Refined isogeometric analysis for generalized Hermitian eigenproblems

Ali Hashemian[a,*], David Pardo[b,a,c], Victor M. Calo[d,e]

[a] *BCAM – Basque Center for Applied Mathematics, Bilbao, Basque Country, Spain*
[b] *University of the Basque Country UPV/EHU, Leioa, Basque Country, Spain*
[c] *Ikerbasque – Basque Foundation for Sciences, Bilbao, Basque Country, Spain*
[d] *Faculty of Science and Engineering, Curtin University, Perth, Australia*
[e] *Commonwealth Scientific and Industrial Research Organisation (CSIRO), Perth, Australia*

Abstract

We use refined isogeometric analysis (rIGA) to solve generalized Hermitian eigenproblems ($\mathbf{Ku} = \lambda\mathbf{Mu}$). rIGA conserves the desirable properties of maximum-continuity isogeometric analysis (IGA) while it reduces the solution cost by adding zero-continuity basis functions, which decrease the matrix connectivity. As a result, rIGA enriches the approximation space and reduces the interconnection between degrees of freedom. We compare computational costs of rIGA versus those of IGA when employing a Lanczos eigensolver with a shift-and-invert spectral transformation. When all eigenpairs within a given interval $[\lambda_s, \lambda_e]$ are of interest, we select several shifts $\sigma_k \in [\lambda_s, \lambda_e]$ using a spectrum slicing technique. For each shift σ_k, the factorization cost of the spectral transformation matrix $\mathbf{K} - \sigma_k\mathbf{M}$ controls the total computational cost of the eigensolution. Several multiplications of the operator matrix $(\mathbf{K} - \sigma_k\mathbf{M})^{-1}\mathbf{M}$ by vectors follow this factorization. Let p be the polynomial degree of the basis functions and assume that IGA has maximum continuity of $p - 1$. When using rIGA, we introduce C^0 separators at certain element interfaces to minimize the factorization cost. For this setup, our theoretical estimates predict computational savings to compute a fixed number of eigenpairs of up to $\mathcal{O}(p^2)$ in the asymptotic regime, that is, large problem sizes. Yet, our numerical tests show that for moderate-size eigenproblems, the total observed computational cost reduction is $\mathcal{O}(p)$. In addition, rIGA improves the accuracy of every eigenpair of the first N_0 eigenvalues and eigenfunctions, where N_0 is the total number of modes of the original maximum-continuity IGA discretization.

Keywords: Refined isogeometric analysis; generalized Hermitian eigenproblem; Lanczos eigensolver; spectral transformation; shift-and-invert approach.

1. Introduction

Hughes et al. [1] introduced isogeometric analysis (IGA), a widely used numerical technique to approximate solutions to partial differential equations (PDEs). IGA addresses many scientific and engineering problems (c.f., [2–5]). Some of its specific applications are in, e.g., structural analysis [6–8], fluid mechanics [9–12], fluid–structure interactions [13, 14], chemical physics [15–17], and electromagnetics [18, 19]. IGA uses spline basis functions, which are standard in computer-aided design (CAD), as shape functions of finite element analysis (FEA). These functions can have high continuity (up to C^{p-1}, where p is the polynomial order of spline bases) across the element interfaces.

Compared to the minimal interconnection of elements in traditional FEA, maximum-continuity IGA discretizations strengthen the interconnection between elements. This increased interconnectivity degrades the performance of sparse direct solvers (see, e.g., [20–22]). Garcia et al. [23] introduced the refined isogeometric analysis (rIGA) to ameliorate this performance degradation and to exploit the recursive partitioning capability of multifrontal direct solvers [24]. rIGA preserves the desirable properties of maximum-continuity IGA discretizations while significantly reduces the computational cost of the solution by partitioning the computational domain into macroelement blocks that are weakly interconnected by C^0 separators. As a result, the matrix factorization (e.g., LU or Cholesky) step has

*Corresponding author
Email address: ahashemian@bcamath.org (Ali Hashemian)

a lower computational cost. The performance of the rIGA framework on preconditioned conjugate gradient solvers as well as its applicability to mechanical and electromagnetic problems is also studied in [25, 26].

The application of IGA in eigenanalysis is a well-studied topic in the literature (see, e.g., [27–30]). Improving the efficiency of the system integration and accuracy of the spectral approximation of the IGA approach are also considered (see, e.g., [31–38]). Herein, we investigate the beneficial effect of using refined isogeometric analysis for solving eigenproblems. We compare the computational cost and accuracy of the resulting eigenpairs produced by both refined and maximum-continuity IGA. We first review some numerical aspects of eigenanalysis to perform a detailed comparison of the methods and their results.

Eigenanalysis is computationally expensive, especially when seeking to compute a large number of eigenpairs (i.e., eigenvalues and eigenvectors) on a multidimensional domain. Frequently used Krylov eigensolvers such as Lanczos and Arnoldi project the problem onto Krylov subspaces (see, e.g., [39, 40]). The convergence rates of these iterative algorithms degrade as the number of eigenvalues to compute in each range increases, particularly when the eigenmodes are not well separated. Since eigenvalue clustering and repetition are common in multidimensional PDEs, this leads to complications in the application of these iterative techniques. Let us consider the discrete system $\mathbf{Ku} = \lambda\mathbf{Mu}$ as a generalized Hermitian eigenproblem (GHEP), where the term *generalized* distinguishes it from the standard Hermitian eigenproblem $\mathbf{Au} = \lambda\mathbf{u}$. A well-established recommendation in the literature (e.g., [39–43]) is to first perform a spectral transformation (ST), and then solve the shifted eigenproblem $(\mathbf{K} - \sigma\mathbf{M})\mathbf{u} = (\lambda - \sigma)\mathbf{Mu}$. This spectral shift results in fast convergence when calculating eigenvalues near the shift σ. A more efficient way in eigenpairs approximation is to solve a "shift-and-invert" spectral transformed eigenproblem $(\mathbf{K} - \sigma\mathbf{M})^{-1}\mathbf{Mu} = \theta\mathbf{u}$, with $\theta = 1/(\lambda - \sigma)$. This approach preserves the separation of eigenvalues near σ to reach an accurate eigensolution. In many practical cases, we seek all or a large number of eigenpairs λ_i and $\mathbf{u}_i$ within a given (large) interval $\lambda_i \in [\lambda_s, \lambda_e]$, where the magnitude of either λ_s or λ_e can be infinite. Hence, we select several shifts $\sigma_k \in [\lambda_s, \lambda_e]$ to preserve the convergence rate for eigenvalues far from σ. We employ a "spectrum slicing technique" (see, e.g., [44]) to dynamically select σ_ks and find all eigenpairs with the true multiplicities and without incurring in unnecessary computational efforts. Section 4 provides more algorithmic details of the eigenanalysis.

The factorization of the ST matrix $\mathbf{K} - \sigma_k\mathbf{M}$ for each σ_k is a major component of the cost of eigencomputations, especially when dealing with large algebraic systems. Once we compute this factorization, the computation of the Krylov subspace requires several multiplications of the operator matrix $(\mathbf{K} - \sigma_k\mathbf{M})^{-1}\mathbf{M}$ by vectors. These multiplications consist of two steps, namely the forward/backward eliminations of the respective factors of the ST matrix, and the products of matrix $\mathbf{M}$ by vectors. Let N be the total number of degrees of freedom in the system. Using maximum-continuity IGA, the computational cost of factorization is $\mathcal{O}(N^{1.5}p^3)$ and $\mathcal{O}(N^2p^3)$ for 2D and 3D systems, respectively [20, 23]. The cost of performing forward/backward eliminations is $\mathcal{O}(Np^2)$ and $\mathcal{O}(N^{1.33}p^2)$ for 2D and 3D systems, respectively, while it is $\mathcal{O}(Np^2)$ and $\mathcal{O}(Np^3)$ for multiplying the $\mathbf{M}$ matrix by vectors in 2D and 3D cases, respectively. These costs are proportional to the number of nonzero terms of the system matrices and respective factors, expressed in terms of memory requirements in [20, 21]. The contribution of other operations towards the total computational cost is negligible compared to those of the factorization, forward/backward eliminations, and matrix–vector multiplications.

Herein, we propose to use rIGA discretizations to compute eigenpairs of generalized Hermitian eigensystems. In Section 5, we estimate the cost of computing fixed numbers of eigenpairs, and in all cases, rIGA is faster than maximum-continuity IGA. When using multifrontal direct solvers, rIGA reduces the factorization cost by up to $\mathcal{O}(p^2)$ for large systems. Also, the cost of the forward/backward elimination decreases by $\mathcal{O}(p)$, since the factorized form of the ST matrix has fewer nonzero entries in rIGA than its *smooth* IGA counterpart (see [23]). Nevertheless, the matrix multiplication of $\mathbf{M}$ by vectors is slightly more expensive when using rIGA as it has more nonzero entries than its IGA counterpart.

Our theoretical analysis shows that an improvement of $\mathcal{O}(p^2)$ in eigencomputation cost is asymptotically possible when employing an rIGA discretization. Indeed, for sufficiently large problems, the matrix factorization governs the solution cost. However, in practical moderate-size problems, the numerical tests show that rIGA reduces the eigencomputation cost by a factor of approximately $\mathcal{O}(p)$. rIGA also approximates better the first N_0 eigenpairs, where N_0 can be as large as the total number of degrees freedom in the smooth IGA discretization. The improved

accuracy is a consequence of the increased trial space, that is, the continuity reduction of the basis functions enriches the Galerkin space, delivering a better approximation than the smooth IGA version.

The organization of the remainder of this paper is as follows. Section 2 defines the problem, while Section 3 briefly revisits the notation and definitions of smooth (maximum-continuity) IGA and rIGA frameworks. Section 4 describes the eigensolution algorithm for finding the eigenpairs and Section 5 derives theoretical cost estimates of the eigensolution under IGA and rIGA discretizations. We provide implementation details in Section 6, numerical cost evaluations in Section 7, and accuracy assessments in Section 8. Finally, Section 9 draws conclusions.

2. Preliminaries

2.1. Model problem

We consider the eigensolution of the Laplace operator in the unit square:

$$\begin{aligned} &\text{Find } \lambda \in \mathbb{R}^+ \text{ and } u(\mathbf{x}) \in H_0^1(\Omega), \text{ satisfying} \\ &\begin{cases} \nabla^2 u + \lambda u = 0\,, \\ u(\mathbf{x})|_{\partial\Omega} = 0\,, \quad \mathbf{x} \in \Omega : (0,1)^d\,, \end{cases} \end{aligned} \tag{1}$$

where $d = 2$ or 3 is the space dimension. The exact eigenvalues λ^e and eigenfunctions u^e in 2D and 3D are

$$\text{2D}: \quad \lambda^e_{ij} = \pi^2\left(i^2 + j^2\right), \qquad u^e_{ij} = 2\sin(i\pi x)\sin(j\pi y)\,, \tag{2a}$$

$$\text{3D}: \quad \lambda^e_{ijk} = \pi^2\left(i^2 + j^2 + k^2\right), \quad u^e_{ijk} = 2\sqrt{2}\sin(i\pi x)\sin(j\pi y)\sin(k\pi z)\,. \tag{2b}$$

Considering the arbitrary test function $v(\mathbf{x}) \in H_0^1(\Omega)$, a weak form of (1) is

$$\int_\Omega \nabla u \cdot \nabla v \, d\Omega = \int_\Omega \lambda u v \, d\Omega\,. \tag{3}$$

Using symmetric bilinear forms

$$a(u,v) := \int_\Omega \nabla u \cdot \nabla v \, d\Omega\,, \tag{4a}$$

$$(u,v) := \int_\Omega u v \, d\Omega\,, \tag{4b}$$

we write the weak formulation as

$$a(u,v) = \lambda(u,v)\,. \tag{5}$$

Introducing a Galerkin discretization of the continuous eigenproblem leads to the following discrete form (superscript h refers to the numerical computed eigenpairs) [45],

$$a\left(u^h, v^h\right) = \lambda^h\left(u^h, v^h\right). \tag{6}$$

2.2. Eigenvalue and eigenfunction errors

We consider the exact eigenpairs in (2) and their numerical counterparts in (6) to assess the accuracy of the spectral approximation and study the errors in the eigenvalues and eigenfunctions (in L^2 and energy norms). By definition, eigenpairs of multidimensional systems are in the tensor form. In practice, however, eigensolvers typically represent them in the vectorial form. Considering a specific spectrum interval, it helps to sort the eigenvalues in the ascending order and obtain their correct multiplicity. For the i-th discrete eigenmode in this representation, using the Pythagorean eigenvalue error theorem [45], we can relate the eigenvalue error and the eigenfunction errors in L^2 and energy norms as follows [31]:

$$\lambda^h_i - \lambda^e_i + \lambda^e_i \left\|u^h_i - u^e_i\right\|^2_{L^2} = \left\|u^h_i - u^e_i\right\|^2_E\,, \tag{7}$$

where

$$\left\|u_i^h - u_i^e\right\|_{L^2}^2 = \left(u_i^h - u_i^e, u_i^h - u_i^e\right), \tag{8a}$$

$$\left\|u_i^h - u_i^e\right\|_{E}^2 = a\left(u_i^h - u_i^e, u_i^h - u_i^e\right). \tag{8b}$$

Based on the above theorem, we define for the i-th eigenmode of our spectral approximation, its normalized eigenvalue error (EVerr) and eigenfunction L^2 and energy norm errors (EFerr_{L^2} and EFerr_E, respectively) as

$$\mathrm{EVerr} := \frac{\lambda_i^h - \lambda_i^e}{\lambda_i^e}, \tag{9a}$$

$$\mathrm{EFerr}_{L^2} := \left\|u_i^h - u_i^e\right\|_{L^2}^2, \tag{9b}$$

$$\mathrm{EFerr}_{E} := \frac{\left\|u_i^h - u_i^e\right\|_{E}^2}{\lambda_i^e} = \mathrm{EVerr} + \mathrm{EFerr}_{L^2}. \tag{9c}$$

3. Refined isogeometric analysis

We first review some basic concepts of maximum-continuity IGA discretizations. For the sake of simplicity, we assume that the computational grid consists of a tensor-product mesh in $[0,1]^d$ with the same number of equally spaced elements in each direction. For descriptions of how to use tensor-product discretization relying on mapped geometries, refer to [1, 4, 5, 46].

3.1. IGA discretization

We use a uniform mesh of n_e^d elements, where n_e is the number of elements in each direction (see Figure 1 for the 2D case). The approximate solution field $u^h(\mathbf{x})$ uses B-spline basis functions such that

$$\text{2D}: \quad u^h(x,y) = \sum_{i=0}^{n_x-1}\sum_{j=0}^{n_y-1} B_{i,p_x}(x) B_{j,p_y}(y)\, U_{ij}, \tag{10a}$$

$$\text{3D}: \quad u^h(x,y,z) = \sum_{i=0}^{n_x-1}\sum_{j=0}^{n_y-1}\sum_{k=0}^{n_z-1} B_{i,p_x}(x) B_{j,p_y}(y) B_{k,p_z}(z)\, U_{ijk}, \tag{10b}$$

where $\mathbf{U}$ is the d-order tensor of control variables (i.e., degrees of freedom), p_x, p_y, and p_z denote the polynomial degree of the basis functions in x, y, and z directions, respectively, and $n_x = n_e + p_x$, $n_y = n_e + p_y$, and $n_z = n_e + p_z$ refer to the number of control variables in different directions. When using maximum-continuity IGA, knot sequences with single multiplicities at internal knots characterize the parameter space resulting in C^{p-1} basis functions. For example, in the x direction it reads

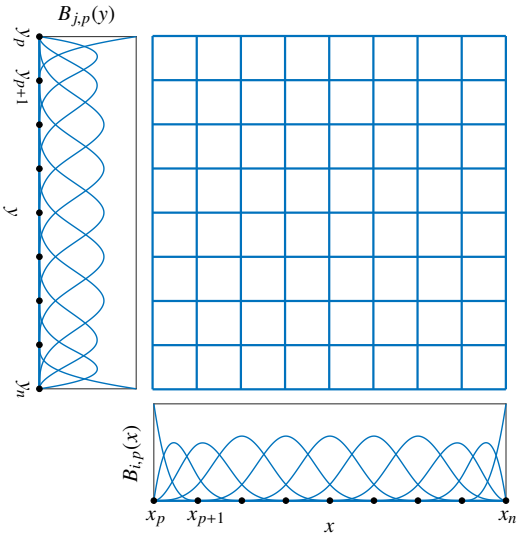
$$\Xi = [\underbrace{0, 0, ..., 0}_{p_x+1}, x_{p_x+1}, x_{p_x+2}, ..., x_{n_x-1}, \underbrace{1, 1, ..., 1}_{p_x+1}], \tag{11}$$

and the B-spline basis functions B_{i,p_x} are expressed by the Cox–De Boor recursion formula [47] as

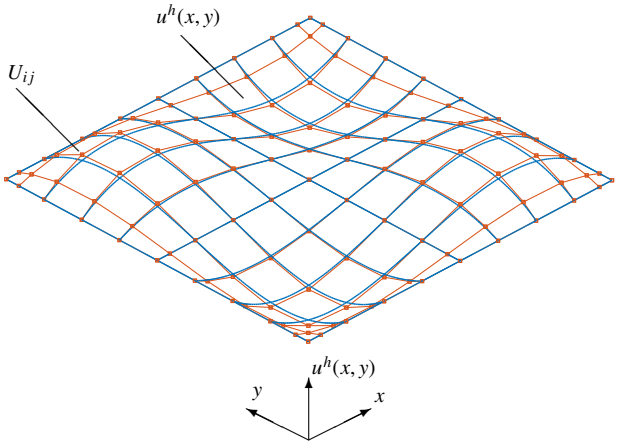
$$B_{i,0}(x) = \begin{cases} 1, & x_i \le x < x_{i+1}, \\ 0, & \text{otherwise}, \end{cases} \tag{12a}$$

$$B_{i,p_x}(x) = \frac{x - x_i}{x_{i+p_x} - x_i} B_{i,p_x-1}(x) + \frac{x_{i+p_x+1} - x}{x_{i+p_x+1} - x_{i+1}} B_{i+1,p_x-1}(x). \tag{12b}$$

When dealing with repeated knots in (11), the issue of division by zero can be fixed by setting the respective term as zero (i.e., $a/0 \equiv 0$ for any arbitrary a). Alternatively, when evaluating basis functions for a specific $x \in \Xi$, one can find the corresponding nonzero knot span in (12a) and efficiently evaluate (12b) avoiding any division by zero. More details are given in [47] (c.f. Algorithms A2.1 and A2.2 therein).



(a) Uniform bicubic 8×8 computational mesh ($p = p_x = p_y = 3$)



(b) Spatial representation of approximate solution $u^h(x, y)$ with control net variables U_{ij} (red) and solution surface (blue)

Figure 1. Tensor product basis and solution. Control variables coincide with solution at boundaries, but not necessarily at interior points

Applying the weak form (6) to the above discretization, the generalized Hermitian eigenproblem (GHEP) is

$$\mathbf{K}\mathbf{u}^h = \lambda^h \mathbf{M}\mathbf{u}^h \,, \tag{13}$$

where $\mathbf{K}$ and $\mathbf{M}$ are real symmetric (or Hermitian) stiffness and mass matrices, respectively.

Remark 1. The eigenvectors $\mathbf{u}^h$ in (13) are the vectorial form of the control variables $\mathbf{U}$. Hence, we obtain the tensor form by $\mathbf{U} = \text{Tensor}(\mathbf{u}^h)$, c.f. [48], and compute the numerical eigenfunction u^h for each eigenmode with (10).

3.1.1. Construction of multidimensional mass and stiffness matrices

We define 1D stiffness and mass matrices in x, y, and z directions as

$$K^x_{ij} := \int_x B'_{i,p_x}(x) B'_{j,p_x}(x)\, dx \,, \quad K^y_{ij} := \int_y B'_{i,p_y}(y) B'_{j,p_y}(y)\, dy \,, \quad K^z_{ij} := \int_z B'_{i,p_z}(z) B'_{j,p_z}(z)\, dz \,, \tag{14a}$$

$$M^x_{ij} := \int_x B_{i,p_x}(x) B_{j,p_x}(x)\, dx \,, \quad M^y_{ij} := \int_y B_{i,p_y}(y) B_{j,p_y}(y)\, dy \,, \quad M^z_{ij} := \int_z B_{i,p_z}(z) B_{j,p_z}(z)\, dz \,. \tag{14b}$$

We build the 2D and 3D system matrices with the following formulae (see, e.g., [48]):

$$\text{2D}: \quad \mathbf{K} = \mathbf{M}^y \otimes \mathbf{K}^x + \mathbf{K}^y \otimes \mathbf{M}^x \,, \qquad \mathbf{M} = \mathbf{M}^y \otimes \mathbf{M}^x \,, \tag{15a}$$

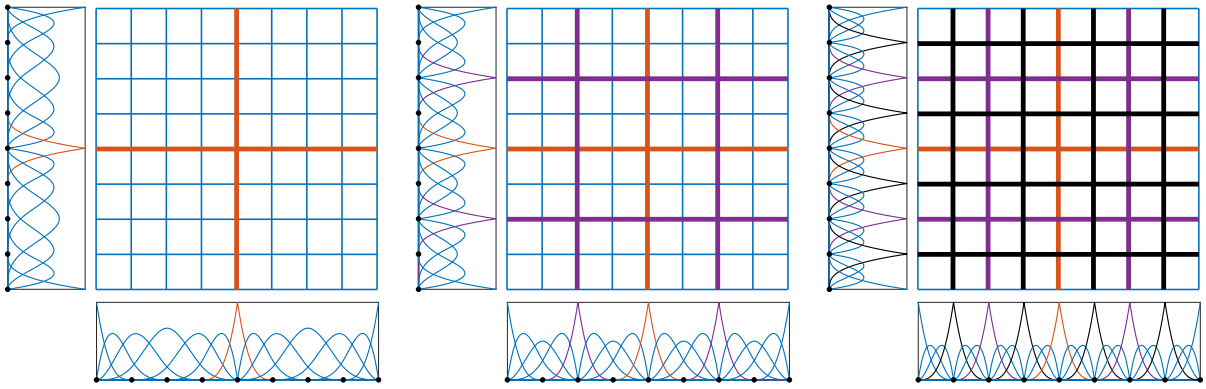
$$\text{3D}: \quad \mathbf{K} = \mathbf{M}^z \otimes \mathbf{M}^y \otimes \mathbf{K}^x + \mathbf{M}^z \otimes \mathbf{K}^y \otimes \mathbf{M}^x + \mathbf{K}^z \otimes \mathbf{M}^y \otimes \mathbf{M}^x \,, \quad \mathbf{M} = \mathbf{M}^z \otimes \mathbf{M}^y \otimes \mathbf{M}^x \,, \tag{15b}$$

where $\otimes$ indicates the Kronecker product.

3.2. Refined IGA discretizations

In refined IGA, we enrich the approximation space, thus, producing better discrete solutions, while reducing the computational cost. To achieve these goals, rIGA reduces the continuity of certain basis functions that, in turns, reduces the interconnection between degrees of freedom of the mesh [23]. By increasing the multiplicity of some existing knots up to the degree of B-spline bases in the h-refinement sense, the continuity and support of the basis functions decrease without adding new elements. The resulting zero-continuity basis functions partition the computational space into interconnected blocks separated by C^0 hyperplanes. This connectivity reduction reduces the solver cost (i.e., lower matrix factorization and forward/backward elimination costs, c.f. [23, 26]). The knot insertion adds new control variables and, therefore, enlarges the Galerkin space, modifying the spectral approximation properties

of the IGA approach. Figure 2 describes three symmetric partitions with respective blocksizes of 4, 2, and 1 for the bicubic 8×8 mesh of Figure 1 (blocksize is the number of elements of C^0 blocks in each direction).



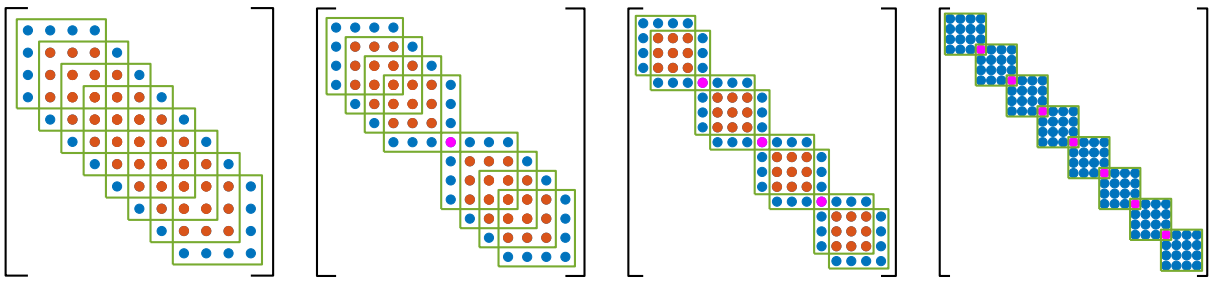
(a) $\ell = 1$ (blocksize = 4) (b) $\ell = 2$ (blocksize = 2) (c) $\ell = 3$ (blocksize = 1)

Figure 2. Different partition levels ($\ell = 1, 2, 3$) and C^0 separators (in red, purple, black) for the bicubic 8×8 grid of Figure 1.

Remark 2. For simplicity, we assume the mesh size in each direction is a power of two (i.e., $n_e = 2^s$). This allows us to split the mesh symmetrically to obtain 2^ℓ blocks (i.e., macroelements) in each direction with blocksize $2^{s-\ell}$, where $\ell = 0, 1, ..., s$ is the partitioning level. Here, $\ell = 0$ refers to the maximum-continuity IGA with C^{p-1} continuity everywhere, while $\ell = s$ is equivalent to FEA with C^0 continuity at all element interfaces (knot lines). We consider the same polynomial degree p in all directions.

Remark 3. Knot insertion does not modify the domain either geometrically or parametrically [47]. This is because the mapping function remains unchanged under h-refinement [46]. Thus, rIGA discretizations do not affect the geometry representation errors of the original IGA discretization.

Figure 3 depicts the matrix patterns of a 1D domain under different discretizations with $n_e = 8$ and $p = 3$. The figure shows the strong interconnectivity between degrees of freedom for maximum-continuity IGA and how rIGA partitions weaken this connectivity, which accelerates the solution and reduces the memory required to solve. For the maximum partitioning level ($\ell = 3$), each macroelement consists of one element, and the interconnection reduces to a single degree of freedom.



(a) C^{p-1} IGA (b) rIGA with $\ell = 1$ (c) rIGA with $\ell = 2$ (d) rIGA with $\ell = 3$ (FEA)

Figure 3. Matrix patterns of a cubic eight-element domain in 1D under different discretizations (green squares delimit elemental matrices). (a) maximum-continuity C^{p-1} IGA, (b) rIGA with blocksize = 4, (c) rIGA with blocksize = 2, and (d) rIGA with blocksize = 1, which is equal to the cubic FEA discretization. Uniform high-continuity implies strong interconnection between degrees of freedom (red dots) in (a). The interconnection weakens for the interior elements of each macroelement under rIGA discretizations in (b) and (c). Magenta dots denote the C^0 interconnection between elements.

Increasing the multiplicity of an existing knot up to p (i.e., the degree of bases) adds $p - 1$ control variables to each direction. Thus, the ℓ-th partitioning level ($\ell > 0$) adds $[2^{\ell-1}(p-1)]^d$ control variables. Consequently, the total

number of degrees of freedom to discretize (1) is

$$\text{IGA}: \quad N = (n_e + p - 2)^d \,, \tag{16a}$$

$$\text{rIGA}: \quad N = \left[n_e + 2^\ell (p-1) - 1\right]^d \,. \tag{16b}$$

4. Solving generalized Hermitian eigenproblems

Eigencomputation consists of several numerical algorithms. Based on the problem type and the structure of matrices, different eigensolver packages have their own methodologies for the eigensolution (see, e.g., [49–51]). When solving a GHEP, the Lanczos eigensolver performs the main numerical part of the solution, projecting onto the Krylov subspace. To obtain eigenpairs within the desired portion of the spectrum, Lanczos algorithm is executed along with a sequence of numerical procedures, namely: shift-and-invert spectral transformation, spectrum slicing, restarting, and deflation. Herein, we present a short review of these numerical algorithms that constitute the building blocks of our eigensolver.

4.1. Shift-and-invert spectral transformation

The eigenproblem (13) defines a large sparse eigensystem with eigenvalues of arbitrary multiplicity. Numerically, we seek to compute the eigenpairs λ_i^h and $\mathbf{u}_i^h$ within the given interval $\lambda_i^h \in [\lambda_s, \lambda_e]$, where the magnitude of either λ_s or λ_e can be infinite. The conversion of a generalized eigenproblem to a standard one is fraught. The transformation can factor $\mathbf{M}$ (or $\mathbf{K}$) into its Cholesky decomposition as $\mathbf{M} = \mathbf{L}\mathbf{L}^T$ (or $\mathbf{K} = \mathbf{L}\mathbf{L}^T$) and solve $\mathbf{L}^{-1}\mathbf{K}\mathbf{L}^{-T}\mathbf{w} = \lambda^h\mathbf{w}$ (or $\mathbf{L}^{-1}\mathbf{M}\mathbf{L}^{-T}\mathbf{w} = \mathbf{w}/\lambda^h$), where $\mathbf{u}^h = \mathbf{L}^{-T}\mathbf{w}$ (superscript T refers to the transpose or conjugate transpose of a real symmetric or a Hermitian matrix, respectively). Either transformation may numerically fail. This fragility may have many causes, such as $\mathbf{K}$ being semidefinite, the eigenvalues being clustered, or $\mathbf{L}$ being poorly conditioned. Any of these characteristics affect the extraction of eigenvectors from $\mathbf{u}^h = \mathbf{L}^{-T}\mathbf{w}$ in the backward elimination [39]. Thus, eigensolvers generally perform a spectral transformation (ST) and solve the following shifted problem

$$(\mathbf{K} - \sigma\mathbf{M})\mathbf{u}^h = (\lambda^h - \sigma)\mathbf{M}\mathbf{u}^h \,, \tag{17}$$

to obtain an accurate approximation of eigenpairs (see, e.g., [39, 40, 42, 43]). Then, most eigensolvers use the shift-and-invert eigenproblem by solving the following system:

$$(\mathbf{K} - \sigma\mathbf{M})^{-1}\mathbf{M}\mathbf{u}^h = \theta\mathbf{u}^h \,, \tag{18}$$

where $\theta = 1/(\lambda^h - \sigma)$. The resulting operator matrix $(\mathbf{K} - \sigma\mathbf{M})^{-1}\mathbf{M}$ is not symmetric, but it is self-adjoint with respect to $\mathbf{M}$. Since matrices $\mathbf{K}$ and $\mathbf{M}$ have different null spaces, the ST matrix $\mathbf{K} - \sigma\mathbf{M}$ is non-singular unless σ is an eigenvalue. Hence, the strategy (18) addresses the (potential) issue of dealing with semidefinite system matrices. Another advantage of shift-and-invert problem is that it transforms the eigenvalues λ_i^h nearest the shift σ into well-separated eigenvalues θ_i of the reciprocal eigenproblem of (18) (see, Figure 4). A well-selected shift enables the eigensolver to compute many eigenpairs in a single iteration.

4.2. Spectrum slicing

When solving a GHEP, in practical cases, the requested eigenvalue interval is large. An example is to find all critical speeds of a turbine shaft in a given working interval. Thus, to prevent deterioration of the convergence rate of the eigensolution for eigenvalues far from σ in (18), we select additional shifts σ_ks using a spectrum slicing technique (see, e.g., [44]). When shifting, we employ an LDL Cholesky factorization of the ST matrix, that is, $(\mathbf{K} - \sigma_k\mathbf{M}) = \mathbf{L}\mathbf{D}\mathbf{L}^T$, where $\mathbf{D}$ is a diagonal matrix. Based on the Sylvester's law of inertia [52], the number of eigenvalues smaller than σ_k is equal to the number of negative eigenvalues of $\mathbf{D}$. Therefore, by defining the inertia $\nu_k := \nu(\mathbf{K} - \sigma_k\mathbf{M})$ as the number of eigenvalues smaller than σ_k, for two consecutive shifts σ_k and σ_{k+1}, the interval $[\sigma_k, \sigma_{k+1}]$ has $\nu_{k+1} - \nu_k$ eigenvalues including their multiplicities. This rule drives the spectrum slicing when determining the required number of shifts, which helps finding the requested eigenpairs with the true multiplicities while minimizing the computational effort.

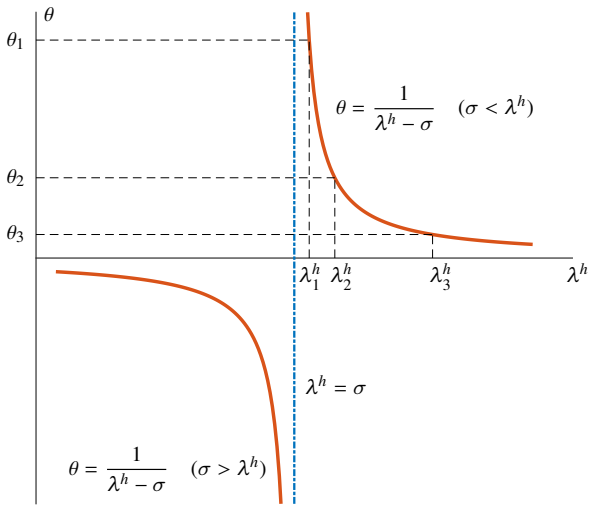


Figure 4. The shift-and-invert spectral transformation: λ^h values near the shift σ are well separated on the θ axis.

4.3. Lanczos method formulation for the generalized Hermitian eigenproblem

Herein, we implement the Lanczos method to solve the generalized Hermitian eigenproblems. For each individual shift σ_k, given the factorization $\mathbf{K} - \sigma_k\mathbf{M} = \mathbf{L}\mathbf{D}\mathbf{L}^T$, we define the operator matrix $\mathbf{H} := (\mathbf{K} - \sigma_k\mathbf{M})^{-1}\mathbf{M}$, so that the eigenproblem (18) becomes $\mathbf{H}\mathbf{u}^h = \theta\mathbf{u}^h$. The m-step Lanczos decomposition consists of reducing the $N \times N$ matrix $\mathbf{H}$ to a symmetric tridiagonal matrix $\mathbf{T}_m$ ($m \ll N$) as follows (see, e.g., [40, 53]),

$$\mathbf{H}\mathbf{V}_m = \mathbf{V}_m\mathbf{T}_m + \beta_{m+1}\mathbf{v}_{m+1}\mathbf{e}_m^T, \tag{19}$$

where $\mathbf{e}_m$ is the m-th coordinate vector, and the term $\beta_{m+1}\mathbf{v}_{m+1}\mathbf{e}_m^T$ is the residual of the m-step Lanczos decomposition. In the above equation,

$$\mathbf{T}_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{bmatrix}, \tag{20}$$

and $\mathbf{V}_m := [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_m]$ is the matrix of Lanczos vectors. Assuming $\beta_1 = 0$ and being $\mathbf{v}_1$ an initial normalized vector of length N, i.e., $\|\mathbf{v}_1\|_\mathbf{M} = (\mathbf{v}_1^T\mathbf{M}\mathbf{v}_1)^{1/2} = 1$, the components of $\mathbf{T}_m$ and vectors $\mathbf{v}_{j+1}$ ($j = 1, 2, ..., m$) are obtained by the following recurrence formulae,

$$\alpha_j = \mathbf{v}_j^T\mathbf{M}\mathbf{H}\mathbf{v}_j\,, \tag{21a}$$

$$\beta_{j+1} = \left\|\mathbf{H}\mathbf{v}_j - \alpha_j\mathbf{v}_j - \beta_j\mathbf{v}_{j-1}\right\|_\mathbf{M}\,, \tag{21b}$$

$$\mathbf{v}_{j+1} = \frac{1}{\beta_{j+1}}\left(\mathbf{H}\mathbf{v}_j - \alpha_j\mathbf{v}_j - \beta_j\mathbf{v}_{j-1}\right)\,. \tag{21c}$$

In this way, the Lanczos vector $\mathbf{v}_{m+1}$ is $\mathbf{M}$-orthogonal with respect to the columns of $\mathbf{V}_m$ in the Gram–Schmidt sense, resulting in $\mathbf{V}_m^T\mathbf{M}\mathbf{v}_{m+1} = 0$. Hence, the $\mathbf{M}$-inner product of $\mathbf{V}_m$ premultiplied in (19) leads to the following equation, noting that $\mathbf{V}_m$ is an $\mathbf{M}$-orthogonal matrix (i.e., $\mathbf{V}_m^T\mathbf{M}\mathbf{V}_m = \mathbf{I}$),

$$\mathbf{V}_m^T\mathbf{M}\mathbf{H}\mathbf{V}_m = \mathbf{T}_m\,. \tag{22}$$

The above equation reveals that $\mathbf{T}_m$ is the $\mathbf{M}$-orthogonal projection of $\mathbf{H}$ onto the m-th order Krylov subspace $\mathcal{K}_m(\mathbf{H}, \mathbf{v}_1)$. Therefore, if ω_j and $\mathbf{w}_j$ are the eigenpairs of $\mathbf{T}_m$ (a.k.a., Ritz values and Ritz vectors), the Rayleigh–Ritz

approximation of the eigenpairs of $\mathbf{H}$ is

$$\theta_j = \omega_j \,, \tag{23a}$$

$$\mathbf{u}_j^h = \mathbf{V}_m \mathbf{w}_j \,. \tag{23b}$$

The eigenvalues of the original GHEP (13) for each Lanczos run is then obtained as $\lambda_j^h = \sigma_k + 1/\theta_j$ for each shift σ_k. Since $\mathbf{T}_m$ is a symmetric tridiagonal matrix, there exist multiple methods for computing its eigenpairs (see, e.g., [54]).

4.4. Restarting

Krylov eigensolvers incorporate effective restarting mechanisms. The main reason for restart is that the approximation quality of the Ritz pairs deteriorates after a few steps of the Lanczos run, forcing the eigensolver to start a new recurrence. Well-known restarting techniques are the Sorensen's implicit restart [55], employed in the context of Lanczos methods, the thick-restart Lanczos method of Wu and Simon [56], and its unsymmetrical equivalent for the Arnoldi case referred to as the Krylov–Schur method of Stewart [57, 58].

Herein, we employ the thick-restart approach as an effective restarting technique in the case of Hermitian eigenproblems. For each shift σ_k, we consider a set of iterations. Each iteration itself involves an m-step Lanczos process. The first iteration starts with an initial basis vector $\mathbf{v}_1$, while the next ones benefit from the previously obtained spectral approximation. The quality of the eigenpair approximation in each iteration can be assessed by means of the residual norm of the Ritz pairs as follows [40]:

$$\mathrm{res}_j := \frac{1}{\theta_j} \beta_{m+1} \left| \mathbf{e}_m^T \mathbf{w}_j \right| , \qquad j = 1, 2, ..., m \,. \tag{24}$$

When restarting, the eigensolver keeps an appropriate number of Lanczos vectors, let say $c < m$, and the Lanczos recurrence (21) restarts with the following initial values:

$$\mathbf{r}_{c+1} = \mathbf{H}\mathbf{v}_c - \alpha_c \mathbf{v}_c - \sum_{i=1}^{c} \beta_i \mathbf{v}_{i-1} \,, \tag{25a}$$

$$\beta_{c+1} = \|\mathbf{r}_{c+1}\|_{\mathbf{M}} \,, \tag{25b}$$

$$\mathbf{v}_{c+1} = \frac{\mathbf{r}_{c+1}}{\beta_{c+1}} \,, \tag{25c}$$

where we orthogonalize with respect to all stored Lanczos vectors. This process continues until we compute all eigenpairs corresponding to the current shift.

4.5. Deflation, spectrum recycling, and backtracking

Spectrum slicing is typically followed by a deflation. This algorithm prevents the reconvergence of a cluster of eigenvalues obtained from previous shifts and maintains the orthogonality of their corresponding eigenvectors. Additionally, there are two more numerical algorithms that may be occasionally employed when needed: spectrum recycling and backtracking. The former transforms some Lanczos basis from a previous shift to the current one in case they create the same Krylov subspace. The latter considers a shift backward if the inertia information reveals that the Lanczos process missed some eigenvalues in the interval $[\sigma_k, \sigma_{k+1}]$. Further mathematical details about these algorithms can be found in, e.g., [40, 44, 56, 59].

4.6. Summary of numerical eigenanalysis procedures

To summarize, we herein present a pseudo-code describing the numerical eigenanalysis procedure for solving a generalized Hermitian eigenproblem (see Algorithm 1). However, the entire process might be more complicated. For instance, the number of shifts, the shift distribution within the desired spectrum interval, the number of requested eigenpairs per shift, the size of Krylov subspace, and the adequate number of Lanczos bases to keep before restart play crucial roles in the efficiency of the eigensolution. These variables mainly depend on the size of the eigensystem, the total number of requested eigenpairs, and the selected eigensolver package.

Algorithm 1. Eigensolution procedure for GHEP.

Input: System matrices $\mathbf{K}$ and $\mathbf{M}$, spectrum interval $[\lambda_s, \lambda_e]$, requested number of eigenpairs N_0, initial vector $\mathbf{v}_{\text{in}}$, and residual tolerance ε.

Output: N_0 eigenpairs λ^h and $\mathbf{u}^h$ within the interval $[\lambda_s, \lambda_e]$

1: Check inertia information to determine the number of eigenpairs correspond to the given interval
2: Set the numbers of shifts $\mathcal{N}_{\text{shift}}$ and to-be-computed eigenvalues per shift N_{ev} based on N_0 and inertia information
3: $\mathcal{N}_{\text{iter}} = 0$, /* a counter for total number of iterations */
4: **for all** $k = 1, 2, ..., \mathcal{N}_{\text{shift}}$, **do**
5: $\mathbf{K} - \sigma_k \mathbf{M} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ /* matrix factorization; for the first shift $\sigma_1 = \lambda_s$, we have Cholesky factors from inertia check */
6: Set the size of Krylov subspace $m \geq N_{ev}$
7: Normalize: $\mathbf{v}_1 \leftarrow \mathbf{v}_{\text{in}} / \|\mathbf{v}_{\text{in}}\|_{\mathbf{M}}$, $\mathbf{V}_1 = \mathbf{v}_1$
8: $\beta_1 = 0$, Res $= 10^6$ /* setting a big value for residual norm to get into iterations */
9: **while** Res $> \varepsilon$, **do**
10: $\mathcal{N}_{\text{iter}} \leftarrow \mathcal{N}_{\text{iter}} + 1$
11: **for all** $j = 1, 2, ..., m$, **do** /* m-step Lanczos recurrence */
12: $\mathbf{t} = \mathbf{H}\mathbf{v}_j = (\mathbf{K} - \sigma_k \mathbf{M})^{-1}\mathbf{M}\mathbf{v}_j$
13: $\alpha_j = \mathbf{v}_j^T \mathbf{M} \mathbf{t}$
14: $\mathbf{r} = \mathbf{t} - \alpha_j \mathbf{v}_j - \beta_j \mathbf{v}_{j-1}$
15: $\beta_{j+1} = \|\mathbf{r}\|_{\mathbf{M}}$
16: $\mathbf{v}_{j+1} = \mathbf{r}/\beta_{j+1}$, $\mathbf{V}_{j+1} \leftarrow \begin{bmatrix} \mathbf{V}_j & \mathbf{v}_{j+1} \end{bmatrix}$
17: **end for**
18: Construct symmetric tridiagonal matrix $\mathbf{T}_m$
19: Compute Ritz values $\theta_j = \omega_j$ and Ritz vectors $\mathbf{w}_j$ $(j = 1, 2, ..., m)$
20: Compute numerical eigenpairs λ_j^h and $\mathbf{u}_j^h$ $(j = 1, 2, ..., m)$
21: **for all** $j = 1, 2, ..., m$, **do**
22: $\text{res}_j = \frac{1}{\theta_j} \beta_{m+1} \left| \mathbf{e}_m^T \mathbf{w}_j \right|$ /* residual norm of each Ritz pair */
23: **end for**
24: Res $= \max(\text{res}_j)$
25: Keep adequate numbers of Lanczos bases and compute initial values for restart
26: **end while**
27: Execute deflation and, if applicable, spectrum recycling and backtracking
28: **end for**

5. Cost estimation of the eigensolution

Estimating the cost of an eigensolution is challenging because it contains several numerical algorithms. We focus on Lanczos decomposition, which is called several times during the spectrum slicing. We estimate the computational cost of the eigensolution based on the most expensive operations.

5.1. Most expensive numerical operations

For large eigensystems, we identify the three most expensive operations: 1) Cholesky factorization of the ST matrix, 2) forward/backward eliminations of Cholesky factors, and 3) multiplications of the mass matrix $\mathbf{M}$ by vectors. Other numerical procedures in the eigenanalysis, e.g., vector products and system integration, are of lower orders compared to the most expensive operations, and we exclude them from our cost estimates.

5.1.1. Matrix factorization

For each shift, we perform one Cholesky factorization (line 5 of Algorithm 1). To determine the factorization cost, we follow the theoretical estimates of 2D and 3D systems in terms of floating-point operations (FLOPs) [20, 23]. We have

$$\text{IGA}: \quad \text{FLOPs}_{\text{fact}} = \mathcal{O}\left(N^{(d+1)/2} p^3\right), \tag{26a}$$

$$\text{rIGA}: \quad \text{FLOPs}_{\text{fact}} = 2^{d\ell}\, \mathcal{O}\left((2^{-d\ell} N)^{(d+1)/2} p^3\right) + \mathcal{O}\left(N^{(d+1)/2}\right). \tag{26b}$$

Garcia et al. [23] show the factorization cost in large systems reduces by up to $\mathcal{O}(p^2)$ when using rIGA and that the optimal blocksize is of 2^4 elements. Matrix factorization becomes expensive in multidimensional problems ($d \geq 2$)

especially when n_e is large. For 1D systems, factorization cost grows linearly with n_e, thus, it is rather inexpensive even for large systems. For such problems, rIGA does not bring in any remarkable computational savings.

5.1.2. *Forward/backward elimination*

Projecting onto the Krylov subspace at each iteration of the eigensolution involves m forward/backward eliminations of the Cholesky factors of the ST matrix (line 12 of Algorithm 1). In the following, we refer to such operation as "f/b elimination" for the sake of brevity. The cost of f/b elimination is proportional to the number of nonzero entries of the Cholesky factors. This cost, for 2D and 3D systems, in the form theoretical memory estimates is [20, 21, 23]:

$$\text{IGA}: \quad \text{FLOPs}_{\text{f/b}} = \mathcal{O}\left(N^{(d+1)/3}p^2\right), \tag{27a}$$

$$\text{rIGA}: \quad \text{FLOPs}_{\text{f/b}} = 2^{d\ell}\mathcal{O}\left((2^{-d\ell}N)^{(d+1)/3}p^2\right) + \mathcal{O}\left(N^{(d+1)/3}\right). \tag{27b}$$

When using optimal rIGA discretization with blocksize of 2^4 elements, the f/b elimination cost in large systems reduces asymptotically by up to $\mathcal{O}(p)$ with respect to the IGA cost [23].

5.1.3. *Matrix–vector multiplication*

Each m-step Lanczos recurrence involves m computations of α_j and β_j. Computing α_j requires two multiplications of the mass matrix $\mathbf{M}$ by the respective vectors (lines 12 and 13 of Algorithm 1). Referring to matrix–vector multiplication as "mat–vec", the computation of β_j only needs one mat–vec product for the $\mathbf{M}$-norm calculation (line 15). Thus, the eigensolver calls this operation approximately $3m\mathcal{N}_{\text{iter}}$ times during the eigencomputation (other numerical procedures like restarting and deflation also incorporate this operation, but with significantly lower callings compared to the Lanczos recurrence).

The cost of mat–vec is proportional to the number of nonzeros of the mass matrix, $N_{nz}(\mathbf{M})$. In particular, the number of nonzeros of either mass or stiffness matrix is related to the sum of interactions that each basis function has with all other bases [21]. As a result, referring to the matrix layouts of 1D systems in Figure 3 and the tensor-product property of Section 3.1.1, one obtains the number of nonzeros of $\mathbf{M}$ as

$$\text{IGA}: \quad N_{nz}(\mathbf{M}) = \left[n_e(2p+1) + p^2\right]^d, \tag{28a}$$

$$\text{rIGA}: \quad N_{nz}(\mathbf{M}) = 2^{d\ell}\left[2^{-\ell}n_e(2p+1) + p^2 - 1\right]^d. \tag{28b}$$

Therefore, the cost of mat–vecs with IGA discretization is close to $\mathcal{O}(Np^d)$ when N is sufficiently large, while that of rIGA discretizations is slightly higher. Considering the optimal blocksize of 2^4, the ratio $N_{nz}(\mathbf{M}_{\text{rIGA}})/N_{nz}(\mathbf{M}_{\text{IGA}})$ is equal to $(1.03)^d$ and $(1.14)^d$, for $p = 2$ and 5, respectively, when n_e is in the range of $2^6 \sim 2^{11}$. The degradation incurred by rIGA, however, is not comparable to the improvements we obtain in the factorization and f/b elimination steps by using rIGA.

5.2. *Total eigencomputation cost*

The total cost of the eigensolution is governed by the number of factorizations, $\mathcal{N}_{\text{fact}}$, the number of f/b eliminations, $\mathcal{N}_{\text{f/b}}$, and the number of mat–vecs, $\mathcal{N}_{\text{m–v}}$. Table 1 expresses these numbers in terms of number of shifts, $\mathcal{N}_{\text{shift}}$, and the total number of iterations, $\mathcal{N}_{\text{iter}}$, carried out by the eigensolver. The table also compares how an rIGA discretization improves or degrades the performance of each operation to that of an IGA discretization. To build this table, we assume the following:

- IGA and rIGA discretizations use the same number of shifts. This is derived from inertia information described in Section 4.2 and confirmed by numerical results of Section 7.
- IGA and rIGA discretizations require the same number of iterations. We show this numerically in Section 7 for a sufficiently large number of eigenpairs ($N_0 \geq 2^{10}$).
- The number of Lanczos steps m has the same average per shift through all iterations under both IGA and rIGA discretizations. Numerical results of Section 7 confirm this assumption.
- The number of degrees of freedom is sufficiently large, so the cost improvements due to the use of rIGA described in [23] hold.

Table 1. Most expensive computational operations of the Lanczos eigensolution algorithm and the effect of rIGA versus IGA discretizations. We express the number of times we call each operation in terms of number of shifts, $\mathcal{N}_{\text{shift}}$, and the total number of iterations, $\mathcal{N}_{\text{iter}}$.

Numerical operation	Matrix factorization	F/b elimination	Matrix–vector product
Number of times the operation is called	$\mathcal{N}_{\text{fact}} = \mathcal{N}_{\text{shift}}$	$\mathcal{N}_{\text{f/b}} \approx m\mathcal{N}_{\text{iter}}$	$\mathcal{N}_{\text{m–v}} \approx 3m\mathcal{N}_{\text{iter}}$
Improvement/degradation of performing one operation in rIGA	Improving by $\mathcal{O}(p^2)$	Improving by $\mathcal{O}(p)$	Degrading by $N_{nz}(\mathbf{M}_{\text{rIGA}})/N_{nz}(\mathbf{M}_{\text{IGA}})$

5.3. Theoretical time estimates

Equations (26)–(28) express the number of FLOPs of algorithms described in Table 1 as $\mathcal{O}(N^a p^b)$, where factors a and b vary for different operations and space dimensions as displayed in Table 2. Herein, we are interested in measuring the computational time. Since time and FLOPs are correlated for the type of operations considered here (as already shown in, e.g., [23]), we estimate the time T_{op} required to perform each operation as

$$T_{\text{op}} \approx AN^a p^b. \tag{29}$$

Thus, we estimate the total time required for eigencomputation as

$$T \approx \mathcal{N}_{\text{fact}} T_{\text{fact}} + \mathcal{N}_{\text{f/b}} T_{\text{f/b}} + \mathcal{N}_{\text{m–v}} T_{\text{m–v}}\,. \tag{30}$$

Table 2. Constants a and b in (29) for the theoretical estimation of computational time of the most expensive numerical operations of the Lanczos eigensolution algorithm. Constant a is equal for both IGA and rIGA discretizations.

Constants	Discretization method	Matrix factorization	F/b elimination	Mat–vec product *
a	IGA and rIGA	$(d+1)/2$	$(d+1)/3$	1
b	IGA	3	2	d
	rIGA	1	1	d

* The time of mat–vecs under rIGA increases by a factor in the range of $(1.03)^d$ and $(1.14)^d$ with $p = 2 \sim 5$.

Remark 4. When seeking for a sufficiently large number of eigenpairs, N_0, we assume the computational time grows linearly with respect to N_0. Numerical results of Section 7 confirm this assumption.

6. Implementation details

We discretize the model problem using PetIGA [4], a high-performance isogeometric analysis solver built on PETSc (portable extensible toolkit for scientific computation) [60]. PetIGA has been utilized in many scientific and engineering applications (see, e.g., [10, 11, 16, 20, 21, 23, 25, 26, 61–64]). It allows us to investigate both IGA and rIGA discretizations on test cases with different numbers of elements in 2D and 3D, different polynomial degrees of the B-spline spaces, and different partitioning levels of the mesh.

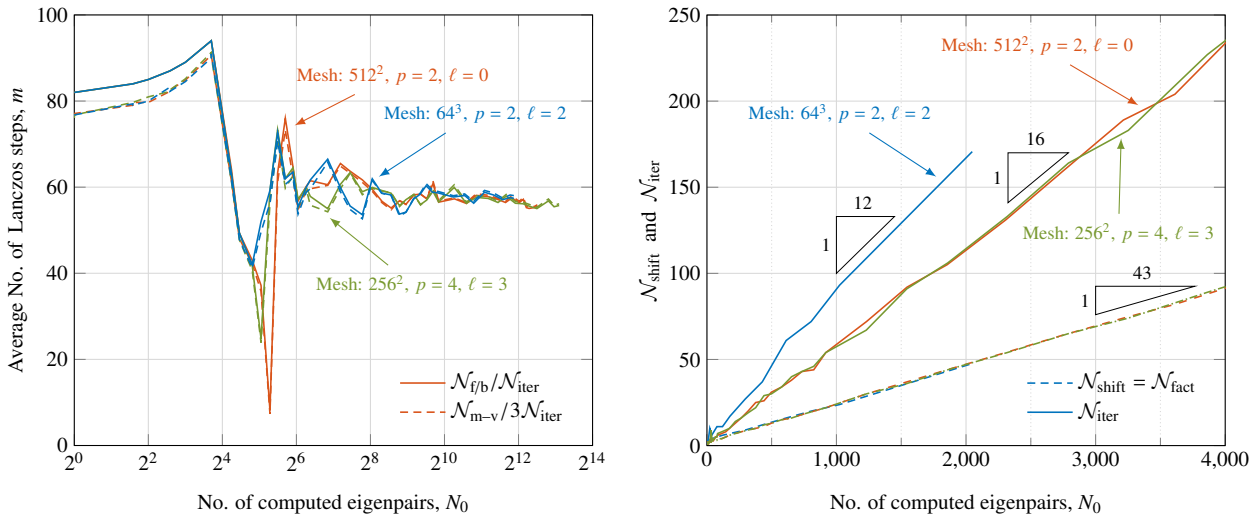
We also use SLEPc, the scalable library for eigenvalue problem computations [49], for performing the eigenanalysis, allowing us to apply the shift-and-invert spectral transformation. SLEPc, used to solve different eigenproblems (see, e.g., [44, 65–69]), employs the Krylov–Schur algorithm by default, which is equal to the thick-restart Lanczos algorithm in the case of generalized Hermitian eigenproblems. SLEPc calculates almost the same number of eigenpairs for each shift, allowing us to estimate the computational costs efficiently.

We use multifrontal direct solver MUMPS [70] to construct the LDL Cholesky factors, compute the required inertia for shift selections, and perform the forward/backward eliminations. We employ the sequential version of MUMPS, which runs on a single thread (core). We also use the automatic orderings provided by METIS [71]. For each test case, since all ST matrices have the same sparsity pattern, we only perform one *symbolic* factorization, followed by a certain number of numerical factorizations depending on the number of required shifts. We executed all tests on a workstation equipped with an Intel Xeon Gold 6230 CPU at 2.10 GHz with 256 GB of RAM.

7. Numerical results

We report the computational times (in seconds) required for finding the eigenpairs of the Laplace operator described in Section 2. We test different mesh sizes with $n_e = 2^s$ elements in each direction, different partitioning levels of the rIGA discretization, namely $\ell = 0$ (IGA), $1, 2, ..., s$ (FEA), and different polynomial degrees p of B-spline bases. For 2D problems, we consider uniform meshes with $s = 8, 9, 10, 11$ and degree of $p = 2, 3, 4, 5$. For the 3D case, we test on $s = 5, 6, 7$ and the same degrees as in 2D. To investigate the computational savings of the rIGA discretization compared to its IGA counterpart, we find the first N_0 eigenpairs of the PDE system with different partitioning levels ℓ, where N_0 can be as large as the number of eigenmodes of the IGA discretization (N_{IGA}). We report the elapsed time for finding all N_0 eigenpairs, T_{N_0}, the average required time per eigenmode, $T_{\text{av}} := T_{N_0}/N_0$, and a normalization of time given by $\widetilde{T} := T_{N_0}/N_0 N$.

Before proceeding with time performance of IGA and rIGA discretizations in eigenanalysis, we show in Figure 5a that the number of Lanczos steps m for a sufficiently large N_0 becomes constant, independently of the mesh size and dimension. This constancy confirms the assumptions of Section 5. Furthermore, Figure 5b demonstrates that the number of shifts and the number of iterations increase linearly with N_0, indicating the proportional relationship of T_{N_0} and T_{av} (see Remark 4). The figure also shows that 3D systems need more iterations than 2D systems for finding the same number of eigenpairs. These observations allow us to predict the expected times for large systems by solving just a small portion of the spectrum.



(a) Number of Lanczos steps

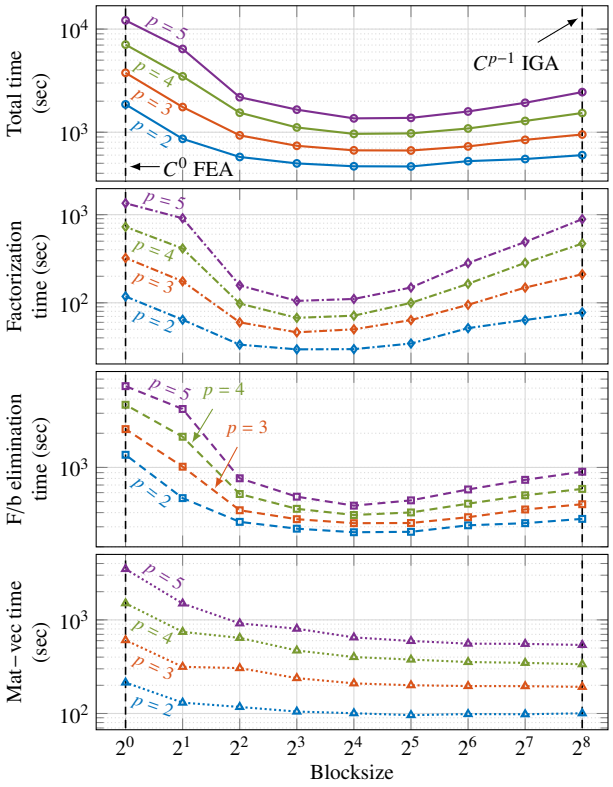
(b) Number of shifts and iterations

Figure 5. Dependence of Lanczos steps (i.e., $m \approx \mathcal{N}_{\text{f/b}}/\mathcal{N}_{\text{iter}} \approx \mathcal{N}_{\text{m–v}}/3\mathcal{N}_{\text{iter}}$), shifts, and iterations as the number of eigenpairs computed (N_0) grows. Three different systems: variation in space dimension, domain size, polynomial degree, and partitioning level.

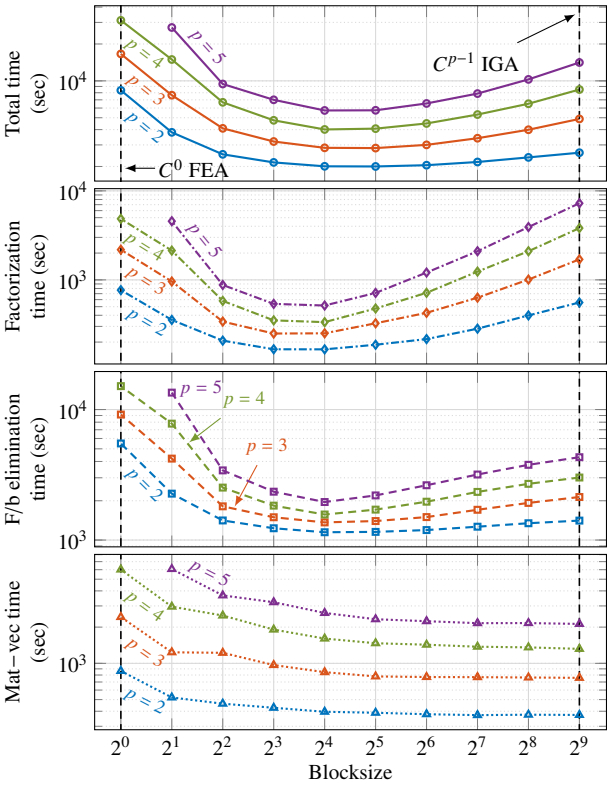
7.1. Time performance of 2D eigenproblems

The partitioning level ℓ of rIGA affects the computational time of the eigenanalysis. To see this, we focus on the most expensive numerical operations described in Section 5.1. We monitor the total elapsed time for different blocksizes considering the partitioning scheme presented in Section 3.2. There is an optimal blocksize of $2^{s-\ell}$ at which the factorization and f/b elimination times are minimum; however, the time of mat–vec products always increases with ℓ. Figure 6 shows the computational times of finding $N_0 = 2^{12}$ eigenpairs (i.e., T_{N_0}) as a function of the blocksize. The rIGA factorization time reaches a minimum at a blocksize of 16 elements almost in all cases, which coincides with the optimal blocksize for the f/b elimination time. Hence, we obtain the maximum savings for the total elapsed time (considering all numerical operations) by employing macroelements of size 16.

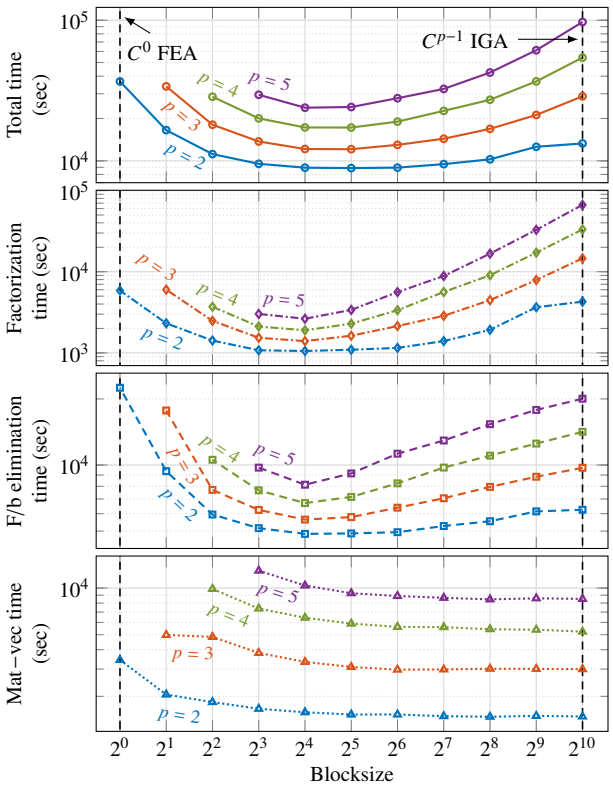
For a sufficiently large matrix, its Cholesky factorization is more expensive than f/b elimination and mat–vec multiplication. However, Krylov eigensolvers perform multiple f/b eliminations and mat–vecs per Cholesky factorization. In our 2D case, $\mathcal{N}_{\text{f/b}} \approx 150\,\mathcal{N}_{\text{fact}}$ and $\mathcal{N}_{\text{m–v}} \approx 450\,\mathcal{N}_{\text{fact}}$ (see Table 1 and Figure 5). This combination makes



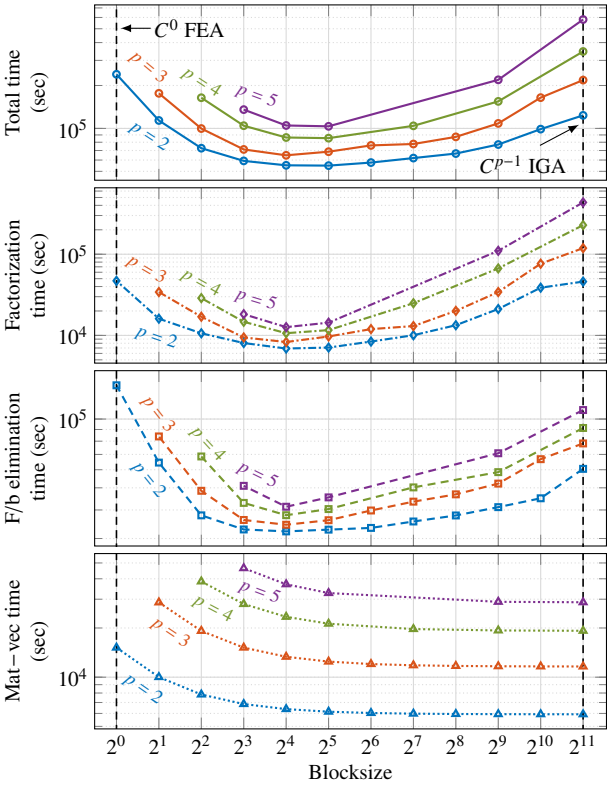
(a) Mesh: 256^2



(b) Mesh: 512^2



(c) Mesh: 1024^2



(d) Mesh: 2048^2

Figure 6. Total times and those of the most expensive numerical operations for finding first $N_0 = 4096$ eigenpairs of the 2D Laplace operator with different mesh sizes $n_e = 2^s$ ($s = 8, 9, 10, 11$) and degrees $p = 2, 3, 4, 5$. We test rIGA discretizations with different blocksizes $2^{s-\ell}$ obtained using different levels of partitioning $\ell = 0, 1, ..., s$.

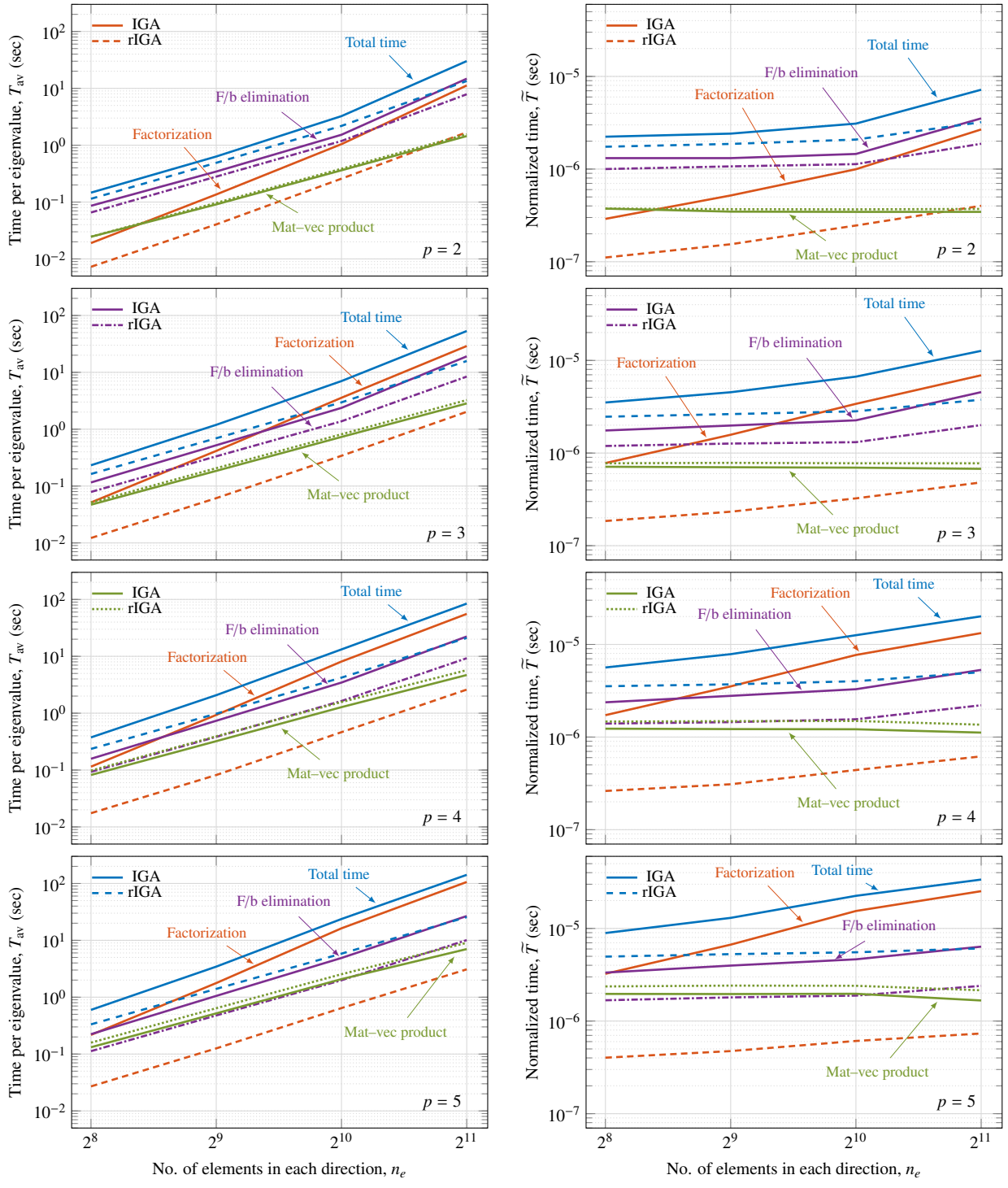
Table 3. Average computational times per eigenvalue, $T_{\text{av}} = T_{N_0}/N_0$, and the number of executions of the most expensive operations for finding $N_0 = 4096$ eigenpairs of the 2D test cases illustrated in Figure 6. For rIGA discretizations, we select a blocksize of 16 elements.

Discretization			Factorization			F/b elimination			Mat–vec product			Total	
Mesh	p	Method	$\mathcal{N}_{\text{fact}}$	$T_{\text{av,fact}}$ (sec)	Improved by	$\mathcal{N}_{\text{f/b}}$	$T_{\text{av,f/b}}$ (sec)	Improved by	$\mathcal{N}_{\text{m–v}}$	$T_{\text{av,m–v}}$ (sec)	Degraded by	T_{av} (sec)	Improved by
256^2	2	IGA	96	0.019	**2.612**	13672	0.086	**1.311**	41143	0.024	0.998	0.146	**1.282**
		rIGA	96	0.007		13538	0.065		40705	0.024		0.114	
	3	IGA	96	0.051	**4.218**	14401	0.115	**1.469**	43459	0.046	0.917	0.232	**1.424**
		rIGA	96	0.012		14026	0.078		42378	0.051		0.163	
	4	IGA	92	0.114	**6.581**	14637	0.157	**1.696**	44109	0.081	0.837	0.375	**1.593**
		rIGA	96	0.017		14905	0.093		45318	0.097		0.235	
	5	IGA	96	0.217	**8.048**	15050	0.223	**1.989**	45511	0.131	0.830	0.598	**1.797**
		rIGA	94	0.026		14878	0.112		44958	0.158		0.332	
512^2	2	IGA	96	0.135	**3.361**	14127	0.344	**1.228**	42693	0.091	0.940	0.632	**1.291**
		rIGA	96	0.040		13495	0.280		40570	0.096		0.489	
	3	IGA	93	0.414	**6.771**	14980	0.521	**1.564**	45344	0.185	0.897	1.193	**1.721**
		rIGA	96	0.061		13881	0.333		41715	0.206		0.693	
	4	IGA	96	0.931	**11.39**	14647	0.735	**1.920**	44237	0.322	0.821	2.073	**2.117**
		rIGA	92	0.081		15162	0.383		45762	0.392		0.979	
	5	IGA	96	1.772	**14.09**	14449	1.053	**2.205**	43401	0.519	0.810	3.447	**2.462**
		rIGA	95	0.125		14465	0.477		43465	0.640		1.399	
1024^2	2	IGA	96	1.044	**4.050**	14642	1.526	**1.287**	44475	0.361	0.938	3.246	**1.486**
		rIGA	96	0.257		14717	1.185		44688	0.385		2.183	
	3	IGA	96	3.572	**10.48**	14503	2.370	**1.721**	43768	0.730	0.897	7.022	**2.369**
		rIGA	94	0.340		14551	1.377		43742	0.814		2.964	
	4	IGA	96	8.101	**17.46**	14942	3.457	**2.110**	45267	1.277	0.813	13.23	**3.137**
		rIGA	95	0.463		15259	1.638		46212	1.570		4.218	
	5	IGA	96	16.26	**25.28**	14245	4.894	**2.464**	42963	2.078	0.820	23.71	**4.067**
		rIGA	96	0.643		14294	1.985		43047	2.534		5.830	
2048^2	2	IGA	96	11.23	**6.671**	14526	14.77	**1.878**	43701	1.447	0.930	30.14	**2.240**
		rIGA	95	1.683		14563	7.868		43954	1.557		13.45	
	3	IGA	96	29.02	**14.33**	14351	19.09	**2.269**	42979	2.838	0.872	53.27	**3.369**
		rIGA	96	2.024		14566	8.415		43976	3.254		15.81	
	4	IGA	94	55.73	**21.43**	14453	22.31	**2.408**	43549	4.694	0.822	84.54	**4.011**
		rIGA	96	2.599		14240	9.263		42728	5.713		21.08	
	5	IGA	96	106.1	**34.38**	14312	26.73	**2.650**	43178	7.015	0.776	141.7	**5.536**
		rIGA	96	3.088		14419	10.08		43526	9.040		25.59	

these costs comparable. To compare results, Table 3 reports the number of executions of each operation for finding $N_0 = 2^{12}$ eigenpairs. We also report the average computational times per eigenvalue obtained by dividing the time of each numerical procedure by N_0 (i.e., $T_{\text{av}} = T_{N_0}/N_0$). Results indicate an improvement in the cost of matrix factorization close to $\mathcal{O}(p^2)$ for large problems, and of almost $\mathcal{O}(p)$ for f/b eliminations. We observe a slight degradation in cost of mat–vec multiplications due to the increase of nonzero terms of the mass matrix under rIGA. In summary, the total observed time saving for the entire eigensolution is up to $\mathcal{O}(p)$ for large domains. The time of finding $N_0 = 2^{12}$ discrete eigenpairs with $n_e = 2048$ and quintic basis functions reduces from 161 hours to 29 hours using the optimal rIGA discretization (see Figure 6d). If the domain is huge, the total computational cost is governed only by matrix factorization. Therefore, we predict the total time improvements of up to $\mathcal{O}(p^2)$. To observe them, we would need larger computational resources. On the other hand, for small problems (e.g., 2D systems with $n_e \leq 256$ and $p \leq 3$), the cost of IGA is comparable (or smaller) to that of rIGA, which occurs when the matrix factorization cost is a small fraction of the total cost.

Figure 7a shows the average time required to compute an eigenvalue, $T_{\text{av}} = T_{N_0}/N_0$. It also describes the contribution of each of the most expensive operations to the total time. Figure 7b demonstrates the same results after normalizing the time per eigenvalue by the number of degrees of freedom as $\widetilde{T} = T_{N_0}/N_0 N$. Both figures confirm that matrix factorization is the most decisive numerical operation of the eigenanalysis for large problems. rIGA with blocksize of 16 reduces the cost of the matrix factorization by a factor of up to $\mathcal{O}(p^2)$. It also improves the cost of f/b elimination, which is the governing cost in dealing with small problems. On the other hand, the cost of mat–vec

multiplication slightly increases when using rIGA, which is not as decisive as the improvements of the other two operations.



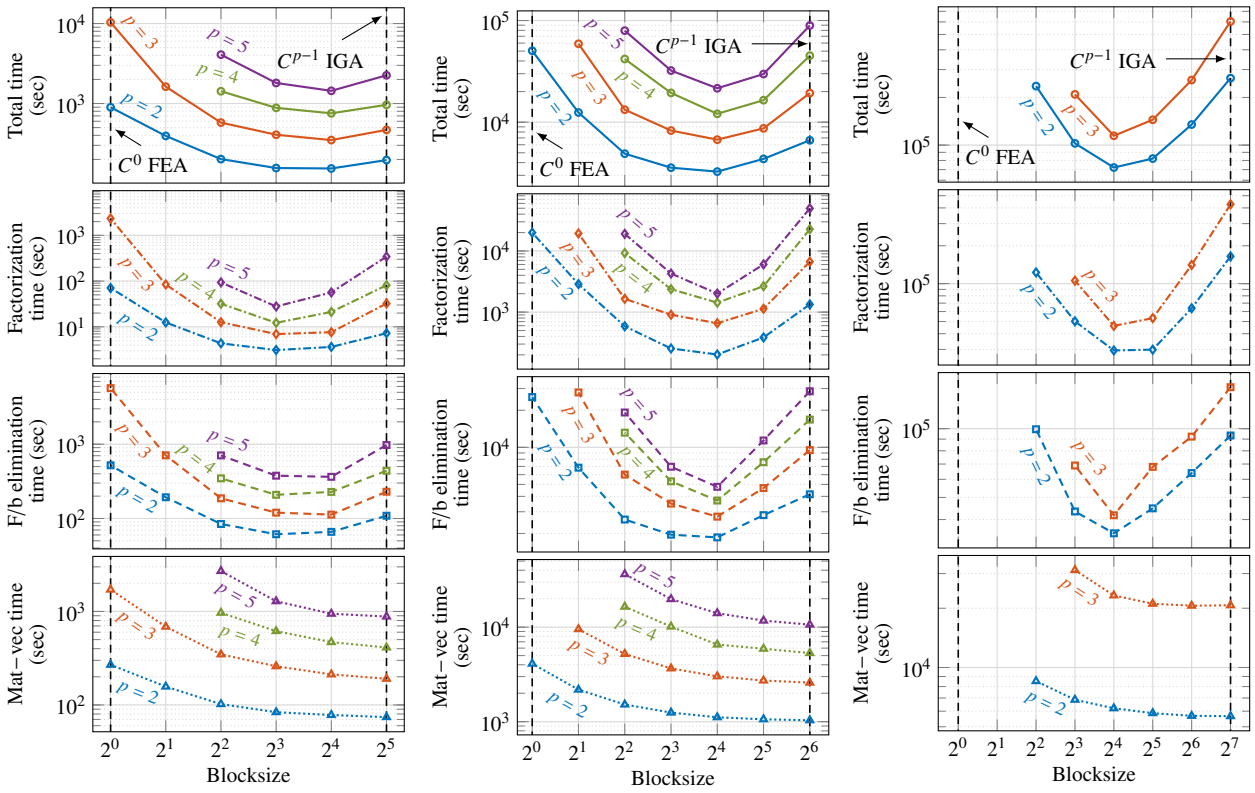
(a) Average time per eigenvalue

(b) Normalized time

Figure 7. Average time per eigenvalue, $T_{\text{av}} = T_{N_0}/N_0$, and normalized time, $\widetilde{T} = T_{N_0}/N_0 N$, versus the number of elements in each direction, n_e. 2D eigenproblems under maximum-continuity IGA (solid lines) and optimal rIGA (dashed/dotted lines) with blocksize of 16 elements in each direction.

7.2. Time performance of 3D eigenproblems

Figure 8 depicts the times required to compute $N_0 = 2^{10}$ eigenpairs of the Laplace operator in 3D versus different blocksizes of the rIGA discretizations. Similar to the 2D test cases, the factorization time of rIGA reaches a minimum

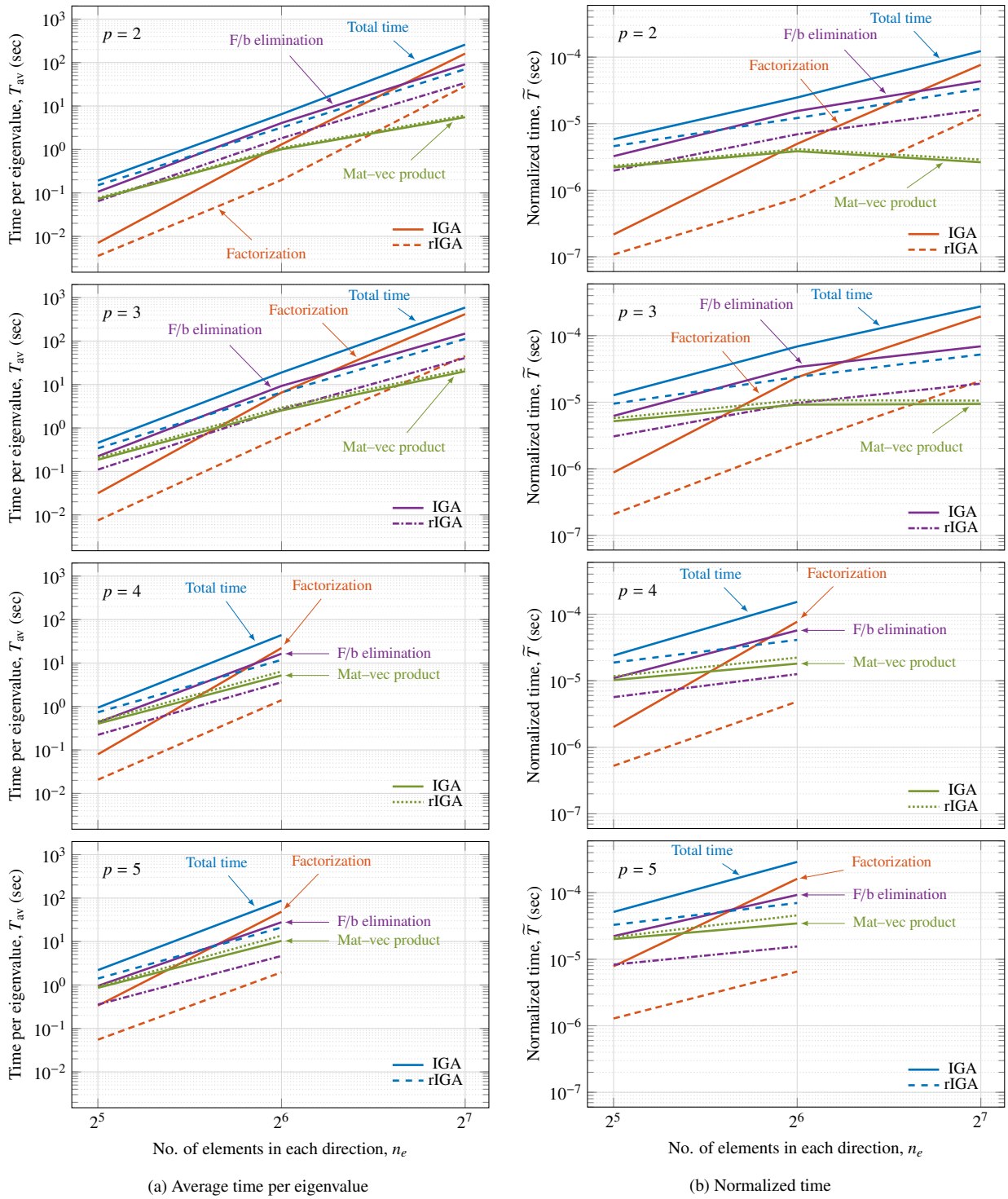


(a) Mesh: 32^3 (b) Mesh: 64^3 (c) Mesh: 128^3

Figure 8. Total times and those of the most expensive numerical operations for finding first $N_0 = 1024$ eigenpairs of the 3D Laplace operator with different mesh sizes $n_e = 2^s$ ($s = 5, 6, 7$) and degrees $p = 2, 3, 4, 5$. We test rIGA discretizations with different blocksizes $2^{s-\ell}$ obtained using different levels of partitioning $\ell = 0, 1, ..., s$. Since the eigenanalysis in 3D case is highly demanding, for $n_e = 128$, we only test up to $p = 3$.

Table 4. The average computational times per eigenvalue, $T_{\text{av}} = T_{N_0}/N_0$, and the number of executions of the most expensive operations for finding $N_0 = 1024$ eigenpairs of the 3D test cases illustrated in Figure 8. We consider the optimal blocksize of 16 elements. For $n_e = 32$, a higher improvement in matrix factorization is achievable with blocksize of 8 elements. However, the overall eigenanalysis is better improved with blocksize of 16 elements.

Discretization			Factorization			F/b elimination			Mat–vec product			Total	
Mesh	p	Method	$\mathcal{N}_{\text{fact}}$	$T_{\text{av,fact}}$ (sec)	Improved by	$\mathcal{N}_{\text{f/b}}$	$T_{\text{av,f/b}}$ (sec)	Improved by	$\mathcal{N}_{\text{m–v}}$	$T_{\text{av,m–v}}$ (sec)	Degraded by	T_{av} (sec)	Improved by
32^3	2	IGA	23	0.007	**1.994**	4425	0.106	**1.648**	13213	0.072	0.949	0.191	**1.273**
		rIGA	22	0.003		4376	0.064		13113	0.076		0.150	
	3	IGA	23	0.031	**4.234**	4584	0.223	**2.030**	13673	0.186	0.898	0.457	**1.340**
		rIGA	24	0.007		4556	0.110		13603	0.207		0.340	
	4	IGA	25	0.079	**3.831**	4725	0.429	**1.927**	14174	0.402	0.876	0.940	**1.277**
		rIGA	23	0.020		4633	0.222		13898	0.458		0.736	
	5	IGA	24	0.335	**6.077**	4735	0.952	**2.680**	14203	0.861	0.933	2.200	**1.563**
		rIGA	24	0.055		4607	0.355		13819	0.923		1.407	
64^3	2	IGA	25	1.305	**6.599**	4595	4.061	**2.237**	13707	1.010	0.929	6.486	**2.034**
		rIGA	23	0.197		4563	1.815		13567	1.087		3.187	
	3	IGA	22	6.517	**10.14**	4653	9.272	**3.466**	13958	2.526	0.857	18.87	**2.871**
		rIGA	22	0.642		4609	2.675		13827	2.946		6.573	
	4	IGA	24	22.14	**15.86**	4619	16.38	**4.536**	13856	5.192	0.812	44.00	**3.727**
		rIGA	23	1.396		4757	3.612		14270	6.391		11.80	
	5	IGA	24	48.57	**24.74**	4733	27.83	**5.961**	14197	10.38	0.756	87.24	**4.144**
		rIGA	22	1.963		4661	4.669		13981	13.72		21.04	
128^3	2	IGA	24	160.9	**5.593**	4781	91.16	**2.680**	14342	5.525	0.911	258.6	**3.669**
		rIGA	24	28.76		4607	34.00		13820	6.061		70.48	
	3	IGA	25	417.1	**9.254**	4684	148.3	**3.636**	14051	20.21	0.891	590.1	**5.275**
		rIGA	23	45.07		4673	40.79		14018	22.68		111.8	



(a) Average time per eigenvalue

(b) Normalized time

Figure 9. Average time of each numerical operation per eigenvalue, $T_{\text{av}} = T_{N_0}/N_0$, and respective normalized time, $\widetilde{T} = T_{N_0}/N_0 N$, versus the number of elements in each direction, n_e. 3D eigenproblems under maximum-continuity IGA (solid lines) and optimal rIGA (dashed/dotted lines) with blocksize of 16 elements in each direction (for degrees $p = 4$ and 5, we only test up to $n_e = 64$).

at blocksize of 16 elements expect for $n_e = 32$, where referring to Figure 8a, the optimal blocksize for matrix factorization is 8 elements (see the same inference in [23] for optimal blocksize of small and large domains). However, the maximum computational saving for the total elapsed time, considering all numerical operations of the eigenanalysis, is achieved by employing an rIGA discretization with blocksize of 16 elements in each direction. Table 4 reports the number of times we perform each operation when finding N_0 eigenpairs as well as the average computational times per eigenvalue, $T_{\text{av}} = T_{N_0}/N_0$. In 3D systems, we observe improvements of $\mathcal{O}(p^2)$ in matrix factorization and $\mathcal{O}(p)$ in

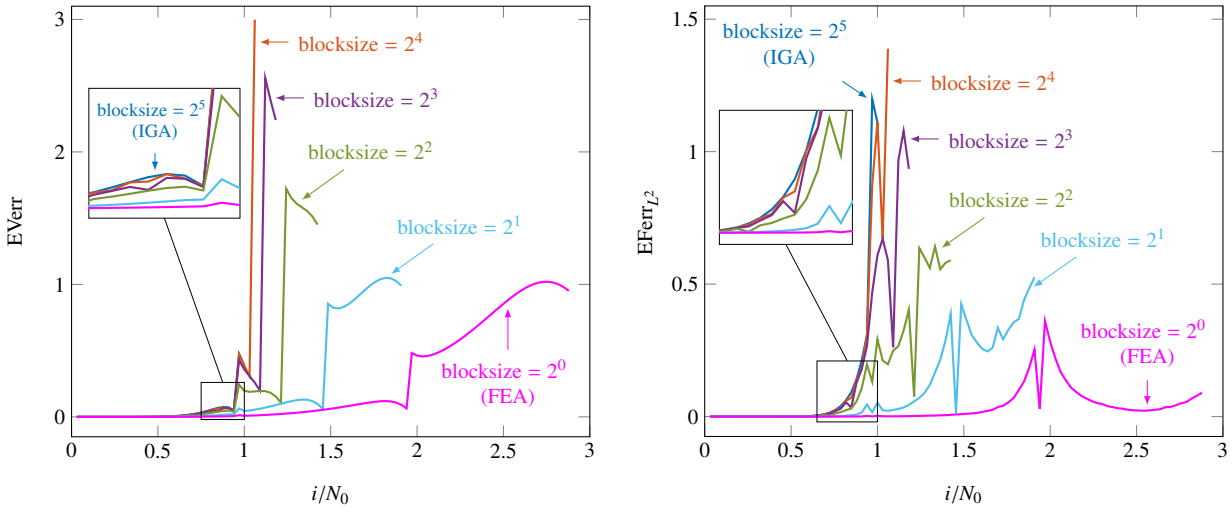
the total eigenanalysis when using rIGA with $n_e \geq 64$. We find $N_0 = 2^{10}$ eigenpairs with a 128^3 mesh and cubic bases in 168 hours using IGA and 32 hours using rIGA (see Figure 8c). However, we expect an improvement of $\mathcal{O}(p^2)$ for sufficiently large problems.

Figures 9a and 9b show the average time per computed eigenvalue, T_{av}, and the normalized time, $\widetilde{T}$. As in the 2D case, the f/b elimination dominates the total cost in small problems. Whereas for large problems, the matrix factorization is the most expensive procedure. However, to find a fixed number of eigenpairs, 3D problems employ more iterations than 2D ones (see Figure 5b), resulting in more f/b eliminations and mat–vecs for each spectral transform (i.e., shift). In our case, the number of f/b eliminations and mat–vec multiplications is close to $\mathcal{N}_{\text{f/b}} \approx 200\,\mathcal{N}_{\text{fact}}$ and $\mathcal{N}_{\text{m–v}} \approx 600\,\mathcal{N}_{\text{fact}}$ in 3D eigenproblems (see Table 4).

8. Accuracy assessment of eigensolution using rIGA

This section analyzes the effect of rIGA discretizations on the accuracy of the eigenanalysis. To assess this impact, we study the eigenvalue error, EVerr, and eigenfunction L^2 and energy norm errors, EFerr_{L^2} and EFerr_E, respectively, as defined in (9). The knot insertion steps of the rIGA approach add new control variables and, therefore, enrich the Galerkin space, modifying the spectral approximation properties of the IGA approach.

To investigate how rIGA affects the accuracy of eigenpairs throughout the entire spectrum, we first introduce a 1D eigenproblem with $n_e = 32$ and $p = 3$. Figure 10 depicts the eigenvalue and L^2 eigenfunction errors of maximum-continuity IGA versus those obtained by different partitioning levels of the rIGA approach. The abscissa of this figure shows the eigenmode numbers i normalized with respect to N_0. In here, we set N_0 as the total number of eigenmodes of the IGA discretization (i.e., $N_0 = N_{\text{IGA}}$). Since the rIGA-discretized system has more discrete eigenmodes, the spectra plots extend to $i/N_0 > 1$. The eigenvalue errors are computed by comparing the approximated values, λ^h, with exact ones, $\lambda^e = i^2\pi^2$, while the approximated eigenfunctions, u^h, are compared with $u^e = \sqrt{2}\sin(i\pi x)$. In terms of eigenvalues (see Figure 10a), rIGA discretizations of lower blocksize reach a lower error at the same mode number in the range of $i/N_0 \leq 1$, improving the outliers behavior of the original IGA-discretized system. By decreasing the blocksize, the eigenvalue errors converge to the acoustic branch of the FEA spectrum and we achieve a better approximation (see, e.g., [31]). However, for $i/N_0 > 1$ we observe larger errors for the outliers. We obtain similar conclusions in terms of eigenvector errors (see Figure 10b).

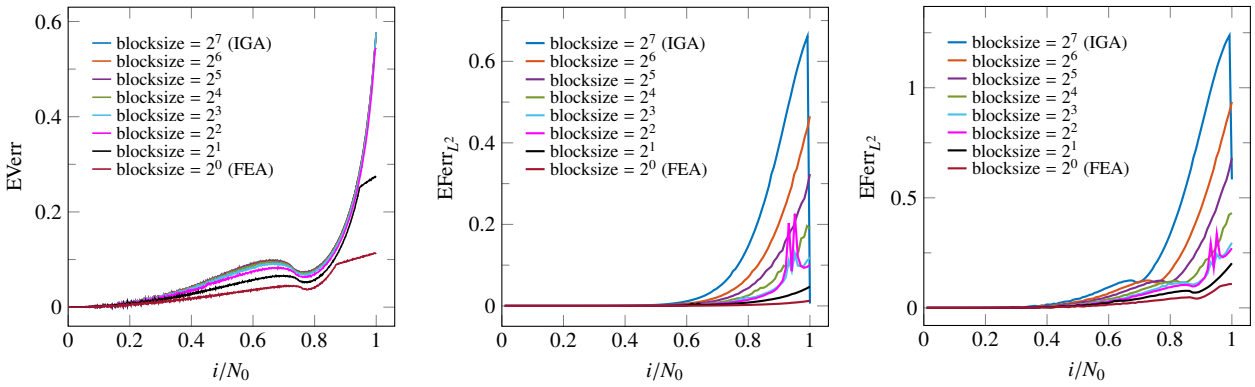


(a) Eigenvalue error

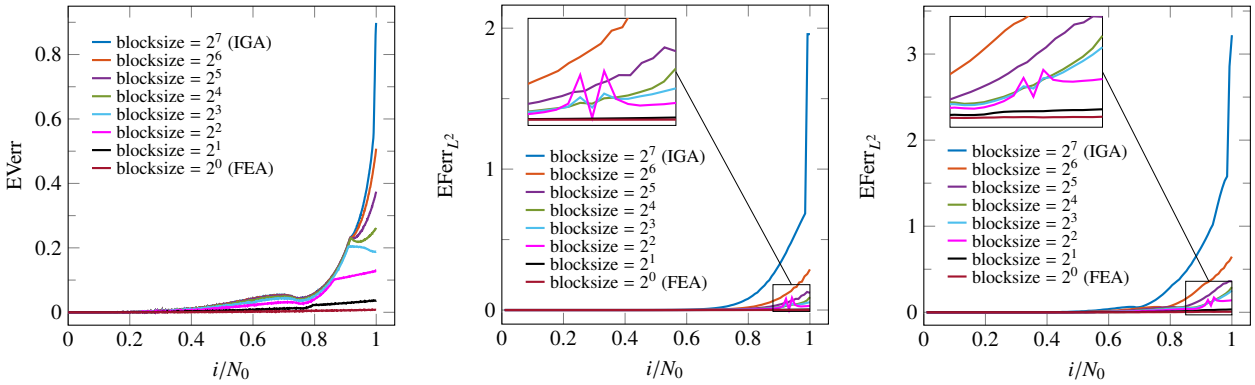
(b) Eigenfunction L^2 error

Figure 10. Eigenvalue and L^2 eigenfunction errors of the 1D eigenproblem discretized by $n_e = 32$ with cubic bases: maximum-continuity IGA versus rIGA of different blocksizes.

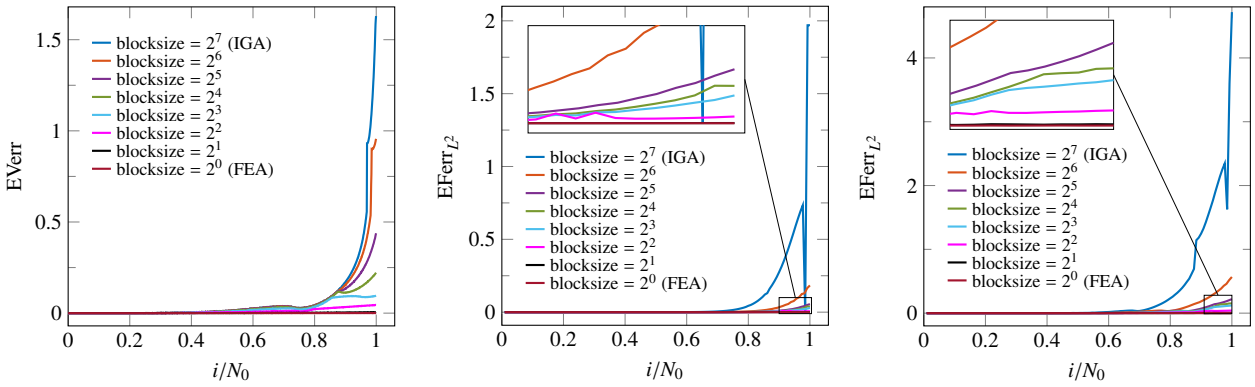
We now consider the accuracy of the eigenanalysis for 2D systems discretized with $n_e = 128$ elements in each direction and polynomial degrees of $p = 2, 3, 4, 5$. Equation (2) gives the exact eigenvalues and eigenfunctions. We use Remark 1 to compare the approximate eigenvalues and eigenfunctions, λ_i^h and u_i^h, respectively, with the tensor-represented exact ones λ_{ij}^e and u_{ij}^e. For any off-diagonal combination of i and j (i.e., $i \neq j$), u_{ij}^e and u_{ji}^e are generally



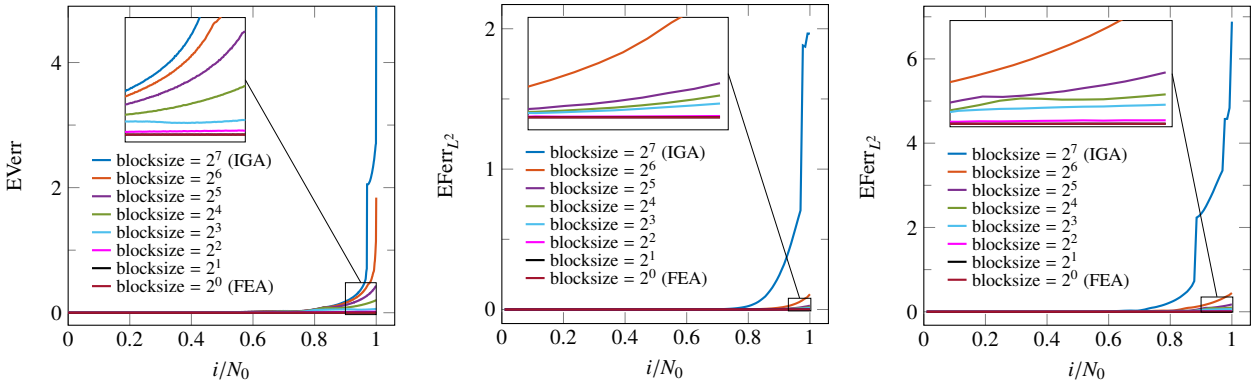
(a) $p = 2$



(b) $p = 3$



(c) $p = 4$



(d) $p = 5$

Figure 11. Eigenvalue error and eigenfunction L^2 and energy norm errors of the 2D eigenproblem discretized by $n_e = 128$ elements in each direction and polynomial degrees of $p = 2, 3, 4, 5$, obtained by maximum-continuity IGA and rIGA of different blocksizes.

referred to as *degenerate* eigenfunctions (see, e.g., [72]). Since degenerate eigenfunctions are not unique, we restrict our eigenfunction accuracy assessments to diagonal eigenfunctions. Figure 11 describes the eigenvalue error as well as eigenfunction L^2 and energy norm errors of the first $N_0 = N_{\text{IGA}}$ eigenpairs of the above-mentioned system. As for the 1D systems, rIGA discretizations improve the accuracy of eigenpairs approximation compared to the maximum-continuity IGA when $i/N_0 \leq 1$ (although we may see higher errors in the outliers for $i/N_0 > 1$). The figure shows that for higher polynomial degrees, the accuracy improvement of the rIGA eigensolution is more noticeable for eigenvalues closer to the end of the spectrum. It is clear that spectral approximation shows the best accuracy when using FEA since the FEA-discretized system has more degrees of freedom. However, as numerical results of Section 7 confirm, higher order FEA brings in costly eigencomputations, especially in large systems. Even computing a few eigenpairs needs an expensive matrix factorization. As a result, when dealing with moderate-to-large eigensystems, we suggest avoiding to reduce the continuity of basis functions further than that required by the optimal rIGA discretization.

9. Conclusions

We propose to use the refined isogeometric analysis (rIGA) discretizations to solve generalized Hermitian eigenproblems (GHEP). We compare the computational performance of rIGA versus that of maximum-continuity IGA when computing a fixed number of eigenpairs using a Lanczos eigensolver with a shift-and-invert spectral transform approach.

We consider two cases attending to the problem size. For large problems, namely, $N \geq 256^2$ in 2D and $N \geq 64^3$ in 3D, the most expensive operation is the matrix factorization, followed by matrix–vector operations. We compute the Cholesky factorization $\mathcal{O}(p^2)$ times faster with an rIGA discretization than with an IGA one. As a result, in the asymptotic regime, we theoretically reach an improvement of $\mathcal{O}(p^2)$ in the total computational time of the eigenanalysis. In our computations with N up to 2048^2 in 2D, and 128^3 in 3D, savings associated with the implementation of rIGA limit to $\mathcal{O}(p)$, as we need to analyze larger systems to observe the asymptotic behavior. In a 2D mesh with 2048^2 elements and $p = 5$, we need an average of 141 seconds to compute each eigenpair when using IGA, and only 26 seconds for rIGA. In a 3D mesh with 128^3 elements and $p = 3$, rIGA reduces the average required time per eigenmode from 590 seconds to 112 seconds.

For smaller problems (and with lower polynomial degrees), the forward/backward elimination is the most expensive numerical operation. This operation, called almost 150 and 200 times per each matrix factorization in our 2D and 3D cases, is theoretically up to $\mathcal{O}(p)$ faster with rIGA than with IGA for sufficiently large problems. For small problems, our improvement hardly reaches the expected rates. As a result, we suggest using the maximum-continuity IGA discretization for small problems.

Finally, we obtain better accuracy in spectral approximation using rIGA discretizations when computing the first N_0 eigenpairs, being N_0 the size of the IGA-discretized system. This accuracy improvement occurs because the continuity reduction of basis functions enriches the Galerkin space and modifies the approximation properties of the IGA approach. However, rIGA eigenvalues beyond N_0 show essential inaccuracies. This poor performance in the high-frequency regime may be detrimental for solving nonlinear problems.

Acknowledgment

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 777778 (MATHROCKS), the European POCTEFA 2014-2020 Project PIXIL (EFA362/19) by the European Regional Development Fund (ERDF) through the Interreg V-A Spain-France-Andorra programme, the Project of the Spanish Ministry of Science and Innovation with reference PID2019-108111RB-I00 (FEDER/AEI), the BCAM "Severo Ochoa" accreditation of excellence (SEV-2017-0718), and the Basque Government through the BERC 2018-2021 programme, the two Elkartek projects ArgIA (KK-2019-00068) and MATHEO (KK-2019-00085), the grant "Artificial Intelligence in BCAM number EXP. 2019/00432", and the Consolidated Research Group MATHMODE (IT1294-19) given by the Department of Education. This publication was also made possible in part by the CSIRO Professorial Chair in Computational Geoscience at Curtin University and the Deep Earth Imaging Enterprise Future Science Platforms of the Commonwealth Scientific Industrial Research

Organisation, CSIRO, of Australia. The Curtin Corrosion Centre and the Curtin Institute for Computation kindly provide ongoing support. The authors also acknowledge the Group of Applied Mathematical Modeling, Statistics, and Optimization (MATHMODE) at the University of the Basque Country (UPV/EHU) for providing HPC resources that have contributed to the research results reported in the paper.

References

[1] T. J. R. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (2005) 4135–4195, doi:10.1016/j.cma.2004.10.008.

[2] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T. J. R. Hughes, S. Lipton, M. A. Scott, T. W. Sederberg, Isogeometric analysis using T-splines, Computer Methods in Applied Mechanics and Engineering 199 (5-8) (2010) 229–263, doi:10.1016/j.cma.2009.02.036.

[3] F. Auricchio, F. Calabrò, T. J. R. Hughes, A. Reali, G. Sangalli, A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 249-252 (2012) 15–27, doi:10.1016/j.cma.2012.04.014.

[4] L. Dalcin, N. Collier, P. Vignal, A. M. A. Côrtes, V. M. Calo, PetIGA: A framework for high-performance isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 308 (2016) 151–181, doi:10.1016/j.cma.2016.05.011.

[5] A. F. Sarmiento, A. M. A. Côrtes, D. A. Garcia, L. Dalcin, N. Collier, V. M. Calo, PetIGA-MF: A multi-field high-performance toolbox for structure-preserving B-splines spaces, Journal of Computational Science 18 (2017) 117–131, doi:10.1016/j.jocs.2016.09.010.

[6] A. Reali, H. Gomez, An isogeometric collocation approach for Bernoulli–Euler beams and Kirchhoff plates, Computer Methods in Applied Mechanics and Engineering 284 (2015) 623–636, doi:10.1016/j.cma.2014.10.027.

[7] H. Casquero, L. Liu, Y. Zhang, A. Reali, J. Kiendl, H. Gomez, Arbitrary-degree T-splines for isogeometric analysis of fully nonlinear Kirchhoff–Love shells, Computer-Aided Design 82 (2017) 140–153, doi:10.1016/j.cad.2016.08.009.

[8] S. F. Hosseini, A. Hashemian, A. Reali, Studies on knot placement techniques for the geometry construction and the accurate simulation of isogeometric spatial curved beams, Computer Methods in Applied Mechanics and Engineering 360 (2020) 112705, doi:10.1016/j.cma.2019.112705.

[9] Y. Bazilevs, V. M. Calo, J. A. Cottrell, T. J. R. Hughes, A. Reali, G. Scovazzi, Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows, Computer Methods in Applied Mechanics and Engineering 197 (1-4) (2007) 173–201, doi:10.1016/j.cma.2007.07.016.

[10] L. F. R. Espath, A. F. Sarmiento, P. Vignal, B. O. N. Varga, A. M. A. Cortes, L. Dalcin, V. M. Calo, Energy exchange analysis in droplet dynamics via the Navier–Stokes–Cahn–Hilliard model, Journal of Fluid Mechanics 797 (2016) 389–430, doi:10.1017/jfm.2016.277.

[11] L. F. R. Espath, A. F. Sarmiento, L. Dalcin, V. M. Calo, On the thermodynamics of the Swift–Hohenberg theory, Continuum Mechanics and Thermodynamics 29 (2017) 1335–1345, doi:10.1007/s00161-017-0581-y.

[12] A. Hashemian, E. Lakzian, A. Ebrahimi-Fizik, On the application of isogeometric finite volume method in numerical analysis of wet-steam flow through turbine cascades, Computers & Mathematics with Applications 79 (6) (2020) 1687–1705, doi:10.1016/j.camwa.2019.09.025.

[13] Y. Bazilevs, V. M. Calo, Y. Zhang, T. J. R. Hughes, Isogeometric fluid–structure interaction analysis with applications to arterial blood flow, Computational Mechanics 38 (2006) 310–322, doi:10.1007/s00466-006-0084-3.

[14] H. Casquero, Y. J. Zhang, C. Bona-Casas, L. Dalcin, H. Gomez, Non-body-fitted fluid–structure interaction: Divergence-conforming B-splines, fully-implicit dynamics, and variational formulation, Journal of Computational Physics 374 (2018) 625–653, doi:10.1016/j.jcp.2018.07.020.

[15] H. Gómez, V. M. Calo, Y. Bazilevs, T. J. R. Hughes, Isogeometric analysis of the Cahn–Hilliard phase-field model, Computer Methods in Applied Mechanics and Engineering 197 (2008) 4333–4352, doi:10.1016/j.cma.2008.05.003.

[16] S. P. Clavijo, A. F. Sarmiento, L. F. R. Espath, L. Dalcin, A. M. A. Cortes, V. M. Calo, Reactive n-species Cahn–Hilliard system: A thermodynamically-consistent model for reversible chemical reactions, Journal of Computational and Applied Mathematics 350 (2019) 143–154, doi:10.1016/j.cam.2018.10.007.

[17] V. Puzyrev, M. Łoś, G. Gurgul, V. Calo, W. Dzwinel, M. Paszyński, Parallel splitting solvers for the isogeometric analysis of the Cahn-Hilliard equation, Computer Methods in Biomechanics and Biomedical Engineering 22 (2019) 1269–1281, doi:10.1080/10255842.2019.1661388.

[18] A. Buffa, G. Sangalli, R. Vázquez, Isogeometric analysis in electromagnetics: B-splines approximation, Computer Methods in Applied Mechanics and Engineering 199 (2010) 1143–1152, doi:10.1016/j.cma.2009.12.002.

[19] R. N. Simpson, Z. Liu, R. Vázquez, J. A. Evans, An isogeometric boundary element method for electromagnetic scattering with compatible B-spline discretizations, Journal of Computational Physics 362 (2018) 264–289, doi:10.1016/j.jcp.2018.01.025.

[20] N. Collier, D. Pardo, L. Dalcin, M. Paszynski, V. M. Calo, The cost of continuity: A study of the performance of isogeometric finite elements using direct solvers, Computer Methods in Applied Mechanics and Engineering 213-216 (2012) 353–361, doi:10.1016/j.cma.2011.11.002.

[21] N. Collier, L. Dalcin, D. Pardo, V. M. Calo, The Cost of Continuity: Performance of Iterative Solvers on Isogeometric Finite Elements, SIAM Journal on Scientific Computing 35 (2013) A767–A784, doi:10.1137/120881038.

[22] N. Collier, L. Dalcin, V. M. Calo, On the computational efficiency of isogeometric methods for smooth elliptic problems using direct solvers, International Journal for Numerical Methods in Engineering 100 (8) (2014) 620–632, doi:10.1002/nme.4769.

[23] D. Garcia, D. Pardo, L. Dalcin, M. Paszyński, N. Collier, V. M. Calo, The value of continuity: Refined isogeometric analysis and fast direct solvers, Computer Methods in Applied Mechanics and Engineering 316 (2017) 586–605, doi:10.1016/j.cma.2016.08.017.

[24] I. S. Duff, J. K. Reid, The Multifrontal Solution of Indefinite Sparse Symmetric Linear Equations, ACM Transactions on Mathematical Software 9 (1983) 302–325, doi:10.1145/356044.356047.

[25] D. Garcia, D. Pardo, L. Dalcin, V. M. Calo, Refined Isogeometric Analysis for a preconditioned conjugate gradient solver, Computer Methods in Applied Mechanics and Engineering 335 (2018) 490–509, doi:10.1016/j.cma.2018.02.006.

[26] D. Garcia, D. Pardo, V. M. Calo, Refined isogeometric analysis for fluid mechanics and electromagnetics, Computer Methods in Applied Mechanics and Engineering 356 (2019) 598–628, doi:10.1016/j.cma.2019.06.011.

[27] J. A. Cottrell, A. Reali, Y. Bazilevs, T. J. R. Hughes, Isogeometric analysis of structural vibrations, Computer Methods in Applied Mechanics and Engineering 195 (2006) 5257–5296, doi:10.1016/j.cma.2005.09.027.

[28] T. J. R. Hughes, A. Reali, G. Sangalli, Duality and unified analysis of discrete approximations in structural dynamics and wave propagation: Comparison of p-method finite elements with k-method NURBS, Computer Methods in Applied Mechanics and Engineering 197 (2008) 4104–4124, doi:10.1016/j.cma.2008.04.006.

[29] T. J. R. Hughes, J. A. Evans, A. Reali, Finite element and NURBS approximations of eigenvalue, boundary-value, and initial-value problems, Computer Methods in Applied Mechanics and Engineering 272 (2014) 290–320, doi:10.1016/j.cma.2013.11.012.

[30] M. Mazza, C. Manni, A. Ratnani, S. Serra-Capizzano, H. Speleers, Isogeometric analysis for 2D and 3D curl–div problems: Spectral symbols and fast iterative solvers, Computer Methods in Applied Mechanics and Engineering 344 (2019) 970–997, doi:10.1016/j.cma.2018.10.008.

[31] V. Puzyrev, Q. Deng, V. Calo, Dispersion-optimized quadrature rules for isogeometric analysis: Modified inner products, their dispersion properties, and optimally blended schemes, Computer Methods in Applied Mechanics and Engineering 320 (2017) 421–443, doi:10.1016/j.cma.2017.03.029.

[32] Q. Deng, V. Calo, Dispersion-minimized mass for isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 341 (2018) 71–92, doi:10.1016/j.cma.2018.06.016.

[33] S. F. Hosseini, A. Hashemian, A. Reali, On the application of curve reparameterization in isogeometric vibration analysis of free-from curved beams, Computers & Structures 209 (2018) 117–129, doi:10.1016/j.compstruc.2018.08.009.

[34] Q. Deng, M. Bartoň, V. Puzyrev, V. Calo, Dispersion-minimizing quadrature rules for C^1 quadratic isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 328 (2018) 554–564, doi:10.1016/j.cma.2017.09.025.

[35] V. Calo, Q. Deng, V. Puzyrev, Dispersion optimized quadratures for isogeometric analysis, Journal of Computational and Applied Mathematics 355 (2019) 283–300, doi:10.1016/j.cam.2019.01.025.

[36] Q. Deng, V. Puzyrev, V. Calo, Optimal spectral approximation of $2n$-order differential operators by mixed isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 343 (2019) 297–313, doi:10.1016/j.cma.2018.08.042.

[37] Q. Deng, V. Puzyrev, V. Calo, Isogeometric spectral approximation for elliptic differential operators, Journal of Computational Science 36 (2019) 100879, doi:10.1016/j.jocs.2018.05.009.

[38] M. Bartoň, V. Puzyrev, Q. Deng, V. Calo, Efficient mass and stiffness matrix assembly via weighted Gaussian quadrature rules for B-splines, Journal of Computational and Applied Mathematics 371 (2020) 112626, doi:10.1016/j.cam.2019.112626.

[39] R. G. Grimes, J. G. Lewis, H. D. Simon, A Shifted Block Lanczos Algorithm for Solving Sparse Symmetric Generalized Eigenproblems, SIAM Journal on Matrix Analysis and Applications 15 (1994) 228–272, doi:10.1137/S0895479888151111.

[40] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst, Templates for the Solution of Algebraic Eigenvalue Problems, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000.

[41] T. Ericsson, A. Ruhe, The Spectral Transformation Lanczos Method for the Numerical Solution of Large Sparse Generalized Symmetric Eigenvalue Problems, Mathematics of Computation 35 (1980) 1251–1268, doi:10.2307/2006390.

[42] B. Nour-Omid, B. N. Parlett, T. Ericsson, P. S. Jensen, How to implement the spectral transformation, Mathematics of Computation 48 (1987) 663–673, doi:10.1090/S0025-5718-1987-0878698-5.

[43] F. Xue, H. C. Elman, Fast inexact subspace iteration for generalized eigenvalue problems with spectral transformation, Linear Algebra and its Applications 435 (2011) 601–622, doi:10.1016/j.laa.2010.06.021.

[44] C. Campos, J. E. Roman, Strategies for spectrum slicing based on restarted Lanczos methods, Numerical Algorithms 60 (2012) 279–295, doi:10.1007/s11075-012-9564-z.

[45] G. Strang, G. J. Fix, An analysis of the finite element method, Prentice-Hall series in automatic computation, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[46] J. A. Cottrell, T. J. R. Hughes, Y. Bazilevs, Isogeometric Analysis: Toward Integration of CAD and FEA, John Wiley & Sons, Ltd, New York, NY, 2009.

[47] L. Piegl, W. Tiller, The NURBS Book, Springer-Verlag, New York, NY, 2nd edn., 1997.

[48] L. Gao, V. M. Calo, Preconditioners based on the Alternating-Direction-Implicit algorithm for the 2D steady-state diffusion equation with orthotropic heterogeneous coefficients, Journal of Computational and Applied Mathematics 273 (2015) 274–295, doi:10.1016/j.cam.2014.06.021.

[49] V. Hernandez, J. E. Roman, V. Vidal, SLEPc: A Scalable and Flexible Toolkit for the Solution of Eigenvalue Problems, ACM Transactions on Mathematical Software 31 (2005) 351–362, doi:10.1145/1089014.1089019.

[50] V. Hernández, J. E. Román, A. Tomás, V. Vidal, A Survey of Software for Sparse Eigenvalue Problems, Universitat Politecnica De Valencia, SLEPc Technical Report STR-6 (2009).

[51] R. B. Lehoucq, D. C. Sorensen, C. Yang, ARPACK Users Guide, Solution of Large-Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1998.

[52] B. N. Parlett, The Symmetric Eigenvalue Problem, Prentice-Hall, Inc., Upper Saddle River, NJ, 1998.

[53] G. W. Stewart, Matrix Algorithms, Volume II: Eigensystems, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001.

[54] E. S. Coakley, V. Rokhlin, A fast divide-and-conquer algorithm for computing the spectra of real symmetric tridiagonal matrices, Applied and Computational Harmonic Analysis 34 (2013) 379–414, doi:10.1016/j.acha.2012.06.003.

[55] D. C. Sorensen, Implicit Application of Polynomial Filters in a k-Step Arnoldi Method, SIAM Journal on Matrix Analysis and Applications 13 (1992) 357–385, doi:10.1137/0613025.

[56] K. Wu, H. Simon, Thick-Restart Lanczos Method for Large Symmetric Eigenvalue Problems, SIAM Journal on Matrix Analysis and Applications 22 (2000) 602–616, doi:10.1137/S0895479898334605.

[57] G. W. Stewart, A Krylov–Schur Algorithm for Large Eigenproblems, SIAM Journal on Matrix Analysis and Applications 23 (2002) 601–614, doi:10.1137/S0895479800371529.

[58] G. W. Stewart, Addendum to "A Krylov–Schur Algorithm for Large Eigenproblems", SIAM Journal on Matrix Analysis and Applications 24 (2002) 599–601, doi:10.1137/S0895479802403150.

[59] K. H. A. Olsson, A. Ruhe, Rational Krylov for eigenvalue computation and model order reduction, BIT Numerical Mathematics 46 (2006) 99–111, doi:10.1007/s10543-006-0085-9.

[60] S. Balay, W. D. Gropp, L. C. McInnes, B. F. Smith, Efficient Management of Parallelism in Object Oriented Numerical Software Libraries, in: E. Arge, A. M. Bruaset, H. P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkhäuser Press, 163–202, 1997.

[61] P. Vignal, L. Dalcin, D. L. Brown, N. Collier, V. M. Calo, An energy-stable convex splitting for the phase-field crystal equation, Computers & Structures 158 (2015) 355–368, doi:10.1016/j.compstruc.2015.05.029.

[62] A. M. A. Côrtes, A. L. G. A. Coutinho, L. Dalcin, V. M. Calo, Performance evaluation of block-diagonal preconditioners for the divergence-conforming B-spline discretization of the Stokes system, Journal of Computational Science 11 (2015) 123–136, doi:10.1016/j.jocs.2015.01.005.

[63] P. Vignal, N. Collier, L. Dalcin, D. L. Brown, V. M. Calo, An energy-stable time-integrator for phase-field models, Computer Methods in Applied Mechanics and Engineering 316 (2017) 1179–1214, doi:10.1016/j.cma.2016.12.017.

[64] A. F. Sarmiento, L. F. R. Espath, P. Vignal, L. Dalcin, M. Parsani, V. M. Calo, An energy-stable generalized-α method for the Swift–Hohenberg equation, Journal of Computational and Applied Mathematics 344 (2018) 836–851, doi:10.1016/j.cam.2017.11.004.

[65] E. Romero, J. E. Roman, A parallel implementation of Davidson methods for large-scale eigenvalue problems in SLEPc, ACM Transactions on Mathematical Software 40 (2014) 1–29, doi:10.1145/2543696.

[66] C. Campos, J. E. Roman, Parallel Krylov Solvers for the Polynomial Eigenvalue Problem in SLEPc, SIAM Journal on Scientific Computing 38 (2016) S385–S411, doi:10.1137/15m1022458.
[67] B. J. Faber, M. J. Pueschel, P. W. Terry, C. C. Hegna, J. E. Roman, Stellarator microinstabilities and turbulence at low magnetic shear, Journal of Plasma Physics 84 (2018) 905840503, doi:10.1017/s0022377818001022.
[68] M. Keçeli, F. Corsetti, C. Campos, J. E. Roman, H. Zhang, Á. Vázquez-Mayagoitia, P. Zapol, A. F. Wagner, SIESTA-SIPs: Massively parallel spectrum-slicing eigensolver for an *ab initio* molecular dynamics package, Journal of Computational Chemistry 39 (2018) 1806–1814, doi:10.1002/jcc.25350.
[69] J. C. Araujo C., C. Campos, C. Engström, J. E. Roman, Computation of scattering resonances in absorptive and dispersive media with applications to metal-dielectric nano-structures, Journal of Computational Physics 407 (2020) 109220, doi:10.1016/j.jcp.2019.109220.
[70] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, J. Koster, A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling, SIAM Journal on Matrix Analysis and Applications 23 (2001) 15–41, doi:10.1137/s0895479899358194.
[71] G. Karypis, V. Kumar, A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs, SIAM Journal on Scientific Computing 20 (1998) 359–392, doi:10.1137/s1064827595287997.
[72] H. J. Korsch, On the nodal behaviour of eigenfunctions, Physics Letters A 97 (1983) 77–80, doi:10.1016/0375-9601(83)90514-5.