, the overall complexity of the algorithm for the symbol inversion for each frequency is $O(R^4)$. For an arbitrary dimension d, the complexity is $O(R^{d+1})$. Therefore, in 3D this represents $O(R^5)$ less operations than the naive approach, that is a tremendous gain.

4.2. Neumann or Dirichlet boundary conditions. The algorithm in the d-dimensional case can be directly deduced by tensorisation also when homogeneous Dirichlet or Neumann conditions are imposed on the boundaries. In order to perform the DFT, the source term is periodically and symmetrically extended in d directions. This leads to the evaluation of a computational domain that is potentially 2^d times larger than the original domain. However, due to the symmetry properties of the extended source term, we will be able to reduce the main computations to the first $(N+1)^d$ frequencies. The extended discrete source term $\{f_{\mathbf{n},\mathbf{j}}\}$ satisfies, for all $r \in \{1, \cdots, d\}$,

$$\begin{cases} f_{\mathbf{n}+\mathbf{n}_r,\mathbf{j}} = \pm f_{\mathbf{n},\mathbf{j}+\mathbf{j}_r}, & j_r \neq 0, \quad \mathbf{n}_r = \mathbf{e}_r(2N-1-n_r),\, \mathbf{j}_r = \mathbf{e}_r(R-j_r), \\ f_{\mathbf{n}+\mathbf{n}_r,\mathbf{j}} = \pm f_{\mathbf{n},\mathbf{j}}, & j_r = 0, \quad \mathbf{n}_r = \mathbf{e}_r(2N-n_r), \end{cases} \tag{4.15}$$

where $\mathbf{e}_r$ is the $r-$th vector of the canonical basis of $\mathbb{R}^d$ and (n_r, j_r) is such that

$$\mathbf{n}+\mathbf{n}_r \in \{0, \cdots, 2N-1\}^d, \quad \mathbf{j}+\mathbf{j}_r \in \{0, \cdots R-1\}^d,$$

and where the even extension corresponds to Neumann BCs, whereas the odd extension is associated with Dirichlet BCs, as before.

Lemma 4.4. *The DFT of the extended sequence $\{f_{\mathbf{n},\mathbf{j}}\}$ satisfies, $\forall r \in \{1, \cdots, d\}$,*

$$\begin{cases} \hat{f}_{\mathbf{k}+\mathbf{k}_r,\mathbf{j}} = \pm \hat{f}_{\mathbf{k},\mathbf{j}+\mathbf{j}_r}\, e^{\frac{-i\pi k}{N}}, & \text{if } j_r \neq 0, \quad \mathbf{k}_r = \mathbf{e}_r(2N-k_r),\, \mathbf{j}_r = \mathbf{e}_r(R-j_r), \\ \hat{f}_{\mathbf{k}+\mathbf{k}_r,\mathbf{j}} = \pm \hat{f}_{\mathbf{k},\mathbf{j}}, & \text{if } j_r = 0, \quad \mathbf{k}_r = \mathbf{e}_r(2N-k_r), \end{cases}$$

with (k_r, j_r) such that

$$\mathbf{k}+\mathbf{k}_r \in \{0, \cdots, 2N-1\}^d, \quad \mathbf{j}+\mathbf{j}_r \in \{0, \cdots R-1\}^d$$

and with the usual convention for the plus and minus signs.

Therefore, given a direction $i \in \{1, 2, .., d\}$, the value of the DFT coefficients for the last $N-1$ coefficients in that direction are directly retrieved from the first N coefficients, and the relationship depends on the frequency and the Gauss-Lobatto point taken into account.
Eventually, due to Lemma 4.4 and Proposition 3.3, the solution $\hat{u}_{\mathbf{k},\mathbf{j}}$ inherits the same properties as $\hat{f}_{\mathbf{k},\mathbf{j}}$. Therefore, the main steps of the optimised algorithm to solve (4.2), when homogeneous Neumann or Dirichlet conditions are considered, are

1. Perform a DFT of the discrete source term for each $\mathbf{j}$ in $\mathcal{R}$

$$\hat{f}_{\mathbf{k},\mathbf{j}} = \sum_{\mathbf{n} \in \{0,..,(2N-1)\}^d} f_{\mathbf{n},\mathbf{j}}\, e^{\frac{-i\pi \mathbf{n}\cdot\mathbf{k}}{N}},\ \forall \mathbf{k} \in (\mathcal{N} \cup \{N\})^d,$$

   using Eq. (4.15) for evaluating $f_{\mathbf{n},\mathbf{j}}$ when $\mathbf{n} \notin \mathcal{N}^d$;
2. Solve for each frequency

$$\boldsymbol{\mathcal{S}}_{\mathbf{k}}\, \underline{\hat{U}}_{\mathbf{k}} = \underline{\hat{F}}_{\mathbf{k}},\ \forall \mathbf{k} \in (\mathcal{N} \cup \{N\})^d;$$

3. Perform an inverse DFT for each $\mathbf{j}$ in $\mathcal{R}$

$$u_{\mathbf{n},\mathbf{j}} = \frac{1}{(2N)^d} \sum_{\mathbf{n} \in \{0,..,(2N-1)\}^d} \hat{u}_{\mathbf{k},\mathbf{j}}\, e^{\frac{i\pi \mathbf{n}\cdot\mathbf{k}}{N}},\ \forall \mathbf{n} \in (\mathcal{N} \cup \{N\})^d,$$

   where, thanks to Lemma 4.4,

$$\begin{cases} \hat{u}_{\mathbf{k}+\mathbf{k}_r,\mathbf{j}} = \pm \hat{u}_{\mathbf{k},\mathbf{j}+\mathbf{j}_r}\, e^{\frac{-i\pi k}{N}}, & \text{if } \ j_r \neq 0, \quad \mathbf{k}_r = \mathbf{e}_r(2N-k_r),\, \mathbf{j}_r = \mathbf{e}_r(R-j_r), \\ \hat{u}_{\mathbf{k}+\mathbf{k}_r,\mathbf{j}} = \pm \hat{u}_{\mathbf{k},\mathbf{j}}, & \text{if } \ j_r = 0, \quad \mathbf{k}_r = \mathbf{e}_r(2N-k_r), \end{cases}$$

   with the usual convention for signs, and with (k_r, j_r) such that

$$\mathbf{k} + \mathbf{k}_r \in \{0, \cdots, 2N-1\}^d, \quad \mathbf{j} + \mathbf{j}_r \in \{0, \cdots R-1\}^d.$$

It is possible to do an efficient implementation of Steps 1 and 3 that never requires the storage and the pre-computation of the odd or even extensions. On the contrary, Eq. (4.15) is used on-the-flight, for an improved performance in terms of storage and memory access time.

5. Numerical Results and complexity.

5.1. Convergence analysis. In order to demonstrate that the HOFFT method preserves the order of convergence of the underlying SEM discretisation, the method is tested against an exact solution of Problem (2.2) with $\rho = 0$ and homogeneous Dirichlet boundary conditions, that reads

$$u(\underline{x}) = \prod_{i=1}^{d} (1-x_i)^3\, x_i^3\, \cos(k\, x_i), \quad \forall \underline{x} \in \Omega = [0,1]^d,\ d = 2, 3,$$

with $k = 10$. The expression of the source term f is directly obtained from (2.2) as $f(\underline{x}) = -\Delta u(\underline{x})$. Figures 4 and 5 show the convergence error between the exact solution and the output of the HOFFT algorithm in relative $L^2(\Omega)$ and $H^1(\Omega)$ norm w.r.t. the space step h and for different values of order R, in two and three dimensions, respectively. The convergence rate of the error in $L^2(\Omega)$ and $H^1(\Omega)$-norm corresponds

Figure 4 Convergence of the two-dimensional scheme w.r.t. the space step h, for different values of order R. Density $\rho = 0$, homogeneous Dirichlet boundary conditions. Left: Relative $L^2(\Omega)$-error. Right: Relative $H^1(\Omega)$-error.

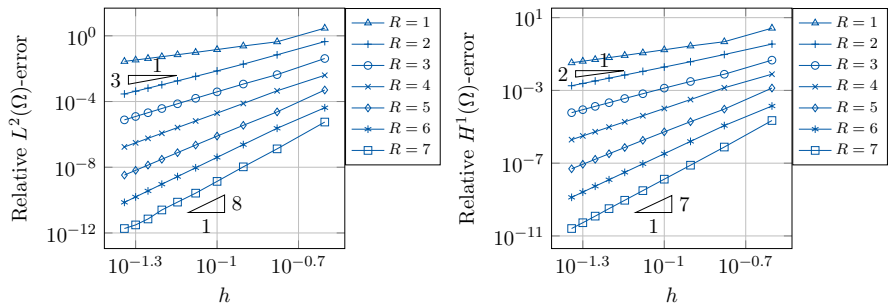
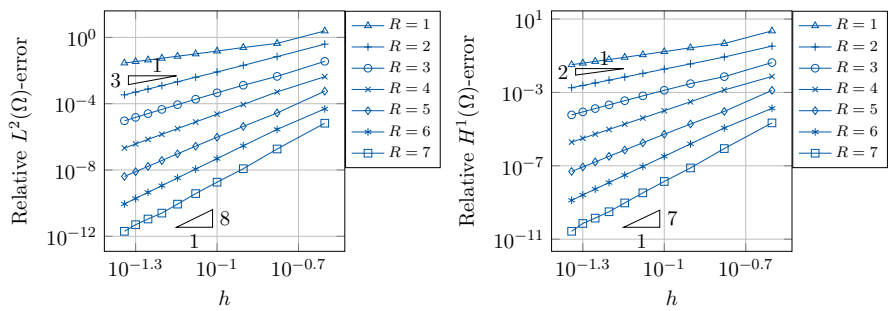


Figure 5 Convergence of the three-dimensional scheme w.r.t. the space step h, for different values of order R. Density $\rho = 0$, homogeneous Dirichlet boundary conditions. Left: Relative $L^2(\Omega)$-error. Right: Relative $H^1(\Omega)$-error.



to the chosen order R. For the sake of completeness, we show in Figures 6 and 7 the convergence error between the exact solution and the output of our algorithm w.r.t. the space step h in two dimensions and choosing density $\rho = 1$, with homogeneous Dirichlet or Neumann boundary conditions, respectively.

5.2. Complexity of the algorithm. As it has been stated in the previous sections, this algorithm is remarkably efficient in multiple dimensions, for several reasons. First, an efficient algorithm – in $O(R^{d+1})$ operations – has been deduced for Step 2 of the algorithm, that relies on the tensorisation properties of the underlying finite element discretisation. Secondly, Steps 1 and 3 can be performed with the FFT. Since the d-dimensional FFT corresponds to a 1D FFT having the size of the product of the dimensions, its complexity is $d(\log N)N^d$ (if the domain is divided into the same number of elements N for each direction). Consequently, the implementation of the FFT causes a further, sensible reduction of the computational cost.
We emphasise that this method is not limited to the solution of the Poisson-like problem with periodic boundary conditions. In fact, we have shown that, in case of Dirichlet or Neumann BCs, it is possible to employ HOFFT after extending symmetrically or anti-symmetrically the computational domain – i.e. the number of frequencies considered – in each dimension. However, thanks to the symmetry properties of the sequence considered and the Fourier transform, most operations only involve the first $N + 1$ spatial frequencies (in each direction), and therefore the computational over-

Figure 6 Convergence of the two-dimensional scheme w.r.t. the space step h, for different values of order R. Density $\rho = 1$, homogeneous Dirichlet boundary conditions. Left: Relative $L^2(\Omega)$-error. Right: Relative $H^1(\Omega)$-error.

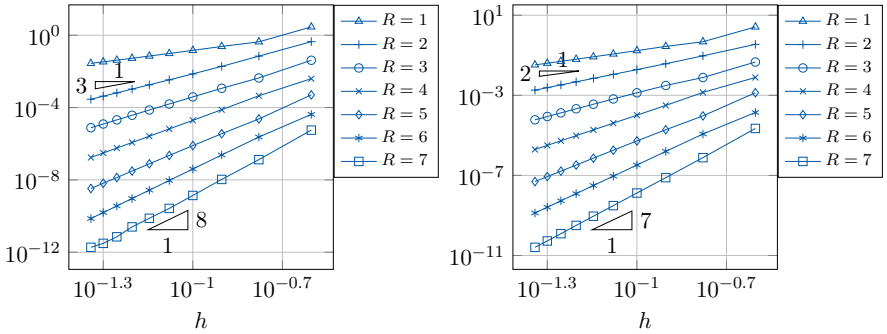
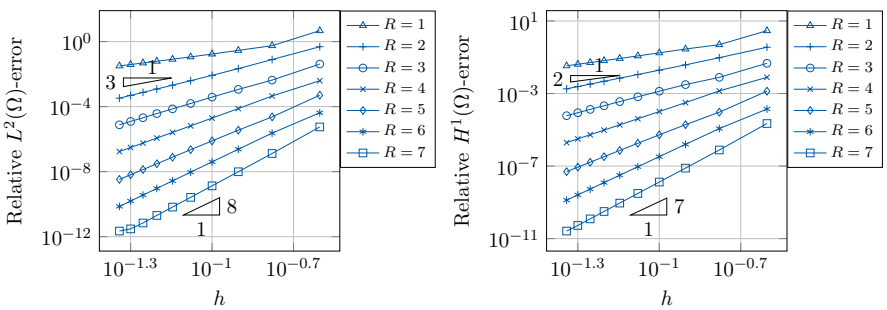


Figure 7 Convergence of the two-dimensional scheme w.r.t. the space step h, for different values of order R. Density $\rho = 1$, homogeneous Neumann boundary conditions. Left: Relative $L^2(\Omega)$-error. Right: Relative $H^1(\Omega)$-error.



cost of the algorithm is reduced. Besides, Steps 1 and 3 are disjoint for each order, whereas the operations involved in the inversion of the symbol (Step 2) are disjoint for each frequency. Therefore, the algorithm is extremely well-adapted to parallelisation. In particular, Steps 1 and 3 can be run in parallel for each order, whereas Step 2 can be performed in parallel for each frequency.

We solved the same test problem proposed in Section 5.1 with $N = 64$, $R = 6$, $\rho = 0$ and homogeneous Dirichlet boundary conditions on a 12-cores workstation (cores at 2.7 GHz and 64 GB of RAM at 1867 MHz) using a straightforward shared-memory multi-threaded parallelisation of the algorithm. We observed that the computational time was divided by approximately 3.2.

Figure 8 shows the computational cost of the sequential solution of the problem with homogeneous Dirichlet Boundary conditions in 3D on a 4-cores workstation (cores at 3.1 GHz and 16 GB of RAM at 1867 MHz). The computational time is in accordance with the expected complexity with respect to the number of elements N and the order R. Finally, note that the memory storage is minimised and, apart from the solution vector, only the eigenvector and eigenvalues of the matrices $\boldsymbol{\mathcal{N}}_k$ – that correspond to 1D symbols – must be stored for each scalar frequency k.

6. Discussion. In this article we have presented a novel, efficient numerical scheme, based on the use of the discrete Fourier transform, to solve the Poisson equa-

Figure 8 Computational cost of the three-dimensional scheme. Left: comparison for different values of elements. Right: comparison for different values of order R.

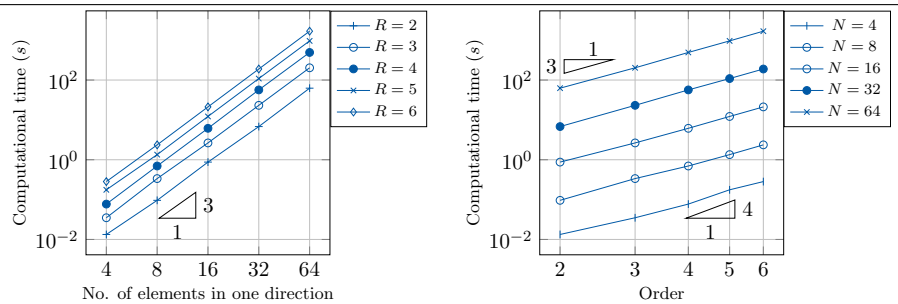
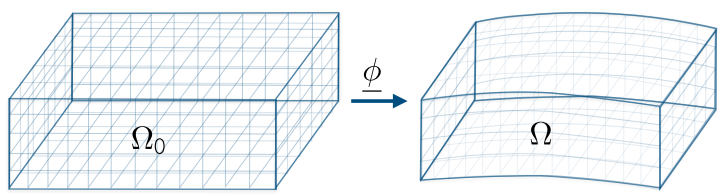


Figure 9 Computational domain Ω seen as the result of a smooth deformation map $\underline{\phi}$ applied on Ω_0.



tion discretised with high-order SEM. As we have already mentioned above, due to the underlying SE discretisation, this method is restricted a-priori to the strong assumptions on the computational domain Ω, that has to be a rectangle in 2D or a rectangular parallelepiped in 3D. However, this does not represent a limitation for the specific problems that we aim to solve, i.e. the resolution of the elastodynamic equations in incompressible solids via a penalisation approach. In more detail, the strategy proposed in [9] consists in penalising the divergence-free constraint at each discrete time t of the wave propagation. We solve

$$-\alpha\Delta p(t) = \operatorname{div}\underline{y}(t), \quad \text{in } \Omega, \quad \underline{\nabla} p(t)\cdot\underline{n} = 0, \quad \text{on } \partial\Omega, \tag{6.1}$$

where α is a small positive penalisation parameter and $\underline{y}(t)$ is the known displacement field. In this context, we can consider a computational domain Ω that is more general than a regular quadrilateral or parallelepiped. The only requirement is that it can be mapped to uniform meshes of a rectangular parallelepiped Ω_0 via a continuous transformation $\underline{\phi}$ (see Fig. 9). In more detail, let us consider, instead of a laplacian operator in (6.1), a more general operator such as

$$-\alpha \operatorname{div}\underline{\underline{A}}\,\underline{\nabla} p(t) = \operatorname{div}\underline{y}(t) \quad \text{in } \Omega, \quad J^{-1}\underline{\underline{F}}^T\underline{\underline{F}}\,\underline{\nabla} p(t)\cdot\underline{n} = 0, \quad \text{on } \partial\Omega,$$

where $\underline{\underline{A}} = J^{-1}\underline{\underline{F}}^T\underline{\underline{F}}$, with $\underline{\underline{F}}$ the gradient of the deformation map $\underline{\phi}$ and J Jacobian of the deformation. Then, in variational formulation, applying the Green formula we obtain

$$\int_\Omega \alpha\, A\,\underline{\nabla} p\cdot\underline{\nabla} q\,\mathrm{d}\Omega = \alpha\int_{\Omega_0}\underline{\nabla}_{\underline{\xi}}\,(p\circ\underline{\phi})\cdot\underline{\nabla}_{\underline{\xi}}\,(q\circ\underline{\phi})\,\mathrm{d}\Omega_0 = \int_{\Omega_0} J\big((\operatorname{div}\underline{y})\circ\underline{\phi}\big)\,(q\circ\underline{\phi})\,\mathrm{d}\Omega_0. \tag{6.2}$$

Hence, we recover a homogeneous Laplace operator on a canonical computational domain, that is the standard problem suitable for the HOFFT solver proposed in this work. Note that this strategy is very specific to penalisation problems. Indeed, the matrix $\underline{\underline{A}}$ is a penalisation parameter that can be chosen so that (6.2) holds and without losing the effectiveness of the penalisation procedure introduced in [9]. Finally, one possible extension of this work would be to understand whitin what limits the strategy that we have presented is compatible with the Fourier Continuation method of [8]. The development of Fourier Continuation methods may allow the HOFFT to be used in more general configurations.

REFERENCES

[1] M. Ainsworth, *Discrete dispersion relation for hp-version finite element approximation at high wave number*, SIAM Journal on Numerical Analysis, 42 (2004), pp. 553–575.

[2] M. Ainsworth and P. Coggins, *A uniformly stable family of mixed hp-finite elements with continuous pressures for incompressible flow*, IMA journal of numerical analysis, 22 (2002), pp. 307–327.

[3] F. Amlani and O. P. Bruno, *An FC-based spectral solver for elastodynamic problems in general three-dimensional domains*, Journal of Computational Physics, 307 (2016), pp. 333 – 354.

[4] S. Börm, L. Grasedyck, and W. Hackbusch, *Introduction to hierarchical matrices with applications*, Engineering analysis with boundary elements, 27 (2003), pp. 405–422.

[5] F. Brezzi and K.-J. Bathe, *A discourse on the stability conditions for mixed finite element formulations*, Computer methods in applied mechanics and engineering, 82 (1990), pp. 27–57.

[6] F. Brezzi and R. S. Falk, *Stability of higher-order Hood–Taylor methods*, SIAM Journal on Numerical Analysis, 28 (1991), pp. 581–590.

[7] F. Brezzi and M. Fortin, *Mixed and hybrid finite element methods*, vol. 15, Springer Science & Business Media, 2012.

[8] O. P. Bruno, Y. Han, and M. M. Pohlman, *Accurate, high-order representation of complex three-dimensional surfaces via fourier continuation analysis*, Journal of Computational Physics, 227 (2007), pp. 1094 – 1125.

[9] F. Caforio and S. Imperiale, *A conservative penalisation strategy for the semi-implicit time discretisation of the incompressible elastodynamics equation*, Advanced Modeling and Simulation in Engineering Sciences, 5 (2018), p. 30.

[10] G. Cohen, *Higher-Order Numerical Methods for Transient Wave Equations*, Scientific Computation, Springer Berlin Heidelberg, 2001.

[11] G. Cohen, *Higher-Order Numerical Methods for Transient Wave Equations*, Scientific computation, Springer, 2002.

[12] G. Cohen and S. Fauqueux, *Mixed spectral finite elements for the linear elasticity system in unbounded domains*, SIAM Journal on Scientific Computing, 26 (2005), pp. 864–884.

[13] J.-L. Guermond, P. Minev, and J. Shen, *An overview of projection methods for incompressible flows*, Computer methods in applied mechanics and engineering, 195 (2006), pp. 6011–6045.

[14] B. Gustafsson, *Fundamentals of Scientific Computing*, Texts in Computational Science and Engineering, Springer Berlin Heidelberg, 2011.

[15] A. Iserles, *A first course in the numerical analysis of differential equations*, no. 44, Cambridge university press, 2009.

[16] D. Komatitsch and J.-P. Vilotte, *The spectral element method: an efficient tool to simulate the seismic response of 2d and 3d geological structures*, Bulletin of the seismological society of America, 88 (1998), pp. 368–392.

[17] M. Lyon and O. P. Bruno, *High-order unconditionally stable FC-AD solvers for general smooth domains ii. elliptic, parabolic and hyperbolic pdes; theoretical considerations*, Journal of Computational Physics, 229 (2010), pp. 3358 – 3381.

[18] Y. Maday and A. T. Patera, *Spectral element methods for the incompressible Navier-Stokes equations*, in State-of-the-art surveys on computational mechanics (A90-47176 21-64). New York, American Society of Mechanical Engineers. Research supported by DARPA., 1989, pp. 71–143.

[19] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical mathematics*, vol. 37, Springer Science

& Business Media, 2010.

[20] A. P. Sarvazyan, O. V. Rudenko, and W. L. Nyborg, *Biomedical applications of radiation force of ultrasound: historical roots and physical basis*, Ultrasound in medicine & biology, 36 (2010), pp. 1379–1394.

[21] C. Van Loan, *Computational frameworks for the fast Fourier transform*, vol. 10, Siam, 1992.