Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 379 (2021) 113730
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

A matrix-free isogeometric Galerkin method for Karhunen–Loève approximation of random fields using tensor product splines, tensor contraction and interpolation based quadrature

Michal L. Mika[a,*], Thomas J.R. Hughes[b], Dominik Schillinger[a], Peter Wriggers[c], René R. Hiemstra[a]

[a] *Institut für Baumechanik und Numerische Mechanik, Leibniz Universität Hannover, Germany*
[b] *Oden Institute for Computational Engineering and Sciences, The University of Texas at Austin, United States of America*
[c] *Institut für Kontinuumsmechanik, Leibniz Universität Hannover, Germany*

Received 3 July 2020; received in revised form 24 January 2021; accepted 9 February 2021
Available online 18 March 2021

Abstract

The Karhunen–Loève series expansion (KLE) decomposes a stochastic process into an infinite series of pairwise uncorrelated random variables and pairwise L^2-orthogonal functions. For any given truncation order of the infinite series the basis is optimal in the sense that the total mean squared error is minimized. The orthogonal basis functions are determined as the solution of an eigenvalue problem corresponding to the homogeneous Fredholm integral equation of the second kind, which is computationally challenging for several reasons. Firstly, a Galerkin discretization requires numerical integration over a $2d$ dimensional domain, where d, in this work, denotes the spatial dimension. Secondly, the main system matrix of the discretized weak-form is dense. Consequently, the computational complexity of classical finite element formation and assembly procedures as well as the memory requirements of direct solution techniques become quickly computationally intractable with increasing polynomial degree, number of elements and degrees of freedom. The objective of this work is to significantly reduce several of the computational bottlenecks associated with numerical solution of the KLE. We present a matrix-free solution strategy, which is embarrassingly parallel and scales favorably with problem size and polynomial degree. Our approach is based on (1) an interpolation based quadrature that minimizes the required number of quadrature points; (2) an inexpensive reformulation of the generalized eigenvalue problem into a standard eigenvalue problem; and (3) a matrix-free and parallel matrix–vector product for iterative eigenvalue solvers. Two higher-order three-dimensional C^0-conforming multipatch benchmarks illustrate exceptional computational performance combined with high accuracy and robustness.

Keywords: Matrix-free solver; Kronecker products; Random fields; Fredholm integral eigenvalue problem; Isogeometric analysis

* Corresponding author.
E-mail addresses: mika@ibnm.uni-hannover.de (M.L. Mika), hughes@ices.utexas.edu (T.J.R. Hughes), schillinger@ibnm.uni-hannover.de (D. Schillinger), wriggers@ikm.uni-hannover.de (P. Wriggers), rene.hiemstra@ibnm.uni-hannover.de (R.R. Hiemstra).

https://doi.org/10.1016/j.cma.2021.113730
0045-7825/

1. Introduction

Most physical systems exhibit randomness, which, because of its lack of pattern or regularity, cannot be explicitly captured by deterministic mathematical models. The randomness may be due to the nature of the phenomenon itself, called *aleatoric* uncertainty, or due to a lack of knowledge about the system, referred to as *epistemic* uncertainty. In the latter the uncertainty may be reduced by obtaining additional data about the system at hand. An example of an epistemic uncertainty encountered in engineering is the fluctuation of material properties throughout a body, which occur due to the inhomogeneity of the medium. Deterministic mechanical models typically feature empirically derived material parameters, such as material stiffness and yield stress, that are assumed constant throughout the body. Their value is typically determined as a statistical volumetric average over a large set of laboratory specimens. This idealized model of reality may be insufficient in e.g. structural risk or reliability analysis and prediction, which is concerned with probabilities of violation of safety limits or performance measures, respectively [1]. In this case the effects of uncertainty on the result of a computation need to be quantified.

Uncertainty in physical quantities that vary in space and or time may be adequately modeled by *stochastic processes* or *random fields* [2]. This approach generalizes a deterministic system modeled by a partial differential equation to a stochastic system modeled by a stochastic partial differential equation or SPDE. Reliable predictions may be obtained by propagating uncertainties in input variables to those in the response. The main objective is to compute the response statistics, such as the mean and variance in the random solution field, or the probability that a set tolerance is exceeded. To compute these statistics it is necessary to discretize the SPDE, not only in space and time, but also in the stochastic dimensions. This can be a complicated task, not because of modeling randomness, but due to the *curse of dimensionality*. Every random variable contributes one dimension to the problem. Hence, it is important to keep their total to a minimum.

1.1. Discrete representation of random fields by the truncated Karhunen–Loève series expansion

One of the relevant questions in stochastic analysis is how to represent random fields discretely, in a manner suitable for use in numerical computation. The essential step is to break down the representation into a tractable number of mutually independent random variables, whose combination preserves the stochastic variability of the process [3,4]. One representation that is of particular interest is the truncated Karhunen–Loève series expansion or KLE [5,6]. The KLE decomposes a stochastic process into an infinite series of pairwise uncorrelated random variables and pairwise L^2-orthogonal basis functions. Truncating the series expansion after M terms yields the best M-term linear approximation of the random field, in the sense that the total mean squared error is minimized [7]. The KLE is useful in practice when satisfactory accuracy is attained with no more than 20–30 terms [3,8].

Computation of the truncated KLE requires the solution of a homogeneous Fredholm integral eigenvalue problem (IEVP) of the second kind. In general this is only possible numerically. The most popular numerical methods to solve IEVPs are the Nyström method, degenerate kernel methods and the collocation and Galerkin method [9,10]. The Galerkin method is widely regarded as superior due to its approximation properties and solid theoretical foundation. Specifically, it can be shown that the eigenvalues converge monotonically towards the exact eigenvalues and, by construction, that the modes preserve exactly the L^2-orthogonality property of the analytical mode-shapes [3].

1.2. Challenges in numerical solution of the KLE by means of the Galerkin method

Efficient solution of the KLE using the Galerkin method is a computationally challenging task [3]. The main challenges are the following:

(i) A Galerkin discretization requires numerical integration over a $2d$ dimensional domain, where d, in this work, denotes the spatial dimension. The computational complexity of classical finite element formation and assembly procedures scales as $\mathcal{O}\left(N_e^2(p+1)^{3d}\right)$, where N_e is the global number of elements, p the polynomial degree and d the spatial dimension.

(ii) The main system matrix of the discretized weak-form is dense and requires $\mathcal{O}\left(8N^2\right)$ bytes of memory in double precision arithmetic, where N is the dimension of the trial space.

(iii) Numerical solution requires one sparse backsolve $\mathcal{O}\left(N^2\right)$ and one dense matrix–vector product $\mathcal{O}\left(N^2\right)$ in each iteration of the eigenvalue solver, thus the solution time of the numerical eigenvalue solver scales $\mathcal{O}\left(N^2 \cdot N_{\text{iter}}\right)$, where N_{iter} is the number of iterations required by the Lanczos solver.

Table 1
Minimum memory required for storage of the main system matrix in the solution of the homogeneous Fredholm integral problem of the second kind assuming double-precision floating point arithmetic.

Number of degrees of freedom	10^3	10^4	10^5	10^6
Matrix storage	8 MB	800 MB	80 GB	8 TB

Table 1 illustrates that explicit storage of the dense system matrix requires impracticable amounts of memory for problems involving more than $100K$ degrees of freedom. Hence, the computational complexity of classical finite element formation and assembly procedures as well as memory requirements of direct solution techniques become quickly computationally intractable with increasing polynomial degree, number of elements and degrees of freedom.

There has been a particular research effort devoted to alleviating the disadvantages of the Galerkin method. In [11–13] an approximation by (Kronecker product) hierarchical matrices is used to efficiently compute the dense matrices as well as to reduce the memory requirements. These matrices are sparse and allow for matrix multiplication, addition and inversion in $\mathcal{O}(N \log N)$ time (or for Kronecker product hierarchical matrices in $\mathcal{O}(N)$ time) where N is the number of degrees of freedom. The generalized Fast Multipole Method, which also scales with $\mathcal{O}(N \log N)$, has been proposed in [14]. This method was shown to *not* yield significant speed-ups for p finite element methods and thus it is recommended for kernels of low regularity. Wavelet Galerkin-schemes [15] are also being used and can be coupled with compression techniques for boundary value problems [16], but have the disadvantage, that the number of eigenmodes to be computed must be known in advance. The pivoted Cholesky decomposition [17] focuses on approximating the discretized random fields with sufficiently fast decaying eigenvalues. In this case a truncation of the pivoted Cholesky decomposition of the covariance operator allows for an estimation of the eigenvalues in the post-processing step in $\mathcal{O}\left(M^2 N\right)$ time, where M is the truncation order of the Cholesky decomposition. One of the advantages of this method is the fact, that the number of eigenmodes required for a certain accuracy of the random field discretization can be estimated in advance.

1.3. Splines as a basis for random fields

Splines are piecewise polynomials with increased smoothness across element boundaries compared to classical finite elements. Traditionally, splines have been primarily used as shape functions in computer aided design. More recently, with the introduction of isogeometric analysis [18], splines have become more established as trial functions in finite element analysis. Although isogeometric analysis was originally introduced to improve the interoperability across several stages of the design to analysis process, it has proven its fidelity as an analysis technology. We refer to the monograph [19] and references contained therein for an exposition of isogeometric analysis applied to deterministic problems in structural and fluid mechanics.

More recently, spline based isogeometric analysis has found its way into the stochastic community. Stochastic methods have been proposed to quantify uncertainty due to material randomness in linear elasticity [20,21], static analysis of plates [22], vibrational analysis of shells [23], static and dynamic structural analysis of random composite structures [24] and functionally graded plates [25–27]. In [28] a method is proposed to quantify the effect due to uncertainty in shape. Of these, the methods proposed in [20,23,26,27] use isogeometric analysis within a spectral stochastic finite element framework [7], which is based on a KLE of random fields. The methods in [22,24,25] use perturbation series of which [22] expands random fields in terms of the KLE. Standard polynomial chaos is used in [28], while the methods in [21,29] and [30] discretize the stochastic dimensions in terms of splines. In particular, in [29] tensor product B-splines are used to expand stochastic variables, [21] proposes a spline-dimensional decomposition (SDD) and [30] proposes a spline chaos expansion, thus extending generalized polynomial chaos [31].

To the best of our knowledge, [32] is the first work in which splines have been used to approximate the truncated KLE. In his work the author applies a degenerate kernel approximation based on tensor product spline interpolation at the Greville abscissa. More recently, in the spirit of isogeometric analysis, non-uniform rational B-splines (NURBS) have been used to approximate the KLE using the Galerkin method [33] and the collocation method [34]. These methods avoid the geometrical errors in the representation of CAD geometry typically made within the classical finite element method. The authors note that the use of splines in the geometry description as

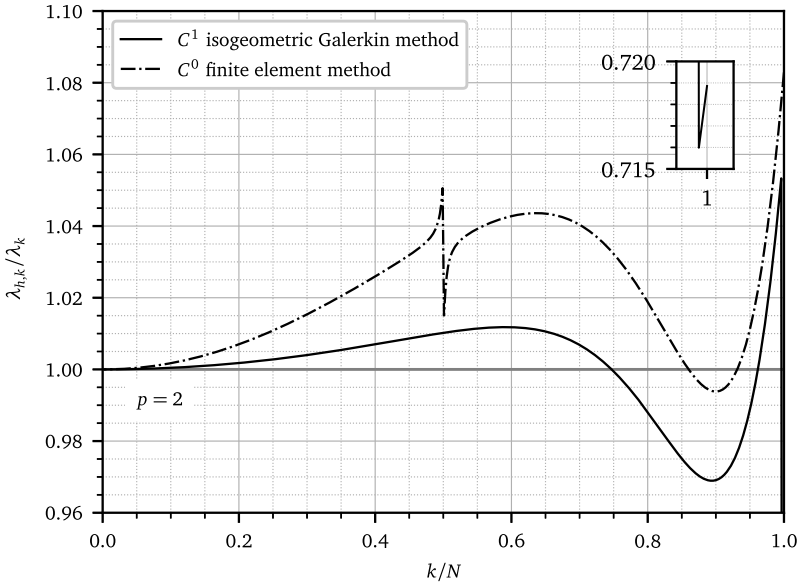


Fig. 1. Normalized discrete eigenvalues corresponding to a univariate Fredholm integral eigenvalue problem with an exponential kernel (correlation length is one). Comparison of eigenvalues obtained by C^1 quadratic splines to C^0 quadratic piecewise polynomials. Both methods employ a standard Galerkin projection based on full Gauss quadrature. The reference solution used to normalize the results is computed by the approach described in Remark 6.2.

well as in discretization of the spatial and stochastic dimensions could enable a "seamless uncertainty quantification pipeline".

In the context of the present work we would like to highlight the superior spectral approximation properties of smooth splines as compared to classical C^0 finite element shape functions. Several studies [35–38] have investigated the spectral approximation properties of splines in eigenvalue problems corresponding to second and fourth order differential operators and have demonstrated that splines have improved robustness and accuracy per degree of freedom across virtually the entire range of modes. The numerical results for the Fredholm integral eigenvalue problem are no different, as corroborated by the results shown in Fig. 1. It is precisely these properties that make splines appealing in the representation of random fields by means of the Karhunen–Loève expansion.

1.4. Contributions

We present a matrix-free isogeometric Galerkin method for Karhunen–Loève approximation of random fields by splines. Our solution methodology resolves several of the aforementioned computational bottlenecks associated with numerical solution of integral eigenvalue problems and enables solution of large-scale three-dimensional IEVPs on complex C^0-conforming multipatch domains. Below we summarize our main contributions.

Conversion to a standard eigenvalue problem

We have chosen a specific trial space of rational spline functions whose Gramian matrix has a Kronecker product structure independent of the geometric mapping. This enables us to perform the backsolve, used to convert the IEVP to standard form, in $\mathcal{O}\left(N \cdot N^{1/d}\right)$ time by utilizing standard linear algebra techniques from [39,40].

Interpolation based quadrature

We present an interpolation based quadrature technique designed and optimized specifically for the variational formulation of the Fredholm integral equation. The approach integrates a rich target space of functions with minimal number of quadrature points and outperforms existing competitive techniques in isogeometric analysis, such as quadrature by interpolation and table look-up [41,42] and weighted quadrature [43–45]. The proposed interpolation based quadrature technique is inspired by a similar technique used within linear finite elements in [46, Chapter 3.1.3]

and [13]. Instead, our approximation of the covariance function is based on higher order tensor product spline interpolation and resembles the kernel approximation made in [32]. Besides requiring as few quadrature points as possible, the interpolation based quadrature technique exposes Kronecker structure in the integral equations, reducing computational complexity significantly.

Matrix-free solution methodology

We present a matrix-free solution methodology to avoid explicit storage of the dense system matrix associated with numerical computation of the KLE. The matrix-free solution methodology not only reduces the memory complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$, but also significantly reduces the solution time. This is achieved by integrating the matrix-free solver with the proposed interpolation based quadrature technique. The latter exposes Kronecker structure in the resulting discrete integral equation, thereby reducing formation costs to $\mathcal{O}\left(N \cdot N^{1/d}\right)$ per iteration, leaving only the dense matrix–vector product that is associated with the Lanczos algorithm that remains $\mathcal{O}(N^2)$ per iteration. The integrative approach to quadrature and matrix-free solution techniques, exploiting Kronecker structure, is inspired by the matrix-free weighted quadrature method proposed recently in [45].

Open source implementation

We provide an open-source Python implementation of the described techniques that is available for download at https://github.com/m1ka05/tensiga. All benchmarks have been obtained using this implementation.

1.5. Outline

In Section 2 we briefly review the necessary mathematical and algorithmic background with regard to Kronecker products, B-splines and NURBS. In Section 3 we present the Karhunen–Loève series expansion of random fields and the weak formulation of the corresponding Fredholm integral eigenvalue problem of the second kind. In Section 4 we introduce our methodology for numerical solution of the truncated KLE. This includes reformulation of the eigenvalue problem to standard form, interpolation based quadrature of the weak form of the Fredholm integral problem, and a matrix-free algorithm with low computational complexity and minimal memory requirements that is embarrassingly parallelizable. The computational complexity is described in more detail in Section 5, where we compare our method with usual formation and assembly techniques used for standard Galerkin methods from the literature. Finally, in Section 6, we present a one-dimensional numerical study and several three-dimensional high-order numerical examples. A conclusion and an outlook with recommendations for future work are given in Section 7.

2. Background and notation

This section introduces some of the machinery that is used throughout the paper. The presented solution methodology for the Fredholm integral equation relies heavily on the properties of Kronecker products in combination with multidimensional tensor contraction [39]. We briefly review the main properties used in this work and illustrate their use in the Kronecker matrix–vector product. The Kronecker structure of the involved matrices is a direct consequence of the chosen tensor product spline function spaces. We briefly introduce B-splines as a basis for polynomial splines and Non-Uniform Rational B-splines (NURBS) for smooth geometrical mappings. For additional details we refer the reader to standard reference books [19,47].

2.1. Evaluation of the computational cost of an algorithm

The computational cost of the algorithms discussed in this work are evaluated in terms of *floating point operations per second* (*flops*). A single *flop* represents the amount of work required to preform one floating point addition, subtraction, multiplication or division [39]. Although the number of flops does not provide a complete assessment of the efficiency of an algorithm, it is widely used in the literature. Indeed, many other considerations such as cache-line efficiency and number of memory allocations can have a large impact on the performance of an algorithm. Typically, we are interested in the leading terms that dominate the computational cost of an algorithm and record the performance in terms of an order-of-magnitude estimate of the number of flops, written in Big-Oh notation as $\mathcal{O}(\cdot)$.

2.2. Kronecker products and tensor contraction

Let $\mathsf{A} \in \mathbb{R}^{m\times n}$, $\mathsf{B} \in \mathbb{R}^{p\times q}$ and $\mathsf{C} \in \mathbb{R}^{s\times t}$ denote real valued matrices. The Kronecker product $\mathsf{A}\otimes\mathsf{B} \in \mathbb{R}^{m\cdot p\times n\cdot q}$ is a matrix defined as

$$\mathsf{A}\otimes\mathsf{B} \coloneqq \begin{bmatrix} A_{11}\mathsf{B} & \cdots & A_{1n}\mathsf{B} \\ \vdots & & \vdots \\ A_{m1}\mathsf{B} & \cdots & A_{mn}\mathsf{B} \end{bmatrix} \tag{1}$$

Kronecker products satisfy the following properties

$$(\mathsf{A}\otimes\mathsf{B})\otimes\mathsf{C} = \mathsf{A}\otimes\big(\mathsf{B}\otimes\mathsf{C}\big) \qquad \text{(associativity)} \tag{2a}$$

$$(\mathsf{A}\otimes\mathsf{B})\big(\mathsf{C}\otimes\mathsf{D}\big) = \big(\mathsf{AC}\big)\otimes(\mathsf{BD}) \qquad \text{(mixed product property)} \tag{2b}$$

$$(\mathsf{A}\otimes\mathsf{B})^{-1} = \mathsf{A}^{-1}\otimes\mathsf{B}^{-1} \qquad \text{(inverse of a Kronecker product)} \tag{2c}$$

$$(\mathsf{A}\otimes\mathsf{B})^{\top} = \mathsf{A}^{\top}\otimes\mathsf{B}^{\top} \qquad \text{(transpose of a Kronecker product)} \tag{2d}$$

Let $\mathsf{X} \in \mathbb{R}^{n_1\times\cdots\times n_d}$, $\mathsf{Y} \in \mathbb{R}^{m_1\times\cdots\times m_d}$ denote two d-dimensional arrays. Vectorization of X is a linear operation that maps X to a vector $\mathrm{vec}(\mathsf{X}) \in \mathbb{R}^{n_1\cdot\ldots\cdot n_d}$ with entries

$$\mathrm{vec}(\mathsf{X})_i \coloneqq X_{i_1\ldots i_d}, \qquad \text{where } i = i_1 + (i_2-1)n_1 + (i_3-1)n_1\cdot n_2 + \cdots + (i_d-1)n_1\cdot\ldots\cdot n_{d-1}. \tag{3}$$

One recurring theme in this paper involving Kronecker matrices is efficient matrix–vector multiplication. Let $\mathsf{D}_k \in \mathbb{R}^{m_k\times n_k}$ denote a set of d matrices $\{D_{i_k j_k},\ k=1,\ldots,d\}$, $i_k = 1,\ldots,m_k$ and $j_k = 1,\ldots,n_k$. The matrix–vector product

$$\mathrm{vec}\,(\mathsf{Y}) = \Big(\mathsf{D}_d\otimes\cdots\otimes\mathsf{D}_1\Big)\mathrm{vec}\,(\mathsf{X}) \qquad \mathcal{O}(M\cdot N)\text{ flops} \tag{4a}$$

can be written as a tensor contraction instead

$$Y_{i_1\cdots i_d} = \sum_{j_1\cdots j_d} D_{i_1 j_1}\cdots D_{i_d j_d} X_{j_1\cdots j_d} \qquad \mathcal{O}(\max\,(N\cdot m_1,\ n_d\cdot M))\text{ flops} \tag{4b}$$

Here $N = n_1\cdot\ldots\cdot n_d$ and $M = m_1\cdot\ldots\cdot m_d$. The second approach scales nearly linearly with matrix size and significantly outperforms standard matrix–vector multiplication which scales quadratically with the matrix size. In practice, highly optimized linear tensor algebra libraries can be used to perform the tensor contraction such as the tensor algebra compiler (TACO) [48]. Our Python implementation uses Numpy's reshaping and matrix–matrix product routines, which call low-level BLAS routines. The implemented reshapes do not require any expensive and unnecessary data copies.

2.3. B-splines

Consider a d-dimensional parametric domain $\hat{\mathcal{D}} = [0,1]^d \subset \mathbb{R}^d$ with local coordinates $\hat{x} = (\hat{x}_1,\ldots,\hat{x}_d)$. Let $(\ B_{i_k,p_k}(\hat{x}_k),\ \ i_k = 1,\ldots,n_k\ )$ denote the univariate B-spline basis of polynomial degree p_k and dimension n_k, corresponding to the kth parametric coordinate $\hat{x}_k$. We consider multivariate B-splines as tensor products of univariate B-splines

$$B_{\mathsf{i}}(\hat{x}) = \prod_{k=1}^{d} B_{i_k,p_k}(\hat{x}_k), \quad \mathsf{i} \coloneqq (i_1,\ldots,i_d). \tag{5}$$

Here $\mathsf{i}\in\mathcal{I}$ is a multi-index in the set $\mathcal{I} \coloneqq \{(i_1,\ldots,i_d) : 1\le i_k\le n_k\}$. The collection of all multivariate B-spline basis functions spans the space

$$\mathcal{B}_h \coloneqq \mathrm{span}\left\{B_{\mathsf{i}}(\hat{x})\right\}_{\mathsf{i}\in\mathcal{I}}. \tag{6}$$

It is important to note that splines allow for increased continuity between polynomial elements as compared to classical C^0-continuous finite element basis functions. This turns out to have significant impact on the spectral accuracy of the Galerkin method. This is evidenced by several studies [35–38] and will be discussed in some detail in this work.

2.4. Geometrical mapping

Let $F : \hat{\mathcal{D}} \to \mathcal{D}$ map a point $\hat{x}$ from the parametric domain $\hat{\mathcal{D}}$ to a point x in the physical domain $\mathcal{D}$. We assume that the map F and its inverse are smooth such that the Jacobian matrix $\left[\mathrm{D}F(\hat{x})\right]_{ij} \coloneqq \frac{\partial F_i}{\partial \hat{x}_j}$ and its inverse are well-defined. In this work F is represented as a linear combination of Non-Uniform Rational B-splines (NURBS). NURBS are rational functions of B-splines that enable representation of common engineering shapes with conic sections, which cannot be represented by polynomial B-splines [47]. The discretization method presented in this work makes heavy use of tensor product properties of the involved function spaces. Since NURBS do not have a tensor product structure, we use them only to represent the geometry and do not consider them as a basis for the function spaces.

3. Isogeometric Galerkin discretization of the Karhunen–Loève series expansion

The Karhunen–Loève series expansion (KLE) decomposes a stochastic process or field into an infinite linear combination of L^2-orthogonal functions and uncorrelated stochastic random variables. In this section we present the probability theory underlying the KLE and discuss its discretization by means of the Galerkin method.

3.1. Karhunen–Loève expansion of random fields

Consider a complete probability space $(\Theta, \Sigma, \mathbb{P})$. Here Θ denotes a sample set of random events, Σ is the σ-algebra of Borel subsets of Θ and $\mathbb{P}$ is a probability measure $\mathbb{P} : \Sigma \to [0, 1]$. A random field $\alpha(\cdot, \theta) : \Theta \mapsto L^2(\mathcal{D})$ on a bounded domain $\mathcal{D} \in \mathbb{R}^d$ is a collection of deterministic functions of $x \in \mathcal{D}$, called realizations, that are indexed by events $\theta \in \Theta$. A subset of realizations $\alpha(\cdot, \Theta_s),\ \Theta_s \in \Sigma$, has a probability of occurrence of $\mathbb{P}(\Theta_s)$.

Let $\mathbb{E}[\cdot]$ denote the expectation operator corresponding to the probability measure $\mathbb{P}$. Assuming $\alpha \in L^2(\mathcal{D} \times \Theta)$ its first and second order moments exist and are given by

$$\mu(x) \coloneqq \mathbb{E}[\alpha(x, \theta)] \quad \text{and} \tag{7a}$$

$$\Gamma(x, x') \coloneqq \mathbb{E}\left[(\alpha(x, \theta) - \mu(x))(\alpha(x', \theta) - \mu(x'))\right]. \tag{7b}$$

Here μ is called the mean or expected value of α over all possible realizations, and $\Gamma : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ is called its *covariance function* or *kernel*. By definition, the kernel is bounded, symmetric and positive semi-definite [7]. Because the kernel is square integrable, that is, $\Gamma \in L^2(\mathcal{D} \times \mathcal{D})$, it is in fact a Hilbert–Schmidt kernel, see [49].

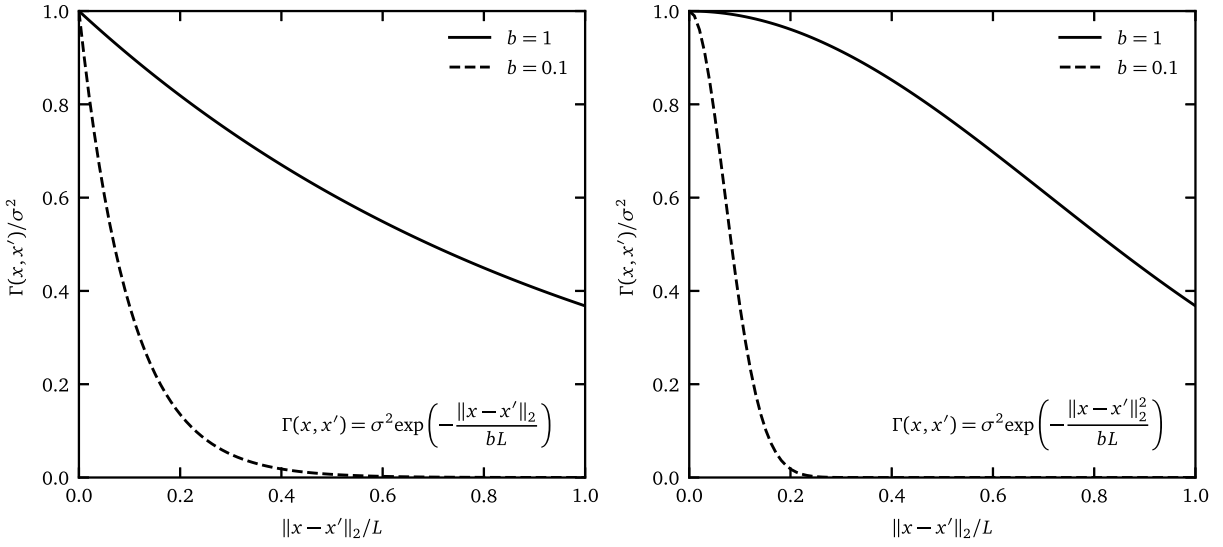
A random field is *stationary* or *homogeneous* if its statistical properties do not vary as a function of the position $x \in \mathcal{D}$. This implies that the covariance function can be written as a function of the difference $x - x'$. Furthermore, for *isotropic* random fields the statistical properties are invariant under rotations, which means the covariance is a function of Euclidean distance $\|x - x'\|_2$.

Remark 3.1. Although, the Euclidean distance is widely used in the literature its use is not always justified. In general, the geodesic distance, i.e. the shortest distance between points x and x' along all paths contained in $\mathcal{D}$, is the true measure of distance. The Euclidean distance can vary significantly from the geodesic distance especially if the correlation length is relatively large and the domain is non-convex. The geodesic distance is, however, difficult and expensive to compute, which explains its non-use. In this work we also use the Euclidean distance measure and assume its choice is a reasonable one in the context of the applied numerical benchmark problems.

Fig. 2 shows two common examples of covariance functions that correspond to stationary isotropic random fields: the *exponential* and the *Gaussian* or *squared exponential kernel*. Important parameters that influence the locality of these correlation functions are the variance σ^2 and correlation length bL. Here L denotes a characteristic length and b is a dimensionless factor.

The KLE of a random field $\alpha(\cdot, \theta)$ requires the solution of an integral eigenvalue problem. Consider the linear operator

$$T : L^2(\mathcal{D}) \mapsto L^2(\mathcal{D}), \quad (T\phi)(x) \coloneqq \int_{\mathcal{D}} \Gamma(x, x')\phi(x')\,\mathrm{d}x'. \tag{8}$$



(a) Exponential kernel

(b) Squared exponential kernel

Fig. 2. The exponential and squared exponential (Gaussian) covariance functions for different correlation lengths with $b = \{0.1, 1.0\}$. Note the difference in the continuity of both kernels at $x = x'$. The exponential kernel is C^0, while the square exponential kernel is C^∞ at $x = x'$.

The operator T is compact. In fact, T is a Hilbert–Schmidt operator, since the covariance function is a Hilbert–Schmidt kernel. Furthermore, since the covariance function is positive semi-definite and symmetric [7], T is a self-adjoint positive semi-definite linear operator. The eigenfunctions $\{\phi_i\}_{i\in\mathbb{N}}$ of T are defined by the homogeneous Fredholm integral eigenvalue problem of the second kind,

$$T\phi_i = \lambda_i\phi_i, \quad \phi_i \in L^2(\mathcal{D}) \text{ for } i \in \mathbb{N}. \tag{9}$$

The important properties of the eigenpairs are (1) the normalized eigenfunctions $\{\phi_i\}_{i\in\mathbb{N}}$ are orthonormal, that is, $(\phi_i, \phi_j)_{L^2(\mathcal{D})} = \delta_{ij}$, and thus form a basis for $L^2(\mathcal{D})$; and (2) the corresponding eigenvalues form a sequence $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$, which in general decays with increasing mode number.

Because Γ in (7b) is symmetric and positive semi-definite, it possesses the spectral decomposition [50,51]

$$\Gamma(x, x') = \sum_{i=1}^{\infty} \lambda_i\phi_i(x)\phi_i(x'). \tag{10}$$

With these definitions, the KLE of a random field $\alpha \in L^2(\mathcal{D} \times \Theta)$ is defined by the following series [5]

$$\alpha(x, \theta) = \mu(x) + \sum_{i=1}^{\infty} \sqrt{\lambda_i}\phi_i(x)\xi_i(\theta), \quad \text{where} \quad \xi_i(\theta) := \frac{1}{\sqrt{\lambda_i}} \int_{\mathcal{D}} (\alpha(x, \theta) - \mu(x))\, \phi_i(x)\, \mathrm{d}x. \tag{11}$$

While $\{\phi_i\}_{i\in\mathbb{N}}$ are pairwise L^2-orthogonal on $\mathcal{D}$, the $\{\xi_i\}_{i\in\mathbb{N}}$ are pairwise uncorrelated zero-mean random variables [7]. For this reason the KL expansion is sometimes said to be bi-orthogonal.

3.2. Truncated Karhunen–Loève expansion

In order to represent a random field in a discrete numerical computation it is necessary to discretize the continuous probability space. This can be achieved by truncating the KLE after M terms and thus reducing the dimension of the stochastic space to M uncorrelated random variables

$$\tilde{\alpha}_M(x, \theta) = \mu(x) + \sum_{i=1}^{M} \sqrt{\lambda_i}\phi_i(x)\xi_i(\theta). \tag{12}$$

The mean of a random field is not affected by the discretization. The variance of the discretization on the other hand can be derived from the spectral decomposition in Eq. (10)

$$\mathbb{E}\left[(\tilde{\alpha}_M(x,\theta)-\mu(x))^2\right]=\sum_{i=1}^{M}\lambda_i\phi_i^2(x). \tag{13}$$

The variance of the discretized random field converges uniformly in $\mathcal{D}$ and in $L^2(\Theta,\Sigma,\mathbb{P})$ towards the true variance [33]

$$\lim_{M\to\infty}\sum_{i=1}^{M}\lambda_i\phi_i^2(x)=\Gamma(x,x). \tag{14}$$

Furthermore, it can be shown that the KLE is optimal with respect to the global mean-squared error among all series expansions of truncation order M [7].

3.3. Variational formulation

The variational formulation or weak form of the integral eigenvalue problem introduced in Eq. (9) states
Find $\{\lambda,\phi\}\in\mathbb{R}_0^+\times L^2(\mathcal{D})$ *such that*

$$\int_{\mathcal{D}}\left(\int_{\mathcal{D}'}\Gamma(x,x')\phi(x')\,\mathrm{d}x'-\lambda\phi(x)\right)\psi(x)\,\mathrm{d}x=0\quad\forall\psi\in L^2(\mathcal{D}). \tag{15}$$

Confining the solution to the finite-dimensional subspace $\mathcal{S}_h\subset L^2(\mathcal{D})$ we obtain the discrete variational formulation
Find $\{\lambda,\phi\}\in\mathbb{R}_0^+\times\mathcal{S}_h$ *such that*

$$\int_{\mathcal{D}}\left(\int_{\mathcal{D}'}\Gamma(x,x')\phi_h(x')\,\mathrm{d}x'-\lambda_h\phi_h(x)\right)\psi_h(x)\,\mathrm{d}x=0\quad\forall\psi_h\in\mathcal{S}_h. \tag{16}$$

This is the Galerkin method for the homogeneous Fredholm integral eigenvalue problem of the second kind [33,52]. Within the trial space under consideration, the Galerkin method produces the best L^2 approximation of the analytical modes. The resulting discrete modes preserve exactly the L^2 orthogonality property of the analytical mode-shapes. Furthermore, it can be shown that a variational treatment using the Galerkin method leads to eigenvalues that converge monotonically, under mesh refinement, towards the true eigenvalues [7].

3.4. Choice of the trial space

The choice of the trial space $\mathcal{S}_h$ provides some freedom in the design of the Galerkin method. The recently proposed isogeometric Galerkin method for the KLE of random fields uses NURBS for the test and trial spaces [33]. This choice is motivated by the fact that the geometrical mapping is defined using NURBS and it is natural to remain within the isoparametric paradigm. This method shares the same technical challenges as all classical Galerkin methods applied to this class of problems [11,33]: the formation and assembly costs, which have a time complexity of $\mathcal{O}\left(N_e^2\cdot p^{3d}\right)$, as well as the storage requirements, which have space complexity of $\mathcal{O}\left(N^2\right)$, become quickly intractable with increasing number of elements N_e, polynomial degree p, dimension d and number of degrees of freedom N. A practical Galerkin method must address these difficulties in the design of the method.

We abandon the isoparametric concept and choose a different space to represent the finite-dimensional solution. Our choice offers multiple computational advantages without sacrificing higher-order accuracy and robustness. We define the trial space for the Galerkin method as

$$\mathcal{S}_h:=\operatorname{span}\left\{\frac{B_{\mathrm{i}}(\hat{x})}{\sqrt{\det\mathrm{D}F(\hat{x})}}\right\}_{\mathrm{i}\in\mathcal{I}}. \tag{17}$$

Because the geometrical mapping F is smooth and invertible the Jacobian determinant is never singular, that is, $\det\mathrm{D}F(\hat{x})>0$ for all $\hat{x}\in\hat{\mathcal{D}}$. Importantly, the functions are linearly independent due to linear independence of

B-splines. In general, however, these basis functions will not form a partition of unity. Instead, the characterizing property is that products of these functions are integral preserving, that is, they transform as volume forms

$$\int_{\mathcal{D}} \frac{B_{\mathrm{i}}(\hat{x})}{\sqrt{\det \mathrm{D}F(\hat{x})}} \frac{B_{\mathrm{j}}(\hat{x})}{\sqrt{\det \mathrm{D}F(\hat{x})}} \,\mathrm{d}x = \int_{\mathcal{D}} \frac{B_{\mathrm{i}}(\hat{x}) B_{\mathrm{j}}(\hat{x})}{\det \mathrm{D}F(\hat{x})} \,\mathrm{d}x = \int_{\hat{\mathcal{D}}} B_{\mathrm{i}}(\hat{x}) B_{\mathrm{j}}(\hat{x}) \,\mathrm{d}\hat{x}.$$

3.5. Matrix formulation

After substituting the desired subspace for the test and trial functions and performing minor algebraic manipulations, the discretized Galerkin method results in a generalized algebraic eigenvalue problem

$$\mathsf{A}\mathsf{v}_h = \lambda_h \mathsf{Z}\mathsf{v}_h, \tag{18}$$

where the system matrices are formed by evaluating

$$\begin{aligned}\mathsf{A}_{\mathrm{ij}} &= \int_{\hat{\mathcal{D}}} \int_{\hat{\mathcal{D}}'} \Gamma(x(\hat{x}), x(\hat{x}')) \frac{B_{\mathrm{i}}(\hat{x})}{\sqrt{\det \mathrm{D}F(\hat{x})}} \frac{B_{\mathrm{j}}(\hat{x}')}{\sqrt{\det \mathrm{D}F(\hat{x}')}} \det \mathrm{D}F(\hat{x}) \det \mathrm{D}F(\hat{x}') \,\mathrm{d}\hat{x}' \mathrm{d}\hat{x} \\ &= \int_{\hat{\mathcal{D}}} \int_{\hat{\mathcal{D}}'} \Gamma(x(\hat{x}), x(\hat{x}')) B_{\mathrm{i}}(\hat{x}) B_{\mathrm{j}}(\hat{x}') \sqrt{\det \mathrm{D}F(\hat{x}) \det \mathrm{D}F(\hat{x}')} \,\mathrm{d}\hat{x}' \mathrm{d}x' \end{aligned} \tag{19}$$

and

$$\begin{aligned}\mathsf{Z}_{\mathrm{ij}} &= \int_{\hat{\mathcal{D}}} \frac{B_{\mathrm{i}}(\hat{x})}{\sqrt{\det \mathrm{D}F(\hat{x})}} \frac{B_{\mathrm{j}}(\hat{x})}{\sqrt{\det \mathrm{D}F(\hat{x})}} \det \mathrm{D}F(\hat{x}) \,\mathrm{d}\hat{x} \\ &= \int_{\hat{\mathcal{D}}} B_{\mathrm{i}}(\hat{x}) B_{\mathrm{j}}(\hat{x}) \,\mathrm{d}\hat{x}. \end{aligned} \tag{20}$$

As a result of the chosen solution space, the mass matrix Z has a Kronecker structure and can be decomposed into $k = 1, \ldots, d$ univariate mass matrices

$$\mathsf{Z}_k \coloneqq Z_{i_k j_k} = \int_0^1 B_{i_k, p_k}(\hat{x}_k) B_{j_k, p_k}(\hat{x}_k) \,\mathrm{d}\hat{x}_k, \qquad i_k, j_k = 1, \ldots, n_k. \tag{21}$$

The system mass matrix Z can be then written as

$$\mathsf{Z} = \mathsf{Z}_d \otimes \cdots \otimes \mathsf{Z}_1. \tag{22}$$

Instead of computing and storing the matrix Z, we precompute and store the matrices Z_k, $k = 1, \ldots, d$. Furthermore, it is the Kronecker structure that allows us to inexpensively reformulate the generalized eigenvalue problem to a standard algebraic eigenvalue problem.

Remark 3.2. In practice Z_k in (21) is computed exactly up to machine precision using Gauss–Legendre numerical quadrature with $p + 1$ quadrature points per element, where p is the polynomial degree in component direction k. For alternative ways of computing integrals of piecewise polynomial products see [53]. Because the domain of integration is one-dimensional the formation and assembly costs of $\mathcal{O}(n_e p^3)$ as well as the storage costs of $\mathcal{O}(pn)$ bytes are negligible compared to the total solver costs. Here n_e is the number of univariate elements and n is the univariate number of degrees of freedom in component direction k.

3.6. Discretization of multipatch geometries

Single patch domains can only represent simple geometric models. In general multipatch domains need to be considered. Because the integral operator in (8) does not involve derivatives, it does not require any smoothness from the finite element spaces. Hence, the techniques presented in this paper are valid for multi-patch domains. The examples in this paper involve C^0-conforming multi-patch domains. Alternatively, non-conforming C^{-1} discretizations are also possible with minor change.

4. Efficient matrix-free solution strategy

There are two major challenges when applying the Galerkin method to discretize the homogeneous Fredholm integral eigenvalue problem in (9). Firstly, the variational formulation requires integration over a $2d$-dimensional domain to evaluate the matrix entries in A. This leads to formation and assembly costs with complexity $\mathcal{O}(N_e^2(p+1)^{3d})$, where N_e is the global number of elements, p the polynomial degree and d the spatial dimension. Secondly, because the matrix is dense, A requires insurmountable memory storage for any practical problem of interest. Several techniques have been presented in the literature in order to deal with these challenges, for example by approximation with low-rank matrices like the hierarchical matrices [11–13] or by using Fast Multipole Methods [14]. In this work we present a combination of four techniques to deal with the aforementioned challenges:

1. Reformulation of the generalized eigenvalue problem into an equivalent standard eigenvalue problem;
2. Interpolation based quadrature for variational formulations of integral equations;
3. Efficient formation of finite element arrays based on Kronecker matrix–vector product;
4. Formulation of a matrix-free and parallel matrix–vector product for the Lanczos algorithm.

The reformulation into a standard algebraic eigenvalue problem significantly reduces the computational cost and simultaneously improves conditioning. By exploiting the Kronecker structure of the right-hand-side mass matrix we can perform this reformulation with negligible overhead. The proposed non-standard quadrature technique that we call *interpolation based integration* is tailored for variational formulations of integral equations. The technique is optimal in the sense that few quadrature points are required while integrating a rich space of tensor product functions on the $2d$-dimensional domain $\mathcal{D}\times\mathcal{D}$. Importantly, the technique lends itself to multidimensional tensor contraction due to the Kronecker structure of the involved matrices. This significantly speeds up the evaluation of integrals over high-dimensional domains and scales favorably with polynomial degree. Finally, all techniques are combined within a matrix-free evaluation scheme that is embarrassingly parallel and requires minimal memory storage. The formation and assembly costs of our approach are negligible compared to the remaining solver costs of the Lanczos eigenvalue solver, which is $\mathcal{O}(\tilde{N}^2\cdot N_{\text{iter}}/N_{\text{thread}})$. Here $\tilde{N}$ is the global number of degrees of freedom of the interpolation space, N_{iter} is the number of iterations of the eigensolver and N_{thread} is the number of simultaneous processes. In the following we discuss each of the proposed techniques in more detail.

4.1. Reformulation into a standard algebraic eigenvalue problem

Let us consider a Cholesky factorization of the mass matrix $\mathsf{Z}=\mathsf{L}\mathsf{L}^\top$ and define a linear transformation of the eigenvectors $\mathsf{v}'_h := \mathsf{L}^\top\mathsf{v}_h$. The generalized eigenvalue problem can then be rewritten (see [40, Chapter 9.2.2]) as a standard eigenvalue problem with unchanged eigenvalues corresponding to new eigenvectors v'_h

$$\mathsf{A}'\mathsf{v}'_h=\lambda_h\mathsf{v}'_h,\quad \text{where}\quad \mathsf{A}':=\mathsf{L}^{-1}\mathsf{A}\mathsf{L}^{-\top}. \tag{23}$$

It is expected that the new system matrix A' has improved conditioning compared to A. The kernel is positive-definite and symmetric. In practice, it often quickly tends to zero for increasing distance $\|x-x'\|_2$. In the limiting case, where $\Gamma(x,x')\to\delta(x,x')$, the system matrix $\mathsf{A}\to\mathsf{Z}$ and hence the preconditioner would be ideal.

Although improved conditioning is beneficial, the main reason for the chosen transformation is efficiency. Solution of a standard algebraic eigenvalue problem is much less expensive than solution of a generalized eigenvalue problem. The transformation itself is inexpensive. Using the Kronecker structure of Z and the properties (2b) and (2d) we may write

$$\begin{aligned}\mathsf{Z}&=\mathsf{Z}_d\otimes\cdots\otimes\mathsf{Z}_1\\&=\mathsf{L}_d\mathsf{L}_d{}^\top\otimes\cdots\otimes\mathsf{L}_1\mathsf{L}_1{}^\top\\&=(\mathsf{L}_d\otimes\cdots\otimes\mathsf{L}_1)\,(\mathsf{L}_d\otimes\cdots\otimes\mathsf{L}_1)^\top=\mathsf{L}\mathsf{L}^\top.\end{aligned} \tag{24}$$

Here $\mathsf{L}_k\mathsf{L}_k^\top,\ k=1,\dots,d$, denote the Cholesky factorizations corresponding to the univariate mass matrices Z_k. Hence, instead of performing the Cholesky factorization for the complete system matrix $\mathsf{Z}\in\mathbb{R}^{N\times N}$, which is the standard procedure in most solvers for generalized algebraic eigenvalue problems, we merely need the Cholesky factorizations for $\mathsf{Z}_k\in\mathbb{R}^{n_k\times n_k},\ k=1,\dots,d$.

The factorization is precomputed once before using it in the eigenvalue solver. The associated computational cost is reduced from $\mathcal{O}\left(N^3\right)$ to $\mathcal{O}\left(n^3\right)$ flops, where $n = \max(n_1, \ldots, n_d)$ and $N = n_1 \cdot \ldots \cdot n_d$. Subsequently, the cost of applying the factorization in a single iteration of the eigenvalue solver is reduced from $\mathcal{O}\left(N^2\right)$ to $\mathcal{O}\left(n \cdot N\right)$. Besides a reduction in computational cost, this approach significantly reduces the required memory storage.

4.2. Interpolation based quadrature for integral equations

One of the most straightforward ways of improving efficiency is to design quadrature rules that require fewer evaluation points. This is especially true for the variational formulation of the Fredholm integral equation which requires numerical integration over a $2d$-dimensional domain. In practice, accurate and efficient quadrature rules are designed as follows. First, one chooses a space of functions $\mathbb{T}$, called the target space for numerical quadrature, whose elements should be exactly integrated by the new quadrature rule. If this space is in some sense rich enough then the error due to the quadrature can be bounded by the discretization error, which is needed to show optimal rates of convergence of the numerical method, see [54]. The next objective is to find a quadrature rule that requires as few points as possible to integrate all functions in $\mathbb{T}$.

We present a non-standard quadrature technique that generalizes the approach to quadrature presented for linear finite elements in [13,46] to higher order splines. Our approach is tailored towards evaluating integrals found in variational formulations of integral equations and achieves a very low number of evaluation points while integrating exactly a rich space of functions. The target space $\mathbb{T}$ is chosen such that the quadrature scheme exactly evaluates the integral

$$\int_{\hat{\mathcal{D}}} \int_{\hat{\mathcal{D}}'} \tilde{G}(\hat{x}, \hat{x}') B_{\mathsf{i}}(\hat{x}) B_{\mathsf{j}}(\hat{x}') \, \mathrm{d}\hat{x}' \mathrm{d}\hat{x}, \qquad \tilde{G} \in \tilde{\mathcal{B}}_h(\hat{\mathcal{D}}) \otimes \tilde{\mathcal{B}}_h(\hat{\mathcal{D}}'). \tag{25}$$

using $\tilde{N}^2$ points. Here $\tilde{\mathcal{B}}_h$ is another d-dimensional spline space that can be chosen independently of $\mathcal{B}_h$ and $\tilde{N}$ is its dimension. In practice this space can be chosen to fit well with the integrand in the variational formulation of the integral equation. This provides additional flexibility to the quadrature scheme.

Because $\tilde{G} \in \tilde{\mathcal{B}}_h(\hat{\mathcal{D}}) \otimes \tilde{\mathcal{B}}_h(\hat{\mathcal{D}}')$ is a real-valued $2d$-variate spline function it can be expanded in terms of B-spline basis functions and real-valued coefficients $\{\tilde{G}_{\mathsf{kl}}\}_{\mathsf{k},\mathsf{l} \in \tilde{\mathcal{I}}}$ as

$$\tilde{G}(\hat{x}, \hat{x}') := \sum_{\mathsf{k},\mathsf{l} \in \tilde{\mathcal{I}}} \tilde{G}_{\mathsf{kl}} \tilde{B}_{\mathsf{k}}(\hat{x}) \tilde{B}_{\mathsf{l}}(\hat{x}') \quad \text{with} \quad \mathsf{k}, \mathsf{l} \in \tilde{\mathcal{I}} := \{(i_1, \ldots, i_d) : 1 \leq i_k \leq \tilde{n}_k\}.$$

Comparing the multidimensional integrand in (19) with the one in (25) we may conclude that the degrees of freedom $\{\tilde{G}_{\mathsf{kl}}\}_{\mathsf{k},\mathsf{l} \in \tilde{\mathcal{I}}}$ should be chosen such that $\tilde{G}$ is a good approximation of the function $G : \hat{\mathcal{D}} \times \hat{\mathcal{D}} \mapsto \mathbb{R}^+$ defined as

$$G(\hat{x}, \hat{x}') := \hat{\Gamma}(\hat{x}, \hat{x}') \sqrt{\det \mathrm{D}F(\hat{x}) \det \mathrm{D}F(\hat{x}')}. \tag{26}$$

Here $\hat{\Gamma}(\hat{x}, \hat{x}')$ is the pull-back of the kernel $\Gamma(x, x')$ from the physical to the parametric space using the geometrical mapping F.

Remark 4.1. To maintain optimal accuracy in numerical quadrature, locally, the smoothness of the interpolation space should not exceed smoothness of the integrand in (26). Indeed, Fig. 3b shows that error convergence due to quadrature of an exponential kernel, which features reduced regularity at $x = x'$, is suboptimal. Similarly, the reduced regularity due to the Jacobian determinant term needs to be taken into account when choosing the interpolation space. Being set in the framework of splines and isogeometric analysis, our method provides sufficient flexibility in enforcing required smoothness.

The approximation $\tilde{G}$ can be estimated in different ways. We follow a similar approach to the degenerate kernel approximation in [32] and choose to collocate G at the Greville abscissa [55]. This approach is both simple and combines high order accuracy with a minimal number of evaluation points. We note that related ideas based on quasi-interpolation have been presented in [56–58] for formation and assembly of boundary integral equations and in [41,42] for matrix assembly in Galerkin discretization of PDEs.

Let $\tilde{\mathsf{B}} = \tilde{B}_{\mathsf{ij}} := \tilde{B}_{\mathsf{j}}(\hat{x}_{\mathsf{i}})$ denote the d-variate spline collocation matrix evaluated at the Greville abscissa $\hat{x}_{\mathsf{i}} \in \hat{\mathcal{D}}$, $\mathsf{i} \in \tilde{\mathcal{I}}$. The interpolation problem states

Find $\{\tilde{G}_{\mathsf{kl}}\}_{\mathsf{k,l}\in\tilde{\mathcal{I}}}$ *such that*

$$\sum_{\mathsf{k,l}\in\tilde{\mathcal{I}}} \tilde{G}_{\mathsf{kl}}\tilde{B}_{\mathsf{k}}(\hat{x}_{\mathsf{i}})\tilde{B}_{\mathsf{l}}(\hat{x}'_{\mathsf{j}}) = G(\hat{x}_{\mathsf{i}},\hat{x}'_{\mathsf{j}}) \quad \forall\, \mathsf{i},\mathsf{j}\in\tilde{\mathcal{I}}. \tag{27}$$

This is equivalent to the matrix problem $\mathsf{G} = \tilde{\mathsf{B}}\tilde{\mathsf{G}}\tilde{\mathsf{B}}^{\top}$. Hence, the matrix of coefficients can be computed as $\tilde{\mathsf{G}} = \tilde{\mathsf{B}}^{-1}\mathsf{G}\tilde{\mathsf{B}}^{-\top}$. The computational cost of the interpolation can be significantly reduced from $\mathcal{O}(\tilde{N}^3)$ to $\mathcal{O}(\tilde{n}\cdot\tilde{N})$ flops, where $\tilde{n} = \max(\tilde{n}_1,\dots,\tilde{n}_d)$, by exploiting the Kronecker structure of $\tilde{\mathsf{B}}$. We decompose $\tilde{\mathsf{B}}$ into d univariate collocation matrices $\tilde{\mathsf{B}}_k$, $k = 1,\dots,d$, and use property (2c) to write its inverse as

$$\tilde{\mathsf{B}}^{-1} = \tilde{\mathsf{B}}_d^{-1}\otimes\cdots\otimes\tilde{\mathsf{B}}_1^{-1} \quad \text{with} \quad \tilde{\mathsf{B}}_k := \tilde{B}_{i_k j_k} = \tilde{B}_{j_k,\tilde{p}_k}(\hat{x}_{i_k}). \tag{28}$$

In practice we compute d LU factorizations, each corresponding to a univariate matrix $\tilde{\mathsf{B}}_k$, to apply the inverse of $\tilde{\mathsf{B}}$ to a vector. Note, that this approach is similar to the approach we took in (24) for the Cholesky factorization of Z.

4.3. Matrix formation

By substituting G in (19) with $\tilde{G}$ we can approximate matrix A by a matrix $\tilde{\mathsf{A}}$ with entries

$$\begin{aligned}\tilde{A}_{\mathsf{ij}} &:= \int_{\hat{\mathcal{D}}}\int_{\hat{\mathcal{D}}'} \tilde{G}(\hat{x},\hat{x}')B_{\mathsf{i}}(\hat{x})B_{\mathsf{j}}(\hat{x}')\,\mathrm{d}\hat{x}'\mathrm{d}\hat{x}\\ &= \sum_{\mathsf{k,l}\in\tilde{\mathcal{I}}} \tilde{G}_{\mathsf{kl}}\int_{\hat{\mathcal{D}}}\int_{\hat{\mathcal{D}}'} \tilde{B}_{\mathsf{k}}(\hat{x})\tilde{B}_{\mathsf{l}}(\hat{x}')B_{\mathsf{i}}(\hat{x})B_{\mathsf{j}}(\hat{x}')\,\mathrm{d}\hat{x}'\mathrm{d}\hat{x}\\ &= \sum_{\mathsf{k,l}\in\tilde{\mathcal{I}}} \tilde{G}_{\mathsf{kl}}\int_{\hat{\mathcal{D}}} \tilde{B}_{\mathsf{k}}(\hat{x})B_{\mathsf{i}}(\hat{x})\,\mathrm{d}\hat{x}\int_{\hat{\mathcal{D}}'} \tilde{B}_{\mathsf{l}}(\hat{x}')B_{\mathsf{j}}(\hat{x}')\,\mathrm{d}\hat{x}'.\end{aligned}$$

Hence, using the tensor product structure of the interpolation space we have separated the $2d$ dimensional integral into a product of two d dimensional integrals. In matrix notation we may write $\tilde{\mathsf{A}} = \mathsf{M}^{\top}\tilde{\mathsf{G}}\mathsf{M}$, or

$$\tilde{A}_{\mathsf{ij}} = \sum_{\mathsf{k,l}\in\tilde{\mathcal{I}}} \tilde{G}_{\mathsf{kl}}M_{\mathsf{ki}}M_{\mathsf{lj}}. \tag{29}$$

Here $\mathsf{M} := M_{\mathsf{ij}}$ is a mass matrix

$$M_{\mathsf{ij}} = \int_{\hat{\mathcal{D}}} \tilde{B}_{\mathsf{i}}(\hat{x})B_{\mathsf{j}}(\hat{x})\,\mathrm{d}\hat{x}. \tag{30}$$

Similarly, as we did for matrix Z in (22), we can exploit the Kronecker structure and decompose M into d univariate mass matrices $\mathsf{M}_k := M_{i_k j_k}$

$$\mathsf{M} = \mathsf{M}_d\otimes\cdots\otimes\mathsf{M}_1 \quad \text{with} \quad M_{i_k j_k} = \int_0^1 \tilde{B}_{i_k,\tilde{p}_k}(\hat{x}_k)B_{j_k,p_k}(\hat{x}_k)\,\mathrm{d}\hat{x}_k. \tag{31}$$

As in the case of Z_k, $k = 1,\dots,d$, these univariate matrices are computed up to machine precision as discussed in Remark 3.2. The approximation error $\mathsf{A}-\tilde{\mathsf{A}}$ is entirely due to the interpolation error $G-\tilde{G}$. Hence, accurate approximation of G should result in an accurate approximation of A.

4.4. Matrix-free solution strategy

The interpolation based quadrature technique introduced in the previous section involves computation of the matrix of coefficients $\tilde{\mathsf{G}} := \tilde{G}_{\mathsf{kl}}$. This matrix is dense and has $\tilde{N}^2$ entries. Consequently, storage of $\tilde{\mathsf{G}}$ is just as inconvenient as storing $\tilde{\mathsf{A}}$ and becomes quickly intractable with problem size. In this section we propose a matrix-free evaluation of the matrix–vector product $\mathsf{v}'\mapsto\tilde{\mathsf{A}}\mathsf{v}'$ that does not require explicit access to matrix $\tilde{\mathsf{G}}$ or $\tilde{\mathsf{A}}$.

4.4.1. Basic setup

We have the following standard algebraic eigenvalue problem

$$\tilde{\mathsf{A}}'\mathsf{v}' = \lambda_h \mathsf{v}'. \tag{32a}$$

Here, the system matrix $\tilde{\mathsf{A}}'$ can be written as

$$\tilde{\mathsf{A}}' = \mathsf{L}^{-1}\mathsf{M}^{\top}\tilde{\mathsf{B}}^{-1}\mathsf{J}\Gamma\mathsf{J}\tilde{\mathsf{B}}^{-\top}\mathsf{M}\mathsf{L}^{-\top}, \tag{32b}$$

where J is a diagonal matrix with diagonal entries given by the square roots of Jacobian determinants evaluated at the Greville abscissa, and B, M and L are all Kronecker product matrices. Consequently, a matrix–vector product with any of these matrices can be performed close to linear time complexity. The matrix–vector product $\mathsf{v}' \mapsto \tilde{\mathsf{A}}'\mathsf{v}'$ can be subdivided into the following operations

$$\Gamma := \hat{\Gamma}(\hat{x}_{\mathsf{k}}, \hat{x}'_{\mathsf{l}}) \qquad \text{(Evaluation of the kernel at the Greville abscissa)} \tag{32c}$$

$$\mathsf{G} = \mathsf{J}\Gamma\mathsf{J} \qquad \text{(Scaling of the kernel)} \tag{32d}$$

$$\tilde{\mathsf{G}} = \tilde{\mathsf{B}}^{-1}\mathsf{G}\tilde{\mathsf{B}}^{-\top} \qquad \text{(Interpolation of the scaled kernel)} \tag{32e}$$

$$\tilde{\mathsf{A}} = \mathsf{M}^{\top}\tilde{\mathsf{G}}\mathsf{M} \qquad \text{(Evaluation of the integrals)} \tag{32f}$$

$$\tilde{\mathsf{A}}' = \mathsf{L}^{-1}\tilde{\mathsf{A}}\mathsf{L}^{-\top} \qquad \text{(Application of the preconditioner)} \tag{32g}$$

In the following subsection we present a matrix-free matrix–vector product that incorporates each of the above steps. Except for the diagonal matrix J, none of the above matrices are stored explicitly. Only the corresponding univariate matrices are stored and used in the Kronecker products, while the entries of Γ are computed on the fly.

4.4.2. Matrix-free algorithm

Let $\hat{\Gamma}_{k_1\dots k_d l_1\dots l_d} := \hat{\Gamma}(\hat{x}_{1,k_1}, \dots, \hat{x}_{d,k_d}, \hat{x}'_{1,l_1}, \dots, \hat{x}'_{d,l_d}) \in \mathbb{R}^{\tilde{n}_1\times\cdots\times\tilde{n}_d\times\tilde{n}_1\times\cdots\times\tilde{n}_d}$ denote the function values of the kernel evaluated at the tensor product grid of the Greville abscissa in $\hat{\mathcal{D}} \times \hat{\mathcal{D}}$. The proposed evaluation order of the matrix-free matrix–vector product is summarized in Algorithm 1.

Algorithm 1 Matrix-free evaluation of the matrix–vector product $\mathsf{v}' \mapsto \tilde{\mathsf{A}}'\mathsf{v}'$

Input: $v_{i_1\dots i_d} \in \mathbb{R}^{n_1\times\cdots\times n_d}$, $J_{l_1\dots l_d} \in \mathbb{R}^{\tilde{n}_1\times\cdots\times\tilde{n}_d}$, $\tilde{B}_{i_k j_k} \in \mathbb{R}^{\tilde{n}_k\times\tilde{n}_k}$ and $M_{l_k j_k} \in \mathbb{R}^{\tilde{n}_k\times n_k}$
Output: $v'_{i_1\dots i_d} \in \mathbb{R}^{n_1\times\cdots\times n_d}$

$V_{j_1\dots j_d} \leftarrow L^{-1}_{i_1 j_1} \cdots L^{-1}_{i_d j_d} v_{i_1\dots i_d}$ ▷ Preconditioning from right
$X_{k_1\dots k_d} \leftarrow M_{k_1 j_1} \cdots M_{k_d j_d} V_{j_1\dots j_d}$
$Y_{l_1\dots l_d} \leftarrow \tilde{B}^{-1}_{k_1 l_1} \cdots \tilde{B}^{-1}_{k_d l_d} X_{k_1\dots k_d}$ ▷ Use LU-factorization of $\tilde{\mathsf{B}}_k,\ k = 1, ..., d$
$Y'_{l_1\dots l_d} \leftarrow J_{l_1\dots l_d} \odot Y_{l_1\dots l_d}$
$Z'_{k_1\dots k_d} \leftarrow \hat{\Gamma}_{k_1\dots k_d l_1\dots l_d} Y'_{l_1\dots l_d}$ ▷ Evaluate in parallel without forming Γ
$Z_{k_1\dots k_d} \leftarrow J_{k_1\dots k_d} \odot Z'_{k_1\dots k_d}$
$Y_{j_1\dots j_d} \leftarrow \tilde{B}^{-1}_{j_1 k_1} \cdots \tilde{B}^{-1}_{j_d k_d} Z_{k_1\dots k_d}$ ▷ Use LU-factorization of $\tilde{\mathsf{B}}_k,\ k = 1, ..., d$
$V_{l_1\dots l_d} \leftarrow M_{j_1 l_1} \cdots M_{j_d l_d} Y_{j_1\dots j_d}$
$v'_{i_1\dots i_d} \leftarrow L^{-1}_{i_1 l_1} \cdots L^{-1}_{i_d l_d} V_{l_1\dots l_d}$ ▷ Preconditioning from left

The matrix-free matrix–vector product $\mathsf{v}' \mapsto \tilde{\mathsf{A}}'\mathsf{v}'$ is evaluated in nine separate stages. Stage one applies back-substitution of the upper triangular matrix $\mathsf{L}^{\top}$ and exploits its Kronecker structure. Stage two applies a matrix–vector product with matrix M and again exploits its Kronecker structure. Stage three applies back-substitution using the factorization of the interpolation matrix $\tilde{\mathsf{B}}$. Again, Kronecker structure is essential to reduce both the space and time complexity of the back-substitution. In stage four the coefficient vector is element-wise multiplied by the square root of the Jacobian determinant evaluated at the Greville abscissa. Here, element-wise multiplication is denoted by the $\odot$ symbol. Stage five dominates the computational cost of Algorithm 1. This stage represents a dense matrix–vector product. To perform this step without explicitly forming matrix Γ we compute its entries on

the fly, one row at a time. We compute products of the coefficient vector with several rows of Γ in parallel. Stages six to nine are equivalent to stages four to one, due to the symmetry of the operator.

Due to the iterative solution process, the matrix–vector product needs to be evaluated at each iteration. The number of iterations is dependent on the number of required eigenmodes, the conditioning of the algebraic eigenvalue problem and the efficiency of the eigensolver. In this work we use the standard implicitly restarted Lanczos method [39].

5. Computational complexity analysis

The goal of a time-complexity analysis is to obtain an estimate of the computational cost that scales linearly with time. This cost is expressed in terms of certain parameters that depend on the problem size, the dimension and the polynomial degree. For this purpose, let us introduce the following notation:

n	Number of degrees of freedom of the trial space in one component direction;
$\tilde{n}$	Number of degrees of freedom of the interpolation space in one component direction;
$N := n^d$	Total number of degrees of freedom of the trial space;
$\tilde{N} := \tilde{n}^d$	Total number of degrees of freedom of the interpolation space;
N_e	Total number of spatial elements in the trial space;
N_q	Number of quadrature points in a standard quadrature loop.
N_{iter}	Number of iterations of the matrix-free algorithm;
N_{thread}	Number of simultaneous shared memory processes in the matrix–vector product.

5.1. Standard finite element procedures

In the following we present the computational complexity of standard finite element procedures for higher-order finite elements. We use the tensor product structure of the high-dimensional space $\hat{\mathcal{D}} \times \hat{\mathcal{D}}$ to minimize the involved computations. The general setting for this analysis is (1) $\hat{\mathcal{D}} \times \hat{\mathcal{D}}$ has N_e^2 elements; and (2) we assume a quadrature rule $Q(f) := \sum_{k=1}^{N_q} w_k f(x_k)$, with $1 \leq N_q \leq (p+1)^d$, to integrate the products on every d-dimensional element $\square^d$ in $\hat{\mathcal{D}}$.

The leading term in formation and assembly is determined by the cost of forming the element matrices. Consider the following element matrix

$$
\begin{aligned}
A^e_{\mathsf{ij}} &= \int_{\square^d} B_{\mathsf{i}}(\hat{x}) \int_{\square^d} \Gamma(\hat{x}, \hat{x}') B_{\mathsf{j}}(\hat{x}') \, \mathrm{d}\hat{x}' \mathrm{d}\hat{x} \\
&\approx \sum_{k=1}^{N_q} w_k B_{\mathsf{i}}(\hat{x}_k) \sum_{l=1}^{N_q} w_l \Gamma(\hat{x}_k, \hat{x}'_l) B_{\mathsf{j}}(\hat{x}'_l) \\
&= \sum_{k=1}^{N_q} C_{\mathsf{ik}} D_{\mathsf{kj}} \quad \text{with} \quad D_{\mathsf{kj}} = \sum_{l=1}^{N_q} w_l \Gamma(\hat{x}_k, \hat{x}_l) B_{\mathsf{j}}(\hat{x}_l)
\end{aligned}
$$

with $\mathsf{i}, \mathsf{j} \in \mathcal{I}$. We see that A^e_{ij} can be formed by the matrix product of matrices $\mathsf{C} \in \mathbb{R}^{(p+1)^d \times N_q}$ and $\mathsf{D} \in \mathbb{R}^{N_q \times (p+1)^d}$. This matrix product costs $\mathcal{O}\left(N_q (p+1)^{2d}\right)$. The formation of C_{ik} is negligible. The formation of D_{kj} on the other hand is $\mathcal{O}\left(N_q^2 (p+1)^d\right)$. Since $N_q \leq (p+1)^d$ the leading term is $N_q (p+1)^{2d}$. Hence, the total cost of forming one element matrix is $\mathcal{O}\left(N_q (p+1)^{2d}\right)$. In total we have to integrate over all N_e^2 multidimensional elements of $\hat{\mathcal{D}} \times \hat{\mathcal{D}}$. With that, the total cost of forming A is $\mathcal{O}\left(N_e^2 N_q (p+1)^{2d}\right)$. Using a Gauss–Legendre quadrature rule with $(p+1)$ quadrature points in every coordinate direction gives in total $N_q = (p+1)^d$ quadrature points, and we can expect a leading cost proportional to $\mathcal{O}\left(N_e^2 (p+1)^{3d}\right)$.

5.2. Finite element procedures employing sum factorization

Estimates presented in the previous subsection hold for classical *hp* finite element procedures employing a standard quadrature loop. Next we discuss the complexity of finite element procedures that employ sum factorization

instead of a standard quadrature loop. Sum factorization significantly speeds up the element array formation by exploiting the tensorial structure of both the finite element basis and the used quadrature rules [59–63]. It is worth noting that, due to the structure of the integral operator, the sum factorization technique looks somewhat different than is standard in the *hp* finite element method.

The setting for this analysis is (1) $\hat{\mathcal{D}} \times \hat{\mathcal{D}}$ has N_e^2 rectangular elements; (2) we use a tensor product basis of polynomial degree p on every element; and (3) we use a tensor product of univariate Gauss–Legendre quadrature rules $Q(f) \coloneqq \sum_{k=1}^{p+1} \omega_k f(x_k)$ to integrate the products on every d-dimensional element $\square^d$ in $\hat{\mathcal{D}}$. Consider the element matrix

$$
\begin{aligned}
A^e_{\mathsf{ij}} &= \int_{\square^d} B_{\mathsf{i}}(x) \int_{\square^d} \Gamma(x,x') B_{\mathsf{j}}(x') \,\mathrm{d}x' \mathrm{d}x \\
&\approx \sum_{k_1=1}^{p+1} B_{i_1,p}(x_{1,k_1}) \sum_{k_2=1}^{p+1} B_{i_2,p}(x_{2,k_2}) \cdots \sum_{k_d=1}^{p+1} B_{i_d,p}(x_{d,k_d}) \\
&\quad \sum_{l_1=1}^{p+1} B_{j_1,p}(x'_{1,l_1}) \sum_{l_2=1}^{p+1} B_{j_2,p}(x'_{2,l_2}) \cdots \sum_{l_d=1}^{p+1} \Gamma(x_{1,k_1}, \ldots, x_{d,k_d}, x'_{1,l_1}, \ldots, x'_{d,l_d}) B_{j_d,p}(x'_{d,l_d})
\end{aligned}
$$

with $\mathsf{i}, \mathsf{j} \in \mathcal{I}$. Sum factorization is essentially tensor contraction. The kernel evaluated at the grid of quadrature points is a tensor $\Gamma_{k_1 \ldots k_d l_1 \ldots l_d} \in \mathbb{R}^{(p+1)\times\cdots\times(p+1)}$ and is contracted with matrices $B_{j_z,p}(x'_{z,l_z}) \in \mathbb{R}^{(p+1)\times(p+1)}$, for $z = 1, \ldots, d$, and subsequently with matrices $B_{i_z,p}(x_{z,k_z}) \in \mathbb{R}^{(p+1)\times(p+1)}$ for $z = 1, \ldots, d$. The cost of every contraction is $\mathcal{O}\left((p+1)^{2d+1}\right)$ flops. In total there are $2d$ such tensor contractions. Hence, the element matrix formation cost for A^e is $\mathcal{O}\left(2d(p+1)^{2d+1}\right)$ flops. With N_e^2 elements, the leading cost of forming A is $\mathcal{O}\left(2dN_e^2(p+1)^{2d+1}\right)$ flops.

5.3. Proposed strategy using interpolation based quadrature

In order to analyze the computational complexity of the proposed solution strategy we must address each stage of the matrix-free matrix–vector product introduced in Algorithm 1. Let us consider the complexity in one iteration of the matrix-free algorithm.

Stage 1	Has a cost $\mathcal{O}\left(dn^{d+1}\right)$.
Stage 2	Has a cost depending on the chosen projection space, For $n > \tilde{n}$ the cost is $\mathcal{O}\left(dpn^d\right)$, For $n = \tilde{n}$ the cost is $\mathcal{O}\left(dpn^d\right)$, For $n < \tilde{n}$ the cost is $\mathcal{O}\left(dp\tilde{n}^d\right)$.
Stage 3	Has a cost $\mathcal{O}\left(d\tilde{n}^{d+1}\right)$.
Stage 4	Has a cost $\mathcal{O}\left(\tilde{n}^d\right)$.
Stage 5	Has a cost $\mathcal{O}\left(\tilde{N}^2/N_{\text{thread}}\right)$.

The remaining steps 6, 7, 8 and 9 are equivalent to steps 4, 3, 2 and 1. In step 2 we assume sparse matrix algebra. In dense algebra, p can be replaced by n. In steps 1 and 3 we assume that the Cholesky and LU factorizations of the univariate matrices of size $n \times n$ and $\tilde{n} \times \tilde{n}$, respectively, have been precomputed and are available. Subsequently, the solver costs are $\mathcal{O}\left(n^2\right)$ and $\mathcal{O}\left(\tilde{n}^2\right)$ flops, respectively, for each application of the factorization. The cost of a single iteration is typically dominated by step 5, which does not depend on the polynomial degree p. Fortunately, this step is embarrassingly parallel. Hence, the time complexity of the matrix free algorithm is $\mathcal{O}\left(\tilde{N}^2 N_{\text{iter}}/N_{\text{thread}}\right)$ flops.

5.4. Storage comparison

Both matrix-free and non-matrix-free methods need to store the resulting eigenmodes. Storage of the results takes roughly $L \cdot 8N$ bytes, where L is the number of eigenmodes that need to be computed. Additionally, the standard approach that stores the dense left-hand-side system matrix $\mathsf{A} \in \mathbb{R}^{N\times N}$ requires storage of $N \times N$ floating point

Table 2
The leading terms in storage costs of a method that explicitly stores the main system matrix. Here L refers to the number of eigenmodes that need to be stored. The storage cost is dominated by storage of the system matrix.

Number of degrees of freedom	10^3	10^4	10^5	10^6
Results storage	$L \cdot 8$ kB	$L \cdot 80$ kB	$L \cdot 800$ kB	$L \cdot 8$ MB
Matrix storage	8 MB	800 MB	80 GB	8 TB

Table 3
The leading terms in storage costs of a matrix-free approach. Here L refers to the number of eigenmodes that need to be stored. The storage cost of the matrix-free method is typically dominated by the storage of the results.

Number of degrees of freedom	10^3	10^4	10^5	10^6
Results storage	$L \cdot 8$ kB	$L \cdot 80$ kB	$L \cdot 800$ kB	$L \cdot 8$ MB
Matrix-free approach	$2 \cdot 8$ kB	$2 \cdot 80$ kB	$2 \cdot 800$ kB	$2 \cdot 8$ MB

numbers. Using double precision floating point arithmetic the storage requirements are $8N^2$ bytes. The additional storage of the matrix-free approach scales linearly with problem size, with an asymptotic leading term of roughly $2{\cdot}8N$ bytes, again using double precision floating point arithmetic. If step 4 of the matrix-free algorithm is performed using shared memory parallelism then one can expect this to increase to $(1 + N_{\text{thread}}) \cdot 8N$ where N_{thread} is the number of simultaneous processes. Consequently, the storage cost of the matrix-free approach is typically governed by storage of the results and is thus optimal. Tables 2 and 3 summarize the leading terms in storage of the two alternatives.

6. Numerical results

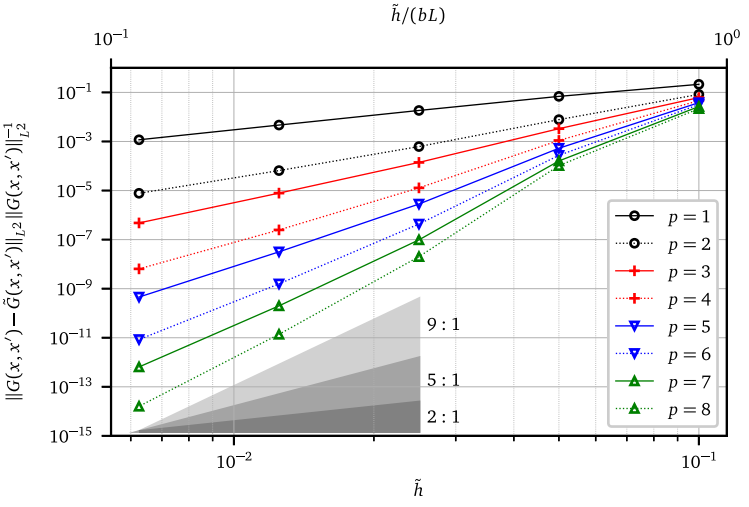
In this section we present numerical results that illustrate the accuracy, robustness and computational efficiency of the proposed matrix-free isogeometric Galerkin method. A spectral study of the accuracy is performed in the case of one dimension. This study case gives insight into the accuracy attained by interpolation based quadrature of the covariance function and its effect on approximating the eigenvalue spectra. Next, several three-dimensional C^0-conforming multipatch benchmarks illustrate the computational performance attained for a range of polynomial degrees and different refinement strategies of the interpolation as well as the solutions space. In the first two three-dimensional examples, the error in the discrete linear operator introduced by the interpolation based quadrature is showcased in the 2- and the Frobenius-matrix-norm. In the last two three-dimensional examples we study the effect of solution space refinement. All computations in the benchmark cases are performed entirely in a *single* process, without taking advantage of parallel execution, neither in the kernel evaluation itself nor in the linear algebra packages behind the implementation. The machine used in these study cases is a laptop equipped with Intel(R) Core(TM) i7-9750H CPU @ 2.60 GHz and 2×16 GB of non-ECC DDR4 2666MHz RAM. We further provide a plot showcasing the scalability of the method for one of the examples. The Python implementation in this work relies heavily on the packages Numpy [64] and Scipy [65] for the linear algebra and solver functionalities. In order to achieve high performance, crucial parts of the code base are just-in-time compiled using the LLVM-based Python-compiler Numba [66,67].

6.1. One-dimensional case study

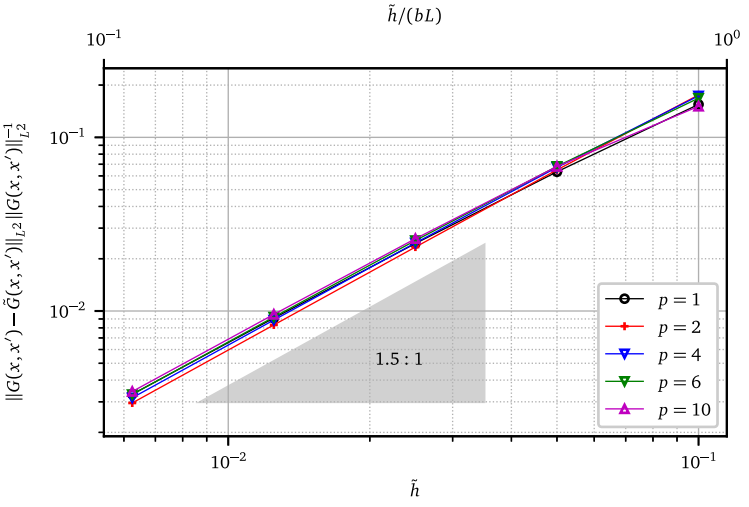
Consider a one-dimensional random field defined on the domain $\mathcal{D} = [0, 1] \subset \mathbb{R}$. We investigate

(i) the relative $L^2(\mathcal{D})$ interpolation error of the kernel, $G - \tilde{G}$, with respect to uniform h-refinement enforcing C^{p-1} continuity across element boundaries. We consider the cases where G is the Gaussian and exponential kernel with $b = 0.1$, a characteristic domain length $L = 1$ and a variance $\sigma^2 = 1$.
(ii) the normalized spectra corresponding to an exponential kernel with $b = 1$.

Remark 6.1. Normalized spectra corresponding to a Gaussian kernel cannot be reliably computed across the full range of eigenvalues because the smallest eigenvalues quickly approach zero up to machine precision.



(a) Gaussian kernel



(b) Exponential kernel

Fig. 3. Normalized L^2 interpolation error in the one-dimensional study case for multiple polynomial degrees. The error is given with respect to the mesh size of the interpolation space (bottom axis), as well as the mesh size of the interpolation space normalized by the correlation length (top axis). The convergence rates are approximately $\mathcal{O}(\tilde{h}^{p+1})$ in (a) and $\mathcal{O}(\tilde{h}^{3/2})$ in (b).

Kernel approximation

Fig. 3a shows the convergence towards the Gaussian kernel for polynomial degrees 1 through 8. It seems that the even degrees $p = \{2, 4, 6, 8\}$ perform relatively better than the preceding odd degrees $p = \{1, 3, 5, 7\}$. It is evident that higher-order interpolation of a smooth kernel leads to a higher-order convergence rate in the approximation. The smooth interpolation space is not as suitable for approximation of kernels that have low regularity. Approximation of the exponential kernel in Fig. 3b, which is C^0 along $x = x'$, shows that higher-order continuity of the basis does not lead to an increased convergence rate. This behavior is in agreement with convergence estimates for spline approximation of arbitrary smoothness [68].

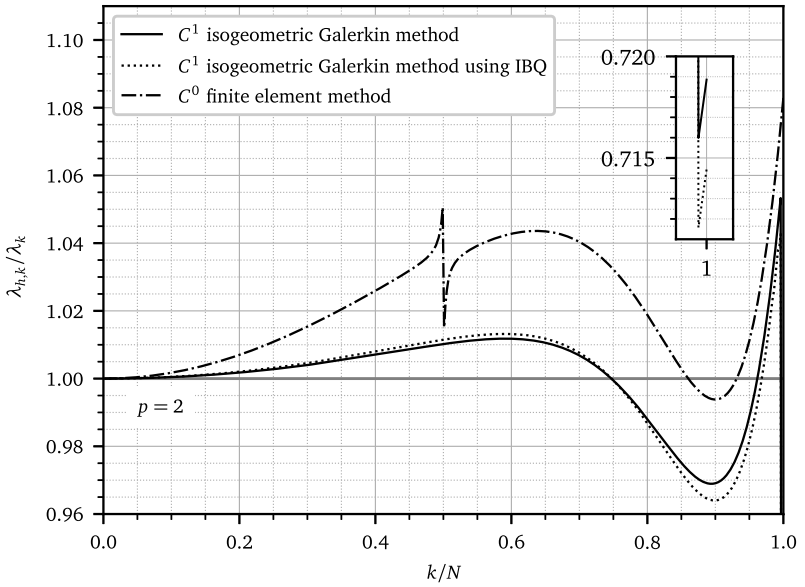


Fig. 4. Ratio of the approximated eigenvalues to the reference eigenvalues over a full spectrum of 501 eigenmodes in the one-dimensional study case with an exponential kernel and a correlation length equal to the domain length.

Spectral approximation

Although the proposed method, in its current form, is best suited for smooth kernels like the Gaussian kernel, excellent approximation of the eigenvalues corresponding to non-smooth kernels is still possible. Fig. 4 depicts the full spectrum corresponding to the exponential kernel with $b = 1$. The proposed Galerkin method using interpolation based quadrature (IBQ) is compared to the isogeometric Galerkin method proposed in [33] and a classical C^0 finite element solution in the case of polynomial degree $p = 2$. The interpolation space is set to $\tilde{h} = 0.005 \cdot bL$. The proposed method (IBQ) exhibits the same advantageous characteristics as the standard isogeometric Galerkin method [33] and exhibits no branching phenomena as in the case of the C^0-continuous finite element approximation. Due to their increased continuity across element boundaries, splines achieve a higher accuracy per degree of freedom and an increased robustness as compared with classical C^0 finite element methods. These results are in agreement with several other studies that have investigated spectral approximations corresponding to eigenvalue problems in structural mechanics [35–38].

Remark 6.2. The reference solution is computed using a standard isogeometric Galerkin method [33] with fifty thousand degrees of freedom. The first twenty eigenvalues have been validated up to machine accuracy by the analytical approach described in [7, Ch. 2.3.3, page 28-35]. The analytical computation of these eigenvalues involves solving for roots of a complex equation and is for that reason avoided beyond the first twenty eigenvalues.

6.2. Random field with exponential kernel in a three-dimensional half-open cylindrical domain

In the first three-dimensional example we investigate a random field defined in a half-open cylindrical domain as shown in Fig. 5. We consider Gaussian and exponential kernels with a correlation length bL equal to the half of the characteristic length L. The variance of the random field is $\sigma^2 = 1$. We note that the example with an exponential kernel is also studied in [33].

Example 1-1

In this example we consider the exponential kernel and since this kernel is C^0 along $x = x'$, there is no use in enforcing higher smoothness on the element boundaries of the interpolation space. Moreover, the coarse geometry is modeled by two patches with C^0 continuity between both patches, therefore, the Jacobian determinants will

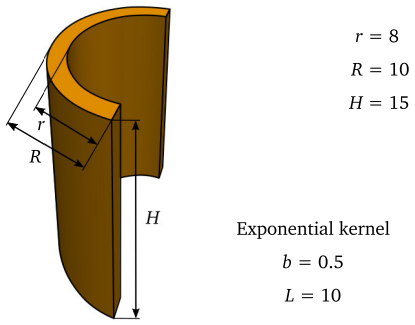


Fig. 5. Half-open cylindrical geometry in the first three-dimensional benchmark. The geometry is modeled using polynomial degrees $p = \{2, 1, 1\}$ and knot vectors $\Xi_1 = (0, 0, 0, 0.5, 0.5, 1, 1, 1)$, $\Xi_2 = (0, 0, 1, 1)$, $\Xi_3 = (0, 0, 1, 1)$. This case is also studied in [33].

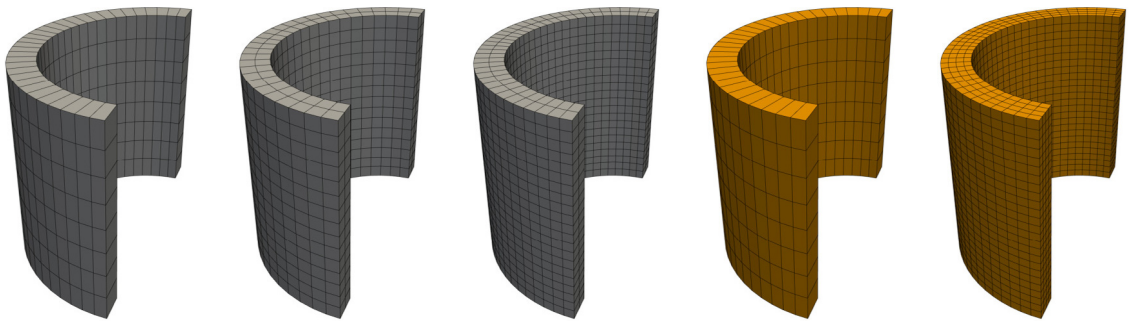


Fig. 6. Meshes of the half-open cylindrical geometry illustrating the interpolation and solution spaces in gray and orange, respectively. The gray meshes from left to right correspond to cases one through three in Tables 4 and 6, which employ the first orange mesh in the corresponding solution space. Except for the first interpolation mesh, all meshes are nearly uniform in each parametric direction (1,2,3). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

be discontinuous at the interface. Recalling Remark 4.1, in order to attain optimal accuracy of our interpolation based quadrature, we enforce a discontinuous interpolation space in the circumferential direction at that interface. Under given considerations, the chosen interpolation space employs quadratic B-splines and is C^0 on most element boundaries, except at the discontinuity, where it is C^{-1} in the circumferential direction. Since the system equations are not affected by the discontinuity, see (22) and (29), we choose the continuity of this space analogously to the example presented in [33] and use quadratic B-splines in each direction with C^1 continuity on the element boundaries. The solution space and the interpolation space meshes for each case are shown in Fig. 6. The first twenty largest eigenvalues together with information about the mesh and computational cost are tabulated in Table 4. The first nine eigenfunctions corresponding to the nine largest eigenvalues are visualized in Fig. 7 by weighting each eigenfunction by the square root of the corresponding eigenvalue.

The presolution formation and assembly time includes the formation and assembly of the univariate mass matrices, the interpolation matrices and their factorizations as well as the computation of the Jacobian determinants at the Greville abscissa. The results in Table 4 indicate that these setup costs are negligibly small compared to the total solution time of the Lanczos eigenvalue solver, which is dominated by the matrix-free evaluation of the matrix–vector product (Step 5 in Algorithm 1). The number of iterations of the Lanczos algorithm is equal in each case. The maximum resident memory increases in each case, but is negligibly small, when compared to standard methods.

The two rightmost columns summarize the results obtained by the isogeometric Galerkin method proposed in [33]. On the same mesh (Case 1) we observe a speed-up of roughly 2 orders in magnitude. This comparison might not be completely fair because a full Gaussian quadrature performed in [33] is much more accurate than our

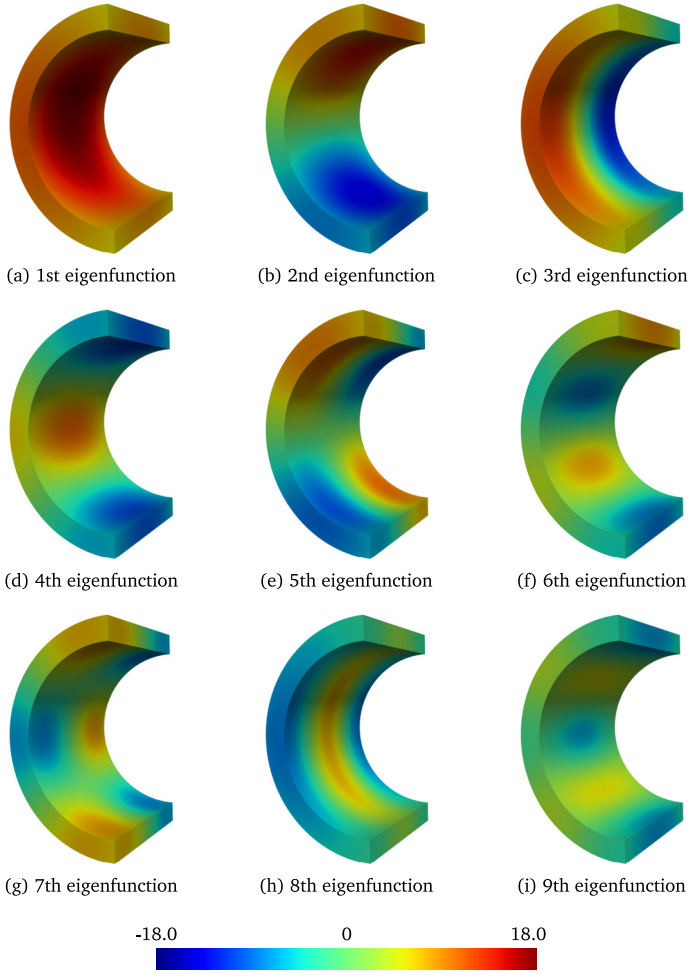


Fig. 7. First nine normalized eigenfunctions weighted by the square root of the corresponding eigenvalues in Example 1-1. Results from the third benchmark case.

interpolation based quadrature technique on the same mesh. Nonetheless, the obtained accuracy in the eigenvalues is convincing, as evidenced also in Table 5. Therein we compare the relative error of the approximated operator $\tilde{\mathsf{A}}$ in (29) in terms of the 2- and Frobenius-norm with respect to the operator A in (19) obtained by standard Gaussian quadrature with $(p+1)^3$ quadrature points. The suboptimal convergence rate for the rough exponential kernel manifests again in the operator errors.

Example 1-2

The second example employs the Gaussian covariance kernel with the same correlation length and variance as in Example 1-1. With that, we take advantage of higher smoothness across the element boundaries and rather than performing h-refinement as in Example 1-1, we set the interpolation mesh fixed (compare Case 1 in Example 1-1) and perform k-refinement for $p = 2, 4, 8$. Nonetheless, at the discontinuity in the coarse geometry model, we enforce C^{-1} in the circumferential direction.

As already discussed, it is evident, that by taking advantage of higher convergence rates the proposed method performs better for smooth kernels, which reflects in the error of the operator in Table 7. It is worth noting, that the timings versus accuracy are in favor of k-refinement.

Table 4
Enumeration of twenty largest eigenvalues corresponding to the half-open cylinder problem with the exponential kernel in Example 1-1. The numerical eigenvalues have been computed by the proposed isogeometric Galerkin method employing interpolation based quadrature for the three different cases of solution and interpolation spaces depicted in Fig. 6 as well as the standard isogeometric Galerkin reference solutions computed with two different meshes and polynomial order $p = (2, 2, 2)$. Computations executed on a *single* core.

Mode	Eigenvalue				
	Case 1	Case 2	Case 3	Galerkin 1[a]	Galerkin 2[a]
1	162.9468999	162.8100625	162.8071703	162.7991539	162.7965791
2	91.57710310	91.44102567	91.43808750	91.43063070	91.42804062
3	57.68111166	57.57587807	57.57369989	57.56702901	57.56447741
4	51.23142243	51.09932034	51.09664525	51.09017918	51.08762278
5	38.91593553	38.80608047	38.80371507	38.79740931	38.79483423
6	28.04104735	27.91172258	27.90928395	27.90386143	27.90128438
7	25.17627947	25.06401793	25.06174037	25.05611145	25.05356161
8	19.44500190	19.37694239	19.37398575	19.36893419	19.36659412
9	16.28927839	16.16360095	16.16148139	16.15700369	16.15443088
10	15.91731898	15.80248489	15.80018997	15.79530798	15.79273209
11	15.22927329	15.15403705	15.15115655	15.14622914	15.14385016
12	11.30279820	11.22038002	11.21784016	11.21328690	11.21090778
13	10.30082590	10.18526145	10.18310045	10.17896939	10.17639037
14	9.821983940	9.698825760	9.697115542	9.693564320	9.690982310
15	8.151359815	8.061279562	8.058884448	8.054783080	8.052352020
16	7.618618090	7.582986789	7.580140848	7.578085990	7.576621410
17	6.835745032	6.727654352	6.725577132	6.722378350	6.719925970
18	6.501550722	6.450711848	6.447892827	6.445576690	6.443915210
19	6.300773813	6.181552964	6.180140556	6.177345410	6.174771170
20	5.866056812	5.769454716	5.767330624	5.763786920	5.761319370
Interpolation space					
Number of elements	256	840	1596	–	–
Number of degrees of freedom	1980	8990	16770	–	–
Mesh size	2.857	1.719	1.423	–	–
Mesh size/correlation length	0.571	0.344	0.284	–	–
Formation and assembly of univariate matrices	0.314 s	0.308 s	0.301 s	–	–
Solution space					
Number of elements		… 256 …			3800
Number of degrees of freedom		… 1050 …			6642
Mesh size		… 2.857 …			1.073
Mesh size/correlation length		… 0.571 …			0.215
Formation and assembly of system matrices	–	–	–	4.86 min	17.3 h
Summary					
Number of iterations	63	63	63	63	63
Maximum resident memory [GB]	0.438	0.441	0.441	0.464	1.566
Solution time	5.031 s	64.13 s	3.367 min	0.09 s	6.21 s
Total time	5.345 s	64.44 s	3.372 min	4.86 min	17.3 h

[a]Exact kernel, NURBS trial and test space, elementwise assembly.

Table 5
Relative operator error with respect to the 2- and Frobenius-norm in Example 1-1 (exponential kernel). For the comparison the exact operator A in (19) was estimated using a Gaussian quadrature rule with $(p+1)^3$ points.

Rel. matrix norm	Case 1	Case 2	Case 3
$\|\mathsf{A}-\tilde{\mathsf{A}}\|_2\|\mathsf{A}\|_2^{-1}$	$9.95 \cdot 10^{-4}$	$7.99 \cdot 10^{-5}$	$6.12 \cdot 10^{-5}$
$\|\mathsf{A}-\tilde{\mathsf{A}}\|_F\|\mathsf{A}\|_F^{-1}$	$3.49 \cdot 10^{-3}$	$1.98 \cdot 10^{-4}$	$1.56 \cdot 10^{-4}$

Table 6
Enumeration of twenty largest eigenvalues corresponding to the half-open cylinder problem with the Gaussian kernel in Example 1-2. The numerical eigenvalues have been computed by the proposed isogeometric Galerkin method for polynomial degrees $p = 2, 4, 8$ in each parametric direction on the coarsest solution and interpolation meshes depicted in Fig. 6 as well as the standard isogeometric Galerkin reference solutions computed with two different meshes and polynomial order $p = (2, 2, 2)$. Computations executed on a *single* core.

Mode	Eigenvalue				
	$p = 2$	$p = 4$	$p = 8$	Galerkin 1[a]	Galerkin 2[a]
1	124.0032406	123.9913749	123.9916387	123.9916388	123.99141826
2	102.6956247	102.6855546	102.6857821	102.6857823	102.68558059
3	75.56585684	75.56078583	75.56096426	75.56096463	75.560806410
4	75.34465933	75.39112714	75.39245804	75.39245720	75.392381700
5	62.39810201	62.43643727	62.43754500	62.43754437	62.437470230
6	49.86332482	49.86567375	49.86580222	49.86580308	49.865713360
7	45.91399154	45.94362160	45.94444340	45.94444308	45.944382590
8	33.55923787	33.66041670	33.66423475	33.66423503	33.664535290
9	30.29707291	30.32008760	30.32063646	30.32063662	30.320605640
10	29.81774644	29.82724920	29.82734187	29.82734291	29.827325750
11	27.79271639	27.87644349	27.87960795	27.87960820	27.879851690
12	20.45053462	20.51277792	20.51510945	20.51510970	20.515286220
13	18.11733255	18.13601905	18.13635697	18.13635738	18.136361050
14	16.33318523	16.34652184	16.34660003	16.3466016	16.346634880
15	13.49460845	13.53722675	13.53876834	13.53876867	13.538889150
16	11.29075514	11.39146819	11.39829646	11.39830777	11.399235970
17	9.924081589	9.939261559	9.939463418	9.939464250	9.9394922100
18	9.350652024	9.434037081	9.439692866	9.439702240	9.4404691900
19	8.282851361	8.296245384	8.296321088	8.296322560	8.2963738600
20	8.069634638	8.097318365	8.098244763	8.098245110	8.0983270800
Interpolation space					
Number of elements	256	256	256	–	–
Number of degrees of freedom	1080	2400	6912	–	–
Mesh size	2.857	2.857	2.857	–	–
Mesh size/correlation length	0.571	0.571	0.571	–	–
Formation and assembly of univariate matrices	0.299 s	0.299 s	0.301 s	–	–
Solution space					
Number of elements		… 256 …			3800
Number of degrees of freedom		… 1050 …			6642
Mesh size		… 2.857 …			1.073
Mesh size/correlation length		… 0.571 …			0.215
Formation and assembly of system matrices	–	–	–	5.01 min	16.97 h
Summary					
Number of iterations	52	52	52	52	52
Maximum resident memory [GB]	0.437	0.436	0.437	0.464	1.615
Solution time	0.817 s	3.412 s	27.95 s	0.10 s	5.61 s
Total time	1.116 s	3.711 s	28.25 s	5.02 min	16.97 h

[a]Exact kernel, NURBS trial and test space, elementwise assembly.

Table 7
Relative operator error with respect to the 2- and Frobenius-norm in Example 1-2 (Gaussian kernel). For the comparison the exact operator A in (19) was estimated using a Gaussian quadrature rule with $(p+1)^3$ points.

Rel. matrix norm	Case 1	Case 2	Case 3
$\|\mathsf{A} - \tilde{\mathsf{A}}\|_2 \|\mathsf{A}\|_2^{-1}$	$9.05 \cdot 10^{-4}$	$5.30 \cdot 10^{-5}$	$3.67 \cdot 10^{-7}$
$\|\mathsf{A} - \tilde{\mathsf{A}}\|_F \|\mathsf{A}\|_F^{-1}$	$1.27 \cdot 10^{-3}$	$6.71 \cdot 10^{-5}$	$3.75 \cdot 10^{-7}$

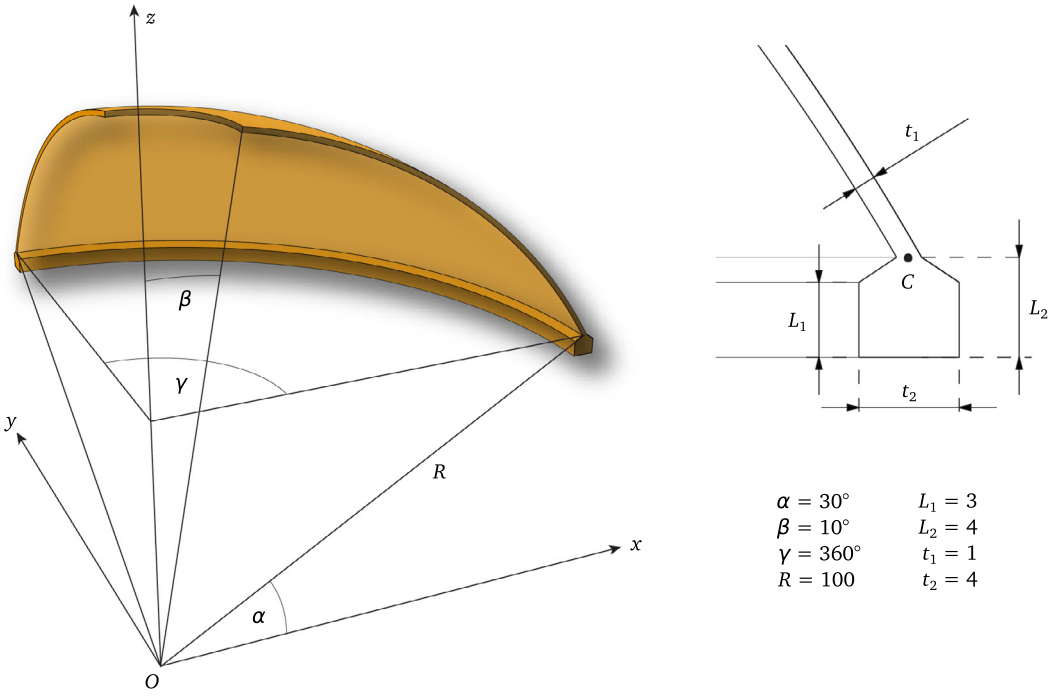


Fig. 8. Hemispherical shell with a stiffener. The geometry is modeled as a single NURBS patch using polynomial degrees $p = \{2, 2, 2\}$ and knot vectors $\Xi_1 = (0, 0, 0, 1, 1, 2, 2, 3, 3, 3)$, $\Xi_2 = (0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 4)$, $\Xi_3 = (0, 0, 0, 1, 1, 1)$.

6.3. Random field with Gaussian kernel in a three-dimensional hemispherical shell

We consider the hemispherical shell with stiffener depicted in Fig. 8. This three dimensional multipatch shell structure is similar to the model published in [69] but has a slightly different stiffener profile. We use the Gaussian covariance function with a correlation length $bL = 0.5L$, where the characteristic domain length, $L \approx 176$, is the diameter of the stiffener ring.

We study two examples across a range of polynomial degrees $p = \{2, 6, 16\}$. The two examples differ in the following way: In both studies we use interpolation and solution meshes obtained by p-refinement of the geometrical model, followed by uniform h-refinement. The continuity of these spaces is thus C^0 at knots that are present in the initial coarse geometrical model and C^{p-1} at new knots introduced by h-refinement. Again, where the solution space is C^0, we enforce C^{-1} in the interpolation space in order to achieve optimal accuracy per degree of freedom.

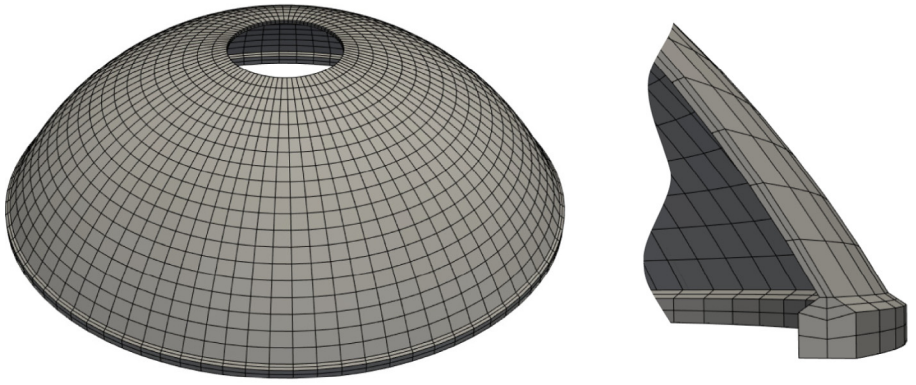
Example 2-1	The solution mesh is the same as the interpolation mesh;
Example 2-2	The solution mesh is twice as fine as the interpolation mesh in every component direction.

Example 2-1

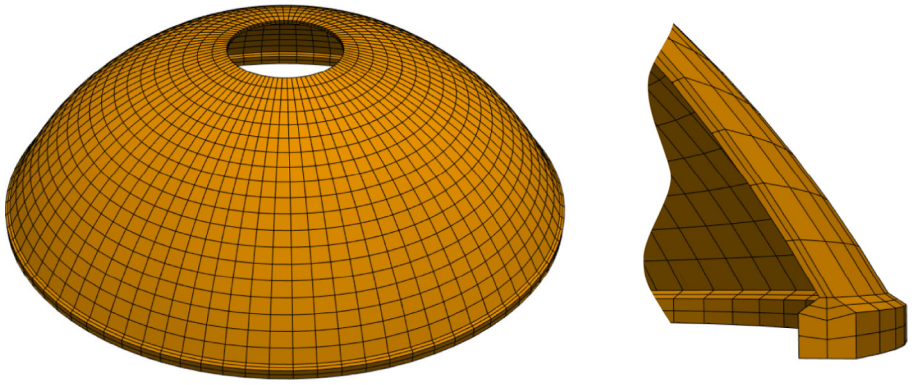
Figs. 9a and 9b depict the interpolation and solution meshes, respectively, that are used in this benchmark case. As described in this benchmark case the solution and interpolation space are identical. The numerical results are summarized in Table 8. As in the previous three-dimensional benchmarks the presolution setup costs hardly contribute to the total cost of the method. Again, the main computational cost lies in the matrix-free matrix–vector product that is evaluated in each iteration of the Lanczos eigenvalue solver. Due to the increased number of degrees of freedom for higher polynomial degrees the associated computational cost increases. Interestingly, the number of iterations is reduced in the case $p = 16$ as compared to the lower polynomial degrees.

Example 2-2

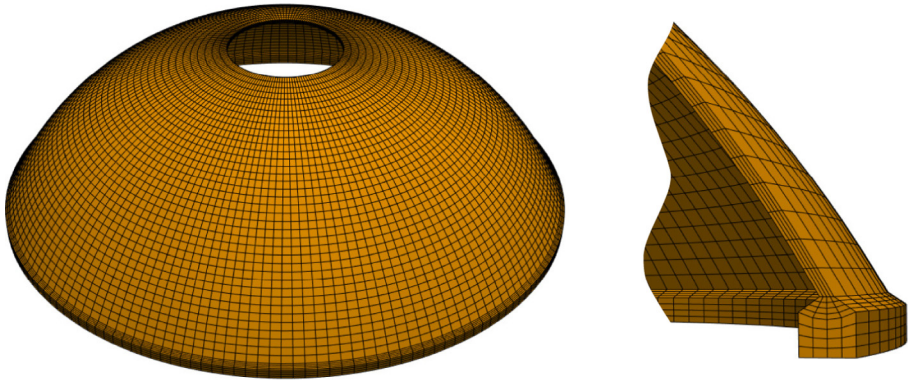
In the second benchmark the element size of the solution mesh is halved, see Fig. 9c. The interpolation mesh is kept the same as in the first benchmark. The obtained results are presented in Fig. 10 and Table 9. By comparing



(a) Interpolation space mesh in Example 2-1 and Example 2-2 with $23 \times 84 \times 2$ elements in parametric directions $(1, 2, 3)$



(b) Solution space mesh in Example 2-1 with $23 \times 84 \times 2$ elements in parametric directions $(1, 2, 3)$



(c) Solution space mesh in Example 2-2 with $46 \times 168 \times 4$ elements in parametric directions $(1, 2, 3)$

Fig. 9. The solution space and interpolation space meshes and the cross-sections used in Example 2-1 and Example 2-2.

Table 9 and Table 8 it may be observed that the dimension of the solution space does not significantly affect the total solver costs. Indeed, the number of degrees of freedom are more than doubled, yet the timings stay more or less the same, independent of polynomial degree. The increased dimension of the solution space mesh is reflected

Table 8
Enumeration of twenty largest eigenvalues corresponding to the hemispherical shell with stiffener problem with Gaussian kernel in Example 2-1. The numerical eigenvalues have been computed by the proposed isogeometric Galerkin method employing interpolation based quadrature for the three different cases using solution and interpolation spaces depicted in Figs. 9a and 9b as well as standard isogeometric Galerkin reference solution computed using mesh depicted in Fig. 9b and polynomial order $p = (2, 2, 2)$. Computations executed on a *single* core.

Mode	Eigenvalue			
	$p = 2$	$p = 6$	$p = 16$	Galerkin 1[a]
1	14 474.18249	14 475.59075	14 476.12924	14 476.26164
2	6530.062180	6531.224499	6531.666465	6531.775099
3	6530.062180	6531.224499	6531.666444	6531.775099
4	2091.112735	2091.638679	2091.836166	2091.884692
5	2091.111050	2091.638679	2091.836147	2091.884692
6	1971.193088	1971.617475	1971.784884	1971.826015
7	552.5819743	552.6961143	552.7418006	552.7530118
8	552.5819743	552.6961143	552.7417949	552.7530118
9	522.0061723	522.1758813	522.2364134	522.2512911
10	522.0061723	522.1758813	522.2364108	522.2512911
11	113.1117043	113.1328509	113.1414885	113.1435938
12	113.1116131	113.1328509	113.1414662	113.1435938
13	106.0162787	106.0601883	106.0743281	106.0778059
14	106.0161093	106.0601883	106.0743158	106.0778057
15	101.1533173	101.1674081	101.1733615	101.1748288
16	21.40721769	21.40924526	21.41014572	21.41036678
17	21.40721769	21.40924526	21.41014233	21.41036678
18	19.11433927	19.11772882	19.11904088	19.11936197
19	19.11433927	19.11772882	19.11903300	19.11936197
20	18.06993704	18.07977268	18.08244326	18.08309832
Interpolation space				
Number of elements	3864	3864	3864	–
Number of degrees of freedom	10 672	35 424	189 144	–
Mesh size	7.992	7.992	7.992	–
Mesh size/correlation length	0.091	0.091	0.091	–
Formation and assembly of univariate matrices	0.354 s	0.427 s	3.530 s	–
Solution space				
Number of elements	3864	3864	3864	3864
Number of degrees of freedom	9612	32 760	180 090	9612
Mesh size	7.992	7.992	7.992	7.992
Mesh size/correlation length	0.091	0.091	0.091	0.091
Formation and assembly of system matrices	–	–	–	17.29 h
Summary				
Number of iterations	52	52	41	52
Maximum resident memory [GB]	0.209	0.275	1.282	2.709
Solution time	70.28 s	12.33 min	5.00 h	13.16 s
Total time	70.63 s	12.33 min	5.00 h	17.29 h

[a]Exact kernel, NURBS trial and test space, elementwise assembly.

in the maximum resident memory, which has increased as compared to the results in Table 8. As witnessed in the previous benchmark, the higher order computations required fewer iterations than the lower order ones.

Remark 6.3. Note that the flexibility in mesh size of the interpolation versus the trial space mesh provides a mechanism by which the error due to quadrature versus the error due to discretization can be effectively controlled.

Remark 6.4. A relevant question in random field discretization is what mesh size is necessary to attain acceptable approximations. The mesh size should clearly depend on the correlation length bL. A rule of thumb, proposed

Table 9
Enumeration of twenty largest eigenvalues corresponding to the hemispherical shell with stiffener problem with Gaussian kernel in Example 2-2. The numerical eigenvalues have been computed by the proposed isogeometric Galerkin method employing interpolation based quadrature for the three different cases using solution and interpolation spaces depicted in Figs. 9a and 9b as well as standard isogeometric Galerkin reference solution computed using mesh depicted in Fig. 9b and polynomial order $p = (2, 2, 2)$. Computations executed on a *single* core.

Mode	Eigenvalue			
	$p = 2$	$p = 6$	$p = 16$	Galerkin 1[a]
1	14 475.49870	14 475.83951	14 476.16143	14 476.26164
2	6531.142590	6531.428669	6531.692867	6531.775099
3	6531.142590	6531.428669	6531.692860	6531.775099
4	2091.595539	2091.729909	2091.847949	2091.884692
5	2091.593853	2091.729909	2091.847938	2091.884692
6	1971.603041	1971.694851	1971.794886	1971.826015
7	552.6940460	552.7172396	552.7445251	552.7530118
8	552.6940460	552.7172396	552.7445226	552.7530118
9	522.1541843	522.2038475	522.2400300	522.2512911
10	522.1541843	522.2038475	522.2400296	522.2512911
11	113.1329195	113.1368433	113.1419923	113.1435938
12	113.1328283	113.1368433	113.1419915	113.1435938
13	106.0508639	106.0667227	106.0751804	106.0778059
14	106.0506944	106.0667227	106.0751771	106.0778057
15	101.1680420	101.1701707	101.1737246	101.1748288
16	21.40946576	21.40966457	21.41020520	21.41036678
17	21.40946576	21.40966457	21.41020194	21.41036678
18	19.11757482	19.11833648	19.11912297	19.11936197
19	19.11757482	19.11833648	19.11911862	19.11936197
20	18.07646765	18.08100654	18.08261032	18.08309832
Interpolation space				
Number of elements	3864	3864	3864	–
Number of degrees of freedom	10 672	35 424	189 144	–
Mesh size	7.992	7.992	7.992	–
Mesh size/correlation length	0.091	0.091	0.091	–
Formation and assembly of univariate matrices	0.367 s	1.229 s	23.21 s	–
Solution space				
Number of elements	30 912	30 912	30 912	3864
Number of degrees of freedom	51 900	117 180	421 360	9612
Mesh size	4.019	4.019	4.019	7.992
Mesh size/correlation length	0.046	0.046	0.046	0.091
Formation and assembly of system matrices	–	–	–	17.29 h
Summary				
Number of iterations	52	52	41	52
Maximum resident memory [GB]	0.261	0.741	7.346	2.709
Solution time	70.05 s	12.433 min	4.850 h	13.16 s
Total time	70.42 s	12.433 min	4.858 h	17.29 h

[a]Exact kernel, NURBS trial and test space, elementwise assembly.

in [70], is that the element size is approximately in the range from a half to a quarter of the given correlation length. Similar rules have been established by other authors, see [8] and references therein. Especially, in a three-dimensional problem this may lead to a large number of degrees of freedom. Engineering models of practical interest are generally more complex than the models shown in this paper and may require millions of degrees of freedom.

Parallel execution

In order to provide comparable results in the benchmarks, all the computations have been performed in a sequential manner in a single process. The proposed method is well suited for parallel execution, as discussed in Section 5.3. Fig. 11 shows nearly optimal scaling with the number of cores for Example 2-2 in the case of $p = 6$.

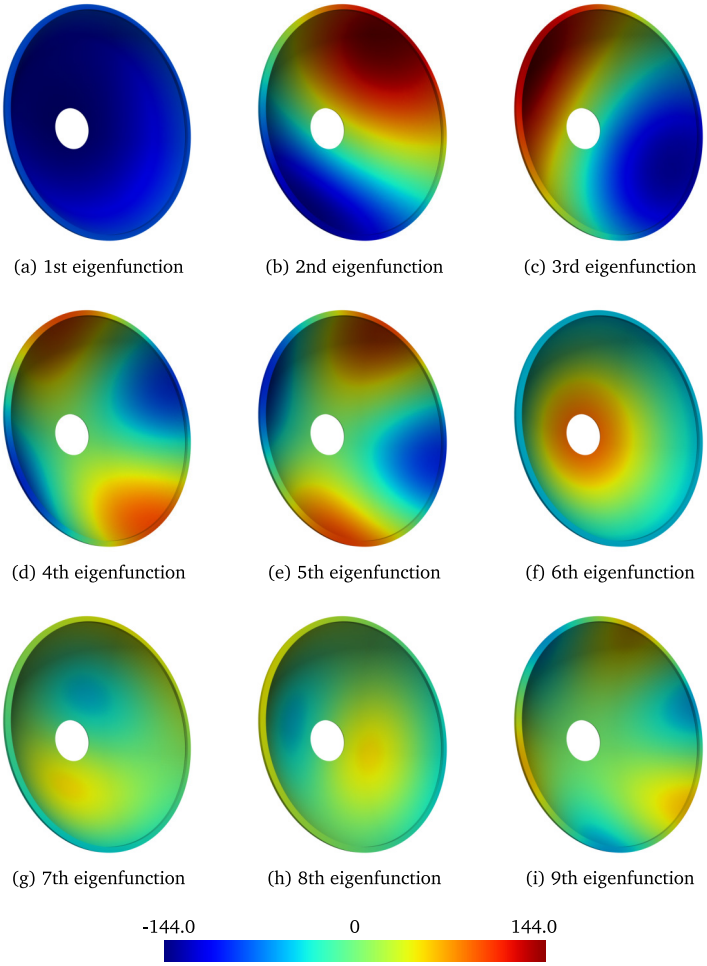


Fig. 10. First nine normalized eigenfunctions weighted by the square root of the corresponding eigenvalues in Example 2-2.

Note, that *only* the kernel evaluation and matrix–vector product in Step 5 of Algorithm 1 have been performed in parallel using shared-memory parallelism. All the other steps, as well as the Lanczos eigensolver are still run sequentially. In this particular case 99.84% of the Algorithm 1 execution time was spent in the parallel execution mode.

7. Conclusion

This paper presented an efficient matrix-free Galerkin method for the Karhunen–Loève series expansion (KLE) of random fields. The KLE requires the solution of a generalized eigenvalue problem corresponding to the homogeneous Fredholm integral eigenvalue problem of the second kind, and is computationally challenging for several reasons. Firstly, the Galerkin method requires numerical integration over a $2d$ dimensional domain, where d, in this work, denotes the spatial dimension. Consequently, classical formation and assembly procedures have a time complexity that scales $\mathcal{O}\left(N_e^2 \cdot (p+1)^{3d}\right)$ with increasing polynomial degree p and number of elements N_e. Secondly, the main system matrix is dense and requires $\mathcal{O}\left(N^2\right)$ bytes of storage, where N is the global number of degrees of freedom. This means that a discretization involving a hundred thousand degrees of freedom requires at least 80 GB of RAM to store the main system matrix in double precision. Hence, the computational complexity as well as memory requirements of standard solution techniques become quickly computationally intractable with increasing polynomial degree, problem size and spatial dimension.

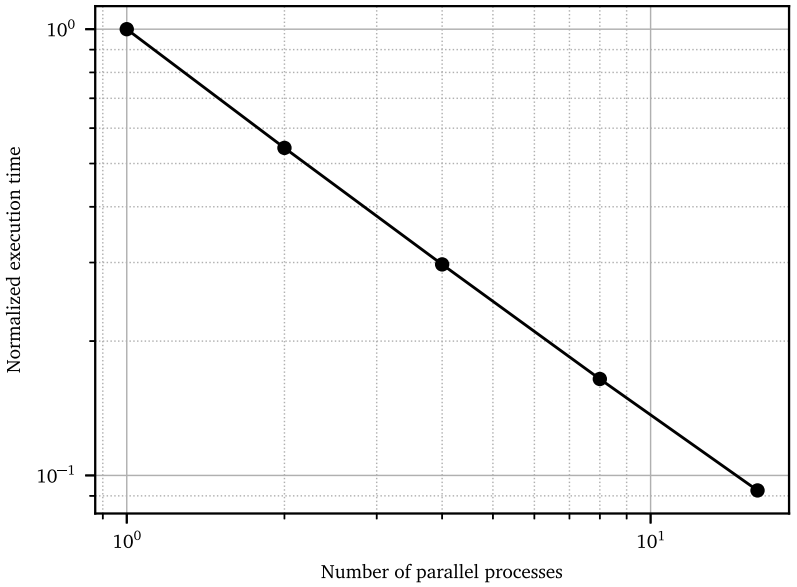


Fig. 11. Scaling of the execution time in Example 2-2 in the case of $p = 6$ and $1, 2, 8$ and 16 parallel processes. Timings normalized with respect to the execution time in a single process.

We proposed an efficient solution methodology that significantly ameliorates the aforementioned computational challenges. Our approach is based on the following key ingredients:

1. A trial space of rational spline functions, whose Gramian or mass matrix has a Kronecker structure independent of the geometric mapping;
2. An inexpensive reformulation of the generalized algebraic eigenvalue problem into a standard algebraic eigenvalue problem;
3. A degenerate kernel approximation of the covariance function using smooth tensor product splines;
4. Formulation of an efficient matrix-free and parallel matrix–vector product for iterative eigenvalue solvers, which utilizes the Kronecker structure of the system matrices.

In Step 2 the reformulation to a standard eigenvalue problem significantly reduces the computational cost while improving conditioning. This can be done efficiently due to the Kronecker structure of the mass matrix, which is a result of the particular choice of the trial space, see Step 1. In Step 3 the degenerate kernel approximation enables us to evaluate the resulting integrals exactly with a minimal number of evaluation points. Both steps involve matrices that are endowed with a Kronecker structure and can be performed matrix-free in $\mathcal{O}\left(N \cdot N^{1/d}\right)$ time. The leading cost of the method is due to the Lanczos eigenvalue algorithm, which involves dense matrix–vector multiplications. As noted in Step 4, we perform this step matrix-free, by computing the necessary components on the fly and in parallel in approximately $\mathcal{O}\left(N^2 N_{\text{iter}}/N_{\text{thread}}\right)$ time. Here N_{iter} denotes the number of iterations of the eigensolver and N_{thread} is the number of simultaneous processes. Several three dimensional benchmark problems involving non-trivial geometrical mappings have illustrated exceptional efficiency and effectiveness of the proposed solution methodology. In particular, we showed that the proposed methodology scales favorably with polynomial degree and works particularly well for smooth covariance functions, such as the Gaussian kernel. The Python implementation used to generate these results and the associated reference benchmarks has been provided as open-source software and is available for download at https://github.com/m1ka05/tensiga.

In a follow-up study we plan to extensively study the accuracy of the proposed solution methodology. There are two sources of error: (1) a quadrature error due to approximation of the covariance function; and (2) a discretization error due to the finite dimensional representation of the eigenmodes. We will perform a priori as well as a posteriori error analysis and formulate criteria for bounding the error due to quadrature by the discretization error. Within the same context of accuracy and robustness it is interesting to extend the spectral analysis results in [37] to generalized

eigenvalue problems corresponding to Fredholm integral equations for different covariance functions as well as polynomial order of the approximation.

We also plan to further improve the efficiency of the proposed method where possible. The proposed matrix-free algorithm lends itself for acceleration on graphics processing units (GPUs). Furthermore, exploiting particular structure (such as sparsity or symmetry) of the covariance function may lead to improved solver cost. For example, the hierarchical matrix method proposed in [13] performs the matrix–vector products in $\mathcal{O}(N \log N)$ time, by exploiting certain structure of the covariance function.

There are several other interesting avenues for future research. Some or all of the techniques proposed here could be applied to linear as well as non-linear Fredholm integral differential equations of the first as well as the second kind. While the proposed method is designed for smooth kernels it would be interesting to develop similar methods that are tailored towards continuous kernels, such as the exponential kernel, or even singular kernels, which are typical in boundary integral equations, see [56–58] for similar ideas making use of quasi-interpolation.

Finally, we would like to mention that similar techniques can be applied in the context of the collocation method. The computational cost of such a method would be similar to that of the proposed Galerkin method .

Declaration of competing interest

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

Acknowledgments

M.L. Mika, R.R. Hiemstra and D. Schillinger gratefully acknowledge funding from the German Research Foundation through the DFG Emmy Noether Award SCH 1249/2-1. T.J.R. Hughes and R.R. Hiemstra were partially supported by the National Science Foundation Industry/University Cooperative Research Center (IUCRC) for Efficient Vehicles and Sustainable Transportation Systems (EV-STS), and the United States Army CCDC Ground Vehicle Systems Center (TARDEC/NSF Project # 1650483 AMD 2). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors thank Mona Dannert and Udo Nackenhorst for very helpful discussions and comments.

References

[1] R.E. Melchers, A.T. Beck (Eds.), Structural Reliability Analysis and Prediction, John Wiley & Sons Ltd, Chichester, UK, 2017.
[2] B. Sudret, A. Kuyreghian, Stochastic Finite Element Methods and Reliability: A State-of-the-Art Report, Berkeley, Department of Civil and Environmental Engineering, University of California, 2000.
[3] M. Eiermann, O.G. Ernst, E. Ullmann, Computational aspects of the stochastic finite element method, Comput. Vis. Sci. 10 (1) (2007) 3–15.
[4] A. Keese, A Review of Recent Developments in the Numerical Solution of Stochastic Partial Differential Equations (Stochastic Finite Elements), Institut für Wissenschaftliches Rechnen, Braunschweig, 2003.
[5] K. Karhunen, Über lineare Methoden in der Wahrscheinlichkeitsrechnung, in: Suomalaisen Tiedeakatemian Toimituksia, Zugl.: Helsinki, Univ., Diss., 1947, Helsinki, 1947.
[6] M. Loève, Functions Aleatoires Du Second Ordre. Processus Stochastique Et Mouvement Brownien, Gauthier-Villars, Paris, 1948, pp. 366–420.
[7] R.G. Ghanem, P.D. Spanos, Stochastic Finite Elements: A Spectral Approach, Springer New York, New York, NY, 1991.
[8] G. Stefanou, The stochastic finite element method: Past, present and future, Comput. Methods Appl. Mech. Engrg. 198 (2009) 1031–1051.
[9] W. Betz, I. Papaioannou, D. Straub, Numerical methods for the discretization of random fields by means of the Karhunen–Loève expansion, Comput. Methods Appl. Mech. Engrg. 271 (2014) 109–129.
[10] R. Kress, Linear Integral Equations, third ed., in: Applied mathematical sciences, No. volume 82, Springer, New York, 2014.
[11] D.L. Allaix, V.I. Carbone, Karhunen-Loève decomposition of random fields based on a hierarchical matrix approach, Internat. J. Numer. Methods Engrg. 94 (11) (2013) 1015–1036.
[12] W. Hackbusch, B.N. Khoromskij, E.E. Tyrtyshnikov, Hierarchical Kronecker tensor-product approximations, J. Numer. Math. 13 (2) (2005).
[13] B.N. Khoromskij, A. Litvinenko, H.G. Matthies, Application of hierarchical matrices for computing the Karhunen–Loève expansion, Computing 84 (1–2) (2009) 49–67.
[14] C. Schwab, R.A. Todor, Karhunen–Loève approximation of random fields by generalized fast multipole methods, J. Comput. Phys. 217 (1) (2006) 100–122.

[15] K. Phoon, S. Huang, S. Quek, Implementation of Karhunen–Loeve expansion for simulation using a wavelet-Galerkin scheme, Probab. Eng. Mech. 17 (3) (2002) 293–303.
[16] W. Dahmen, H. Harbrecht, R. Schneider, Compression techniques for Boundary Integral Equations—Asymptotically optimal complexity estimates, SIAM J. Numer. Anal. 43 (6) (2006) 2251–2271.
[17] H. Harbrecht, M. Peters, M. Siebenmorgen, Efficient approximation of random fields for numerical applications, Numer. Linear Algebra Appl. 22 (4) (2015) 596–617.
[18] T. Hughes, J. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (39–41) (2005) 4135–4195.
[19] J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, Isogeometric Analysis: Toward Integration of CAD and FEA, Wiley, Chichester, West Sussex, U.K. ; Hoboken, NJ, 2009, OCLC: ocn335682757.
[20] K. Li, W. Gao, D. Wu, C. Song, T. Chen, Spectral stochastic isogeometric analysis of linear elasticity, Comput. Methods Appl. Mech. Engrg. 332 (2018) 157–190.
[21] R. Jahanbin, S. Rahman, Stochastic isogeometric analysis in linear elasticity, Comput. Methods Appl. Mech. Engrg. 364 (2020) 112928.
[22] W. Wang, G. Chen, D. Yang, Z. Kang, Stochastic isogeometric analysis method for plate structures with random uncertainty, Comput. Aided Geom. Design 74 (2019) 101772.
[23] K. Li, D. Wu, W. Gao, C. Song, Spectral stochastic isogeometric analysis of free vibration, Comput. Methods Appl. Mech. Engrg. 350 (2019) 1–27.
[24] C. Ding, X. Hu, X. Cui, G. Li, Y. Cai, K.K. Tamma, Isogeometric generalized n th order perturbation-based stochastic method for exact geometric modeling of (composite) structures: Static and dynamic analysis with random material parameters, Comput. Methods Appl. Mech. Engrg. 346 (2019) 1002–1024.
[25] T.D. Hien, H.-C. Noh, Stochastic isogeometric analysis of free vibration of functionally graded plates considering material randomness, Comput. Methods Appl. Mech. Engrg. 318 (2017) 845–863.
[26] K. Li, D. Wu, W. Gao, Spectral stochastic isogeometric analysis for static response of FGM plate with material uncertainty, Thin-Walled Struct. 132 (2018) 504–521.
[27] K. Li, D. Wu, W. Gao, Spectral stochastic isogeometric analysis for linear stability analysis of plate, Comput. Methods Appl. Mech. Engrg. 352 (2019) 1–31.
[28] H. Zhang, T. Shibutani, Development of stochastic isogeometric analysis (SIGA) method for uncertainty in shape, Internat. J. Numer. Methods Engrg. (2018) nme.6008.
[29] C. Eckert, M. Beer, P.D. Spanos, A polynomial chaos method for arbitrary random inputs using B-splines, Probab. Eng. Mech. 60 (2020) 103051.
[30] S. Rahman, A spline Chaos expansion, SIAM/ASA J. Uncertain. Quantif. 8 (1) (2020) 27–57.
[31] D. Xiu, G.E. Karniadakis, The Wiener–Askey Polynomial Chaos for stochastic Differential Equations, SIAM J. Sci. Comput. 24 (2) (2002) 619–644.
[32] D.W. Arthur, The solution of Fredholm Integral Equations using spline functions, IMA J. Appl. Math. 11 (2) (1973) 121–129.
[33] S. Rahman, A Galerkin isogeometric method for Karhunen–Loève approximation of random fields, Comput. Methods Appl. Mech. Engrg. 338 (2018) 533–561.
[34] R. Jahanbin, S. Rahman, An isogeometric collocation method for efficient random field discretization, Internat. J. Numer. Methods Engrg. 117 (3) (2019) 344–369.
[35] J. Cottrell, T. Hughes, A. Reali, Studies of refinement and continuity in isogeometric structural analysis, Comput. Methods Appl. Mech. Engrg. 196 (41–44) (2007) 4160–4183.
[36] T. 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, Comput. Methods Appl. Mech. Engrg. 197 (49–50) (2008) 4104–4124.
[37] T.J. Hughes, J.A. Evans, A. Reali, Finite element and NURBS approximations of eigenvalue, boundary-value, and initial-value problems, Comput. Methods Appl. Mech. Engrg. 272 (2014) 290–320.
[38] V. Puzyrev, Q. Deng, V. Calo, Spectral approximation properties of isogeometric analysis with variable continuity, Comput. Methods Appl. Mech. Engrg. 334 (2018) 22–39.
[39] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., in: Johns Hopkins studies in the mathematical sciences, Johns Hopkins University Press, Baltimore, 1996.
[40] Y. Saad, Numerical Methods for Large Eigenvalue Problems, rev. ed., in: Classics in applied mathematics, No. 66, Society for Industrial and Applied Mathematics, Philadelphia, 2011.
[41] A. Mantzaflaris, B. Jüttler, Integration by interpolation and look-up for Galerkin-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 284 (2015) 373–400.
[42] M. Pan, B. Jüttler, A. Giust, Fast formation of isogeometric Galerkin matrices via integration by interpolation and look-up, Comput. Methods Appl. Mech. Engrg. 366 (2020) 113005.
[43] F. Calabró, G. Sangalli, M. Tani, Fast formation of isogeometric Galerkin matrices by weighted quadrature, Comput. Methods Appl. Mech. Engrg. 316 (2017) 606–622.
[44] R.R. Hiemstra, G. Sangalli, M. Tani, F. Calabró, T.J. Hughes, Fast formation and assembly of finite element matrices with application to isogeometric linear elasticity, Comput. Methods Appl. Mech. Engrg. 355 (2019) 234–260.
[45] G. Sangalli, M. Tani, Matrix-free weighted quadrature for a computationally efficient isogeometric k -method, Comput. Methods Appl. Mech. Engrg. 338 (2018) 117–133.
[46] A. Keese, Numerical solution of systems with stochastic uncertainties: A general purpose framework for stochastic finite elements, 2004.

[47] L. Piegl, W. Tiller, The NURBS Book, in: Monographs in Visual Communications, Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.
[48] F. Kjolstad, S. Kamil, S. Chou, D. Lugato, S. Amarasinghe, The tensor algebra compiler, Proc. ACM Program. Lang. 1 (OOPSLA) (2017) 1–29.
[49] S. Lang, S. Lang, Real and Functional Analysis, third ed., in: Graduate texts in mathematics, No. 142, Springer-Verlag, New York, 2012.
[50] R. Courant, D. Hilbert, Methods of Mathematical Physics, first ed., Wiley, 1989.
[51] J. Mercer, Functions of positive and negative type, and their connection with the Theory of Integral Equations, Phil. Trans. R. Soc. A 209 (441–458) (1909) 415–446.
[52] K.E. Atkinson, The Numerical Solution of Integral Equations of the Second Kind, first ed., Cambridge University Press, 1997.
[53] A.H. Vermeulen, R.H. Bartels, G.R. Heppler, Integrating Products of B-splines, SIAM J. Sci. Stat. Comput. 13 (4) (1992) 1025–1038.
[54] R.R. Hiemstra, F. Calabró, D. Schillinger, T.J. Hughes, Optimal and reduced quadrature rules for tensor product and hierarchically refined splines in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 316 (2017) 966–1004, Special Issue on Isogeometric Analysis: Progress and Challenges.
[55] C. De Boor, A Practical Guide to Splines, in: Applied mathematical sciences, No. 27, Springer-Verlag, New York, 1978.
[56] F. Calabró, A. Falini, M.L. Sampoli, A. Sestini, Efficient quadrature rules based on spline quasi-interpolation for application to IGA-BEMs, J. Comput. Appl. Math. 338 (2018) 153–167.
[57] A. Falini, C. Giannelli, T. Kanduč, M.L. Sampoli, A. Sestini, An adaptive IgA-BEM with hierarchical B-splines based on quasi-interpolation quadrature schemes, Internat. J. Numer. Methods Engrg. 117 (10) (2019) 1038–1058.
[58] A. Falini, T. Kanduč, A study on spline Quasi-interpolation based quadrature rules for the isogeometric Galerkin BEM, in: C. Giannelli, H. Speleers (Eds.), Advanced Methods for Geometric Modeling and Numerical Simulation, Vol. 35, Springer International Publishing, Cham, 2019, pp. 99–125, Series Title: Springer INdAM Series.
[59] P. Antolin, A. Buffa, F. Calabró, M. Martinelli, G. Sangalli, Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization, Comput. Methods Appl. Mech. Engrg. 285 (2015) 817–828.
[60] A. Bressan, S. Takacs, Sum factorization techniques in Isogeometric Analysis, Comput. Methods Appl. Mech. Engrg. 352 (2019) 437–460.
[61] S.A. Orszag, Spectral methods for problems in complex geometries, J. Comput. Phys. 37 (1) (1980) 70–92.
[62] E. Tino, J.M. Melenk, Fast algorithms for setting up the stiffness matrix in hp-FEM: a comparison, 2005.
[63] P.E.J. Vos, S.J. Sherwin, R.M. Kirby, From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low- and high-order discretisations, J. Comput. Phys. 229 (13) (2010) 5161–5181.
[64] S. van der Walt, S. Colbert, G. Varoquaux, The NumPy array: A structure for efficient numerical computation, Comput. Sci. Eng. 13 (2) (2011) 22–30.
[65] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K.J. Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C.J. Carey, I. Polat, Y. Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: fundamental algorithms for scientific computing in Python, Nat. Methods 17 (3) (2020) 261–272.
[66] S.K. Lam, A. Pitrou, S. Seibert, Numba: a LLVM-based Python JIT compiler, in: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC - LLVM '15, ACM Press, Austin, Texas, 2015, pp. 1–6.
[67] C. Lattner, V. Adve, LLVM: A compilation framework for lifelong program analysis & transformation, in: Proceedings of the International Symposium on Code Generation and Optimization: Feedback-Directed and Runtime Optimization, in: CGO '04, IEEE Computer Society, USA, 2004, p. 75, event-place: Palo Alto, California.
[68] E. Sande, C. Manni, H. Speleers, Explicit error estimates for spline approximation of arbitrary smoothness in isogeometric analysis, Numer. Math. 144 (4) (2020) 889–929.
[69] E. Rank, A. Düster, V. Nübel, K. Preusch, O. Bruhns, High order finite elements for shells, Comput. Methods Appl. Mech. Engrg. 194 (21–24) (2005) 2494–2512.
[70] A. Der Kiureghian, J.-B. Ke, The stochastic finite element method in structural reliability, Probab. Eng. Mech. 3 (2) (1988) 83–91.