Matrix Generation in Isogeometric Analysis by Low Rank Tensor Approximation

A. Mantzaflaris[1], B. Jüttler[1,2], B.N. Khoromskij[3], and U. Langer[1,2]

[1] RICAM, Austrian Academy of Sciences, Linz, Austria
[2] Johannes Kepler University, Linz, Austria
[3] MPI for Mathematics in the Sciences, Leipzig, Germany

Abstract. It has been observed that the task of matrix assembly in Isogeometric Analysis (IGA) is more challenging than in the case of traditional finite element methods. The additional difficulties associated with IGA are caused by the increased degree and the larger supports of the functions that occur in the integrals defining the matrix elements. Recently we introduced an interpolation-based approach that approximately transforms the integrands into piecewise polynomials and uses look-up tables to evaluate their integrals [17, 18]. The present paper relies on this earlier work and proposes to use tensor methods to accelerate the assembly process further. More precisely, we show how to represent the matrices that occur in IGA as sums of a small number of Kronecker products of auxiliary matrices that are defined by univariate integrals. This representation, which is based on a low-rank tensor approximation of certain parts of the integrands, makes it possible to achieve a significant speedup of the assembly process without compromising the overall accuracy of the simulation.

Key words: isogeometric analysis, matrix assembly, tensor decomposition, low rank tensor approximation, numerical integration, quadrature

1 Introduction

Isogeometric Analysis (IGA) has been conceived as a new approach to reconcile the conflicting approaches that are used in design and analysis [5, 8]. This is achieved by using NURBS (non-uniform rational B-splines) for defining the discretization spaces that provide the basis of a numerical simulation. IGA has led to several new challenges, one of which is to find efficient methods for assembly, i.e., for evaluating the elements of the matrices and vectors that appear in the linear systems arising in isogeometric simulations.

The standard approach to perform the assembly task consists in using Gaussian quadrature, but this does not give optimal runtimes in IGA, due to its high computational cost. On the one hand, this has motivated the use of GPU programming [11, 12] for accelerating the assembly process. On the other hand, several alternatives to Gauss quadrature have been explored. Special quadrature rules for spline functions [2, 4, 9, 19] have been derived, but these rules are

potentially difficult to compute (since they require solving a non-linear system of equations) and provide only modest improvements. Reduced Bézier element quadrature rules for low degree discretizations have been investigated recently [21]. Another approach, which is based on spline projection and exact integration via look-up tables, has been presented in [17, 18]. The asymptotic time complexity of the method is shown to be $\mathcal{O}(np^{2d})$, where n is the number of elements, p is the spline degree and d the spatial dimension. While this approach provides a significant speedup when using larger polynomial degrees (i.e. degree more than three), it did not exhibit strong improvement with respect to using Gauss quadrature for the low degree case. Similarly, the sum-factorization technique, applied to IGA in [1], provides a significant improvement to the cost of Gauss quadrature, by reducing it to $\mathcal{O}(np^{2d+1})$. Again, the benchmarks presented in this work indicate that this advantage becomes significant for higher polynomial degrees. Finally, it has been proposed to derive discretizations via collocation [20], but the theoretical foundations of this mathematical technology are less well understood than in the case of Galerkin projection.

Tensor-product splines are a very popular approach to construct multivariate representations in IGA. In fact, bi- and tri-variate parameterizations in IGA rely almost exclusively on tensor-product representations. Tensor methods are also a well-known approach to deal with high-dimensional problems and to address the "curse of dimensionality" for simulations of high-dimensional problems.

A major breakthrough concerning the problem of data-sparse representation of multivariate functions and operators on large fine grids in high dimensions was made possible by employing the principle of separation of variables. Modern grid-based tensor methods [14, 15] achieve linear memory costs $\mathcal{O}(dn)$ with respect to dimension d and grid size n. The novel method of quantized tensor approximation is proven to provide a logarithmic data-compression for a wide class of discrete functions and operators [13]. It allows to discretize and to solve multi-dimensional steady-state and dynamical problems with a logarithmic complexity in the volume size of the computational grid.

Even though the problems considered in isogeometric analysis are defined almost exclusively on computational domains of dimension ≤ 3, the use of tensor methods is a promising approach to reduce the computational costs that are associated with isogeometric discretizations, due to the tensor-product structure of the spline spaces that are used for the discretization.

In fact, the complexity of any quadrature-based assembly is bounded from below by the size of the matrix which is to be generated. A further improvement is possible only when considering alternative representations of this matrix, and such representations can be constructed by using tensor decomposition methods. We will show that the use of low rank tensor approximations provides significant gains and is feasible for non-trivial geometries.

In the present paper, we obtain a representation of the matrix as a sum of a few Kronecker products of lower-dimensional matrices. Since linear algebra operations can be performed efficiently on the Kronecker product structure [7], this representation is directly useful for solving the PDE, e.g., when using itera-

tive solvers that are based on matrix-vector multiplications. Nevertheless, even computing the (sparse) Kronecker product as a final step of the assembly has optimal computational complexity, simply because it can be performed in time asymptotically equal to the size of its output.

The remainder of the paper is organized as follows. Section 2 describes the problem of matrix assembly in IGA. The following section recalls the spline projection of the integrands, which was presented in detail in [18]. Section 4 introduces the concept of tensor decomposition and Section 5 shows how this can be exploited for an efficient evaluation of the matrix elements. The efficiency of the overall approach is demonstrated by numerical results, which are described in Section 6. Finally, we conclude the paper in Section 7.

2 Preliminaries

The spaces used for the discretization of partial differential equations on two- and three-dimensional domains in isogeometric analysis are constructed almost exclusively with the help of (polynomial or rational) tensor-product splines. In order to keep the presentation simple, we restrict the presentation to polynomial splines and to the bivariate case. The case of rational splines (non-uniform rational B-splines – NURBS) can be dealt with similarly.

We consider two *univariate spline bases*

$$\boldsymbol{S}(s) = (S_1(s), \ldots, S_m(s))^T \text{ and } \boldsymbol{T}(t) = (T_1(t), \ldots, T_n(t))^T$$

of degree p, which are obtained from the knot vectors

$$(\sigma_1, \ldots, \sigma_{p+m+1}) \text{ and } (\tau_1, \ldots, \tau_{p+n+1}),$$

respectively. The B-splines $S_i(s)$ and $T_j(t)$ are defined by the well-known recurrence formulas, see [6]. We use column vectors $\boldsymbol{S}$ and $\boldsymbol{T}$ to collect the B-splines forming the two spline bases.

More precisely, the boundary knots appear with multiplicity $p+1$, the inner knots have multiplicity at most p, and the knot vectors are non-decreasing. Moreover we choose $\sigma_1 = \tau_1 = 0$, $\sigma_{p+m+1} = \tau_{p+n+1} = 1$.

The *tensor-product spline space*

$$\mathbb{S} = \text{span } \boldsymbol{S} \otimes \text{span } \boldsymbol{T} = \text{span } \{N_{\boldsymbol{i}} : (1,1) \leq \boldsymbol{i} \leq (m,n)\}$$

is spanned by the products of pairs of univariate B-splines,

$$N_{\boldsymbol{i}}(s,t) = S_{i_1}(s) T_{i_2}(t),$$

which are identified by double-indices $\boldsymbol{i} = (i_1, i_2)$.

The isogeometric approach is based on a parameterization of the computational domain Ω. This domain is represented by a suitable *geometry mapping* $\boldsymbol{G} = (G^{(1)}, G^{(2)})$,

$$\boldsymbol{G} : [0,1]^2 \to \Omega \subset \mathbb{R}^2.$$

The domain parameterization is assumed to be regular, i.e., $\det(\nabla \boldsymbol{G})$ does not change its sign in $[0,1]^2$. It is described by the two univariate spline bases (one for each coordinate)

$$G^{(k)}(s,t) = \boldsymbol{S}(s)^T \, \boldsymbol{D}^{(k)} \, \boldsymbol{T}(t), \quad k = 1, 2.$$

The coefficients of the two coordinate functions form the $m \times n$-matrices

$$\boldsymbol{D}^{(k)} = (D_{\boldsymbol{i}}^{(k)})_{(1,1) \le \boldsymbol{i} \le (m,n)}.$$

Equivalently, this mapping can be written as

$$\boldsymbol{G}(s,t) = \sum_{(1,1) \le \boldsymbol{i} \le (m,n)} N_{\boldsymbol{i}}(s,t) \, \boldsymbol{d}_{\boldsymbol{i}}$$

with the control points

$$\boldsymbol{d}_{\boldsymbol{i}} = (D_{\boldsymbol{i}}^{(1)}, D_{\boldsymbol{i}}^{(2)}),$$

whose coordinates are the elements of the matrices $\boldsymbol{D}^{(k)}$, $k = 1, 2$.

The discretization space used in isogeometric analysis is spanned by the functions

$$\beta_{\boldsymbol{i}} = N_{\boldsymbol{i}} \circ \boldsymbol{G}^{-1} : \ \beta_{\boldsymbol{i}}(\boldsymbol{G}(s,t)) = \boldsymbol{N}_{\boldsymbol{i}}(s,t). \tag{1}$$

In many situations, the coefficient matrix of the resulting system of linear equations can be constructed from the mass and the stiffness matrices that are associated with the functions (1). We do not discuss the construction of the right-hand side vectors in this paper, since it is very similar.

The elements of the *mass matrix* take the form

$$\begin{aligned} a_{\boldsymbol{ij}} &= \int\!\!\int_{\Omega} \beta_{\boldsymbol{i}}(\boldsymbol{x}) \beta_{\boldsymbol{j}}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\ &= \int_0^1 \int_0^1 S_{i_1}(s) \, T_{i_2}(t) \, S_{j_1}(s) \, T_{j_2}(t) \ \underbrace{|\det((\nabla \boldsymbol{G})(s,t))|}_{\star} \, \mathrm{d}s \, \mathrm{d}t. \end{aligned}$$

The integrand is a product of two terms. The second one, which has been marked by an asterisk, is independent of the indices $\boldsymbol{i}$ and $\boldsymbol{j}$ and therefore shared by all matrix elements.

The *stiffness matrix* contains the elements

$$\begin{aligned} b_{\boldsymbol{ij}} &= \int\!\!\int_{\Omega} (\nabla \beta_{\boldsymbol{i}})(\boldsymbol{x}) \cdot (\nabla \beta_{\boldsymbol{j}})(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \qquad (2) \\ &= \sum_{k=1}^{2} \sum_{\ell=1}^{2} \int_0^1 \int_0^1 (\partial^{\delta_1^k} S_{i_1})(s) \, (\partial^{\delta_1^\ell} T_{i_2})(t) \\ &\qquad\qquad (\partial^{\delta_2^k} S_{j_1})(s) \, (\partial^{\delta_2^\ell} T_{j_2})(t) \ \underbrace{q_{k,\ell}(s,t)}_{\star} \, \mathrm{d}s \, \mathrm{d}t, \end{aligned}$$

where the weight functions $q_{k,\ell}$ form the 2×2-matrix

$$Q = |\det(\nabla \boldsymbol{G})|(\nabla \boldsymbol{G})^{-1} K (\nabla \boldsymbol{G})^{-T}.$$

Each of the four integrands is again a product of two terms. Once more, the second one (marked by an asterisk) is independent of the indices $\boldsymbol{i}$ and $\boldsymbol{j}$ and therefore shared by all matrix elements.

The matrix (or matrix-valued function) K contains material coefficients (or is the identity matrix in the simplest case). The symbol ∂ denotes the differentiation operator for univariate functions, i.e., $\partial f = f'$. The Kronecker symbol δ specifies whether a B-spline is differentiated or not.

The stiffness matrix (2) arises when considering the weak form of the diffusion equation, i.e., when solving the following elliptic problem:

$$\text{Find } u \in H^1(\Omega) \text{ such that } \begin{cases} -\nabla \cdot (K \nabla u) = f \text{ in } \Omega \\ u = 0 \text{ on } \partial\Omega \end{cases} \tag{3}$$

with the right-hand side $f : \Omega \to \mathbb{R}$, homogeneous boundary conditions, and the (possibly non-constant) symmetric and uniformly positive definite matrix $K : \Omega \to \mathbb{R}^{2\times 2}$. This problem was also considered in [18] when introducing the method of *integration by interpolation and look-up (IIL)*.

Our approach (introduced in [18]) to the computation of the stiffness and mass matrix elements – which we will refer to as *matrix assembly* – relies on an exact or approximate *projection* of the integrands into suitable spline spaces. More precisely, it is performed in two steps.

1. The terms in the integrals that have been marked by the asterisk $\star$ are independent of the indices $\boldsymbol{i}, \boldsymbol{j}$ and therefore shared by all matrix elements. These terms, which will be called the *weight functions*, are replaced by suitable tensor-product spline functions.
2. The resulting integrals of piecewise polynomial functions, are evaluated exactly, either using look-up tables (as described in [18]) or Gauss quadrature.

In [18], the first step was performed solely by interpolation or quasi-interpolation. The present paper improves the first step by employing *low-rank tensor approximations* for constructing sparse representations of the weight functions. This also implies substantial modifications on the second step, since the integration can be reduced to the evaluation of a small number of univariate integrals.

3 Spline projection

The first step of the computation uses spline spaces $\bar{\mathbb{S}}$, which are (potentially) different from those used to define the geometry mapping and the isogeometric discretizations. We consider univariate spline bases

$$\bar{\boldsymbol{S}}(s) = (\bar{S}_1(s), \ldots, \bar{S}_{\bar{m}}(s))^T \text{ and } \bar{\boldsymbol{T}}(t) = (\bar{T}_1(t), \ldots, \bar{T}_{\bar{n}}(t))^T$$

consisting of B-splines of degree $\bar{p}$, which are defined by knot vectors

$$(\bar{\sigma}_1, \ldots, \bar{\sigma}_{\bar{p}+\bar{m}+1}) \text{ and } (\bar{\tau}_1, \ldots, \bar{\tau}_{\bar{p}+\bar{n}+1}),$$

respectively. Once again, the boundary knots possess multiplicity $\bar{p}+1$, the inner knots have multiplicity less or equal to $\bar{p}$, the knot vectors are non-decreasing, $\sigma_1 = \tau_1 = 0$, and $\sigma_{\bar{p}+\bar{m}+1} = \tau_{\bar{p}+\bar{n}+1} = 1$. In practice, these new knots will be either equal to the original knots of the coarsest isogeometric discretization (determined by the geometry mapping) or a superset of those. For convinience, in this section we consider equidistant (i.e., uniform) inner knots. In this situation, the element size (which quantifies the fineness of the discretization) satisfies $h = \mathcal{O}(\max\{1/m, 1/n\})$ as $m, n \to \infty$. Also, we consider the same degree $\bar{p}$ in both directions only for the sake of simplicity.

The products $\bar{N}_{\boldsymbol{i}}(s,t) = \bar{S}_{i_1}(s)\bar{T}_{i_2}(t)$ of pairs of univariate B-splines span another tensor-product spline space

$$\bar{\mathbb{S}} = \text{span } \bar{\boldsymbol{S}} \otimes \text{span } \bar{\boldsymbol{T}} = \text{span } \{ \bar{N}_{\boldsymbol{i}} \ : \ (1,1) \le \boldsymbol{i} \le (\bar{m}, \bar{n}) \}.$$

We are interested in the approximate evaluation of integrals of the form

$$C_{\boldsymbol{ij}}^{\boldsymbol{d}}(w) = \int_0^1 \int_0^1 (\partial^{d_1} S_{i_1})(s)\, (\partial^{d_3} T_{i_2})(t)\, (\partial^{d_2} S_{j_1})(s)\, (\partial^{d_4} T_{j_2})(t)\, w(s,t)\, \mathrm{d}s\, \mathrm{d}t. \tag{4}$$

These integrals depend on the *weight function* w and on the three multi-indices $\boldsymbol{i}, \boldsymbol{j} \in \mathbb{Z}^2$ and $\boldsymbol{d} \in \{0,1\}^4$. The first two indices identify the corresponding matrix element. The remaining one specifies the order of differentiation of the univariate B-splines.

Indeed, the elements of the mass and stiffness matrix can be rewritten as

$$a_{\boldsymbol{ij}} = C_{\boldsymbol{ij}}^{(0,0,0,0)}(|\det(\nabla \boldsymbol{G})|) \text{ and } b_{\boldsymbol{ij}} = \sum_{k=1}^{2} \sum_{\ell=1}^{2} C_{\boldsymbol{ij}}^{(\delta_1^k, \delta_2^k, \delta_1^\ell, \delta_2^\ell)}(q_{k\ell}).$$

The *first step* of the matrix assembly is to project the weight function into the spline space $\bar{\mathbb{S}}$. It is realized by using a *spline projection operator*

$$\Pi : \mathcal{C}([0,1]^2) \to \bar{\mathbb{S}},$$

e.g., interpolation or quasi-interpolation. A detailed discussion is given in [18, Section 4.3].

More precisely, for each weight function w we create a tensor-product spline approximation $\Pi w \in \bar{\mathbb{S}}$. It is represented in the tensor-product B-spline basis,

$$(\Pi w)(s,t) = \bar{\boldsymbol{S}}(s)^T\, \boldsymbol{W}\, \bar{\boldsymbol{T}}(t)^T \tag{5}$$

with a $\bar{m} \times \bar{n}$ coefficient matrix $\boldsymbol{W}$. Replacing the weight functions (4) with their spline approximations then leads to the *approximate integrals*

$$C_{\boldsymbol{ij}}^{\boldsymbol{d}}(\Pi w). \tag{6}$$

The choice of the spline space $\bar{\mathbb{S}}$ depends on the integrand that is to be computed (i.e., mass or stiffness matrix).

For evaluating the *mass matrix* we use degree $\bar{p} = 2p$ and choose the multiplicity of the knots such that smoothness of the spline functions is equal to the smoothness of $\nabla \boldsymbol{G}$. More precisely, if the geometry mapping possesses an inner knot σ_i of multiplicity μ, then the corresponding knot $\bar{\sigma}$ of $\bar{\mathbb{S}}$ has multiplicity $\bar{\mu} = p + \mu + 1$, and similar for the knots τ_j. It should be noted that this applies only to the knots that are present in the actual geometry mapping, but not to the knots which are created later to obtain a finer isogeometric discretization (i.e. by h-refinement). This choice of knots allows to represent the weight function $|\det(\nabla \boldsymbol{G})|$ exactly in $\bar{\mathbb{S}}$. Thus, no error is introduced for the mass matrix.

For assembling the *stiffness matrix* we use $\bar{p} = p$ as discussed in [18]. Moreover, the knots which are already present in the geometry mapping are kept and their multiplicity is increased by one. Additional single knots can be inserted in order to improve the accuracy of the spline projection.

In the *second step* of matrix assembly, the approximations of the integrals are evaluated exactly, using either Gauss quadrature or suitable look-up tables. This is possible since the integrands are simply piecewise polynomial functions.

For the stiffness matrix, the error ε_{Π} introduced by the spline projection

$$\varepsilon_{\Pi} = \max_{k,\ell=1,2} \|q_{k,\ell} - \Pi q_{k,\ell}\|_{\infty,[0,1]^2}$$

is closely related to the overall error of the isogeometric simulation. We consider the problem (3), which leads to the stiffness matrix (2). Combining [18, Corollary 12] with Strang's first lemma (cited as [18, Lemma 2]) and using results of [3] (cited as [18, Eq. 31]) gives the bound

$$\|u - u^\star\|_{1,\Omega} \leq C_1 h^p + C_2 \varepsilon_{\Pi}$$

for the H^1 (semi-) norm of the difference between the exact solution u and the result $u^\star$ of the isogeometric simulation, which is based on the approximate integrals (6). The first term of the right-hand side is the *discretization error*, which is caused by choosing $u^\star$ from the finite-dimensional space of isogeometric functions. The second term represents the *consistency error* which is due to the approximate evaluation of the integrals.

In order to keep the presentation simple we do not consider the right-hand side explicitly. Instead we will simply assume that its effect is also included in the error bound for the consistency error of the stiffness matrix.

Consequently, under suitable assumptions about the error introduced by the spline projection, the matrix assembly preserves the overall order of approximation. Since we use projection into a spline space $\bar{\mathbb{S}}$ of degree $\bar{p}$, we may expect that the difference between the weight functions and their spline projections is bounded by

$$\varepsilon_{\Pi} \leq C_3 h^{\bar{p}+1}, \tag{7}$$

where the constant C_3 depends only on the problem (geometry map, material properties, right-hand side) but not on the isogeometric discretization (i.e., on

the knots which are introduced in addition to the knots of the geometry mapping). This has been formulated in [18, Assumption 5], and a possibility to find a spline projection providing theoretical guarantees (based on quasi-interpolation operators) is also described there.

Choosing $\bar{p} = p - 1$ was proved to be sufficient to preserve the overall order of approximation with respect to the H^1 semi-norm (according to Strang's first lemma). However, this choice was experimentally found to give sub-optimal results for the convergence with respect to the L^2 norm for even degrees p, while $\bar{p} = p$ gave optimal results with respect to both norms in all cases. We will therefore assume that assumption (7) is satisfied with $\bar{p} = p$ in the remainder of this paper.

Once a spline projection has been computed, the actual error ε_{Π} can be estimated by sampling.

4 Tensor decomposition

We consider a singular value decomposition

$$\boldsymbol{W} = \boldsymbol{U}^T \boldsymbol{\Sigma} \boldsymbol{V}$$

of the coefficient matrix in (5). It consists of the diagonal $m \times n$ matrix of singular values $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_{\min(m,n)})$, and of the two orthogonal matrices

$$\boldsymbol{U}^T = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m) \text{ and } \boldsymbol{V}^T = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n),$$

which are represented by their row vectors $\boldsymbol{u}_i \in \mathbb{R}^m$ and $\boldsymbol{v}_j \in \mathbb{R}^n$. More precisely, the *rows* of these two matrices are given by the *column vectors* $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$. Without loss of generality we assume that the singular values are ordered, $\sigma_r \geq \sigma_{r+1}$.

Consequently, we may rewrite the spline projection of the weight function (5) as a sum of products of univariate spline functions, as follows:

$$(\Pi w)(s,t) = \bar{\boldsymbol{S}}(s)^T \boldsymbol{W}\, \bar{\boldsymbol{T}}(t) = \sum_{r=1}^{\min(m,n)} \left(\bar{\boldsymbol{S}}(s)^T \boldsymbol{u}_r \sqrt{\sigma_i}\right) \left(\sqrt{\sigma_i} \boldsymbol{v}_r{}^T \bar{\boldsymbol{T}}(t)\right).$$

As we will see in the next section, using this representation and truncated versions thereof makes the numerical quadrature very efficient.

While the representation in (5) is a sum of mn products of univariate spline functions (one for each tensor-product B-spline), using singular value decomposition allows to reduce the number of terms in the sum to $\min(m, n)$.

The spline projection Πw of the weight function can be approximated by omitting the terms that correspond to smaller singular values. We choose a positive integer $R \leq \min(m, n)$, which specifies the number of terms to be retained by the approximation, and define the *rank R tensor approximation operator*

$$\Lambda_R : \mathcal{C}([0,1]^2) \to \bar{\mathbb{S}}.$$

Given a weight function w, we project into the spline space $\bar{\mathbb{S}}$, generate a singular value decomposition of the coefficient matrix and keep only the contributions of the largest R singular values,

$$(\Lambda_R w)(s,t) = \sum_{r=1}^{R} \left(\bar{\boldsymbol{S}}(s)^T \sqrt{\sigma_r}\boldsymbol{u}_r\right) \left(\bar{\boldsymbol{T}}(t)^T \sqrt{\sigma_r}\boldsymbol{v}_r\right). \tag{8}$$

The function is generated by this operator is called a *low rank* (more precisely: rank R) *tensor approximation of the weight function* w, and R is called its *rank*.

For typical geometry mappings, using a small rank R is sufficient and gives accurate results. The Frobenius norm of the difference between the coefficient matrices can be bounded as follows:

$$\| \sum_{r=R+1}^{\min(m,n)} \sigma_r \boldsymbol{u}_r {\boldsymbol{v}_r}^T \|_F \leq \sqrt{\sum_{r=R+1}^{\min(m,n)} \sigma_r^2}.$$

Since the Frobenius norm of the matrix is an upper bound of the 2-norm, which in turn is an upper bound of the element-wise maximum norm, we can use the convex hull property of tensor-product splines to conclude that

$$\|\Pi w - \Lambda_R w\|_{\infty,[0,1]^2} \leq \sqrt{\sum_{i=R+1}^{\min(m,n)} \sigma_r^2}.$$

Once the singular value decomposition has been computed, the error can be controlled by adjusting the rank R.

In particular, performing the singular value decomposition for the four coefficient matrices of the spline projections $\Pi q_{k\ell}$ $(k,\ell = 1,2)$ gives four sets of singular values $(\sigma_{r,k\ell})_{r=1,\ldots,\min(m,n)}$. Each of the four low-rank tensor approximations satisfies the error bound

$$\max_{k,\ell=1,2} \|\Pi q_{k\ell} - \Lambda_R q_{k\ell}\|_{\infty,[0,1]^2} \leq \varepsilon_\Lambda = \max_{k,\ell=1,2} \sqrt{\sum_{r=R+1}^{\min(m,n)} \sigma_{r,k\ell}^2}.$$

The contribution of the low rank tensor approximation to the overall error can be analyzed as in the previous section. We consider again the problem (3), which leads to the stiffness matrix (2). When using the low rank tensor approximation for assembling the stiffness matrix, the H^1 (semi-) norm of the difference between the exact solution u and the result $u^\star$ of the isogeometric discretization, which is based on the approximate integrals

$$C_{\boldsymbol{ij}}^{\boldsymbol{d}}(\Lambda_R w)$$

is bounded by (6)

$$\|u - u^\star\|_{1,\Omega} \leq C_1 h^p + C_2(\varepsilon_\Pi + \varepsilon_\Lambda).$$

Consequently, when choosing the tensor approximation error ε_Λ in the same order as the spline projection error ε_Π – which can be estimated by sampling – then the order of accuracy of the method remains unchanged.

5 Evaluation of matrix elements

Using a low rank tensor approximation (8) of the weight function w allows to rewrite the integrals (4) as sums of products

$$C_{\boldsymbol{ij}}^{\boldsymbol{d}}(\Lambda_R w) = \sum_{r=1}^{R} X_{r,i_1 j_1}^{\boldsymbol{d}}(\Lambda_R w) Y_{r,i_2 j_2}^{\boldsymbol{d}}(\Lambda_R w)$$

of R pairs of univariate integrals

$$X_{r,ij}^{\boldsymbol{d}}(\Lambda_R w) = \int_0^1 (\partial^{d_1} S_i)(s)\ (\partial^{d_2} S_j)(s)\ (\bar{\boldsymbol{S}}(s)^T \sqrt{\sigma_r}\boldsymbol{u}_r)\,\mathrm{d}s \text{ and}$$
$$Y_{r,ij}^{\boldsymbol{d}}(\Lambda_R w) = \int_0^1 (\partial^{d_3} T_i)(t)\ (\partial^{d_4} T_j)(t)\ (\bar{\boldsymbol{T}}(t)^T \sqrt{\sigma_r}\boldsymbol{v}_r)\,\mathrm{d}t.$$

For each multi-index index $\boldsymbol{d}$, the integrals (4) form a $mn \times mn$ matrix

$$C^{\boldsymbol{d}}(\Lambda_R w) = (C_{\boldsymbol{ij}}^{\boldsymbol{d}}(\Lambda_R w))_{(1,1)\le \boldsymbol{i},\boldsymbol{j} \le (m,n)},$$

and the univariate integrals form R $m \times m$ and R $n \times n$ matrices

$$X_r^{\boldsymbol{d}}(\Lambda_R w) = (X_{r,ij}^{\boldsymbol{d}}(\Lambda_R w))_{1\le i,j\le m} \text{ and } Y_r^{\boldsymbol{d}}(\Lambda_R w) = (Y_{r,ij}^{\boldsymbol{d}}(\Lambda_R w))_{1\le i,j\le n}.$$

The first matrix is the sum of R Kronecker products of matrices obtained from the univariate matrices,

$$C^{\boldsymbol{d}}(\Lambda_R w) = \sum_{r=1}^{R} X_r^{\boldsymbol{d}} \otimes Y_r^{\boldsymbol{d}}. \tag{9}$$

This structure suggests itself for designing an efficient evaluation procedure: In the first step, the matrices $X_r^{\boldsymbol{d}}(\Lambda_R w)$ and $Y_r^{\boldsymbol{d}}(\Lambda_R w)$ which are defined by the univariate integrals are evaluated, e.g., by using Gaussian quadrature. Then, in a second step, the matrix is assembled by summing up the Kronecker products in (9).

We conclude this section with a brief analysis of the computational complexity of the overall procedure, where we assume $m \cong n$ and $\bar{p} = p$. Moreover we assume that the knot vectors of the univariate spline bases $\boldsymbol{S}$ and $\bar{\boldsymbol{S}}$ have the same knots (possibly with different multiplicities) and hence $\mathcal{O}(m)$ knot spans. A similar assumption is made concerning $\boldsymbol{T}$ and $\bar{\boldsymbol{T}}$.

We do not include the singular value decomposition (SVD) into the complexity analysis. In practice one will prefer approximate SVD methods with lower computational costs. A detailed discussion of such methods is beyond the scope of the present paper.

We analyze the computational costs for assembling the $2R$ matrices $X_r^{\boldsymbol{d}}(\Lambda_R w)$ and $Y_r^{\boldsymbol{d}}(\Lambda_R w)$, which possess $\mathcal{O}(mp)$ non-zero entries each.

When using Gaussian quadrature, we need to evaluate the non-zero B-spline basis functions (and their derivatives) at $\mathcal{O}(p)$ Gauss nodes for each of the

$\mathcal{O}(m)$ knot spans at costs of $\mathcal{O}(p^2)$ for each point. In addition we evaluate $2R$ spline functions using de Boor's algorithm. The total complexity of this step is $\mathcal{O}(Rmp^3)$.

We then continue with the assembly step, where each Gauss node contributes to $\mathcal{O}(p^2)$ matrix entries of each of the $2R$ matrices. Thus, the effort for each Gauss node equals $\mathcal{O}(Rp^2)$. The total complexity of this step amounts to $\mathcal{O}(Rmp^3)$.

The matrix (9) has $\mathcal{O}(m^2p^2)$ non-zero entries, and each is evaluated in $\mathcal{O}(R)$ operations from the univariate integrals. Since $m \gg p$, the overall effort is dominated by this step and equals

$$\mathcal{O}(Rm^2p^2).$$

Clearly, the efficiency of the method depends on the rank R which needs to be considered in the tensor approximation. The IIL method described in [18] has complexity of $\mathcal{O}(m^2p^4)$, and this was found to compare well with the Gauss quadrature, where the complexity was found to be $\mathcal{O}(m^2p^6)$. Consequently, if R is small and does not grow with m (i.e., as $h \to 0$), then matrix assembly by tensor decomposition has a clear advantage.

We will demonstrate in the next section that using a low rank tensor approximation is sufficient in most cases and leads to a substantial speedup.

6 Numerical results

In this section we present numerical examples to demonstrate the power of assembly by tensor decomposition and low rank representations. As benchmark examples, we choose a quarter annulus and a multipatch domain consisting of several patches. The annulus is particularly well suited for our method, since it is a "rank one domain" in the sense that using $R = 1$ is sufficient. The multipatch "yeti footprint" domain, which was taken from [16], is a harder benchmark, since the patches do not have any symmetries. We assemble the mass and stiffness of the underlying tensor B-spline basis. In practice, small polynomial degrees are used for discretization; we use the same ($h-$refined) quadratic B-spline functions of the domain as discrete space.

As a baseline algorithm to compare with we use Gauss quadrature assembly with $p + 1$ nodes per parametric direction (FG = "full" Gauss quadrature). The assembly based on tensor decomposition is abbreviated as TD and the computation of the Kronecker product as KP.

We test the computational effort and memory requirements for computing the mass and stiffness matrices. Since these matrices are symmetric, a straightforward saving is to compute only the lower triangular part, and this was done in all examples.

The singular value decompositions were computed by a Jacobi method. Even though we are interested in the highest magnitude singular values and vectors only, our current implementation computes the full SVD. More sophisticated methods exist that compute reduced SVDs, namely the decomposition up to a given tolerance of the singular values. This will explored in more detail in

our future work. For this reason, in our experiments we do not measure the time required for the computation of a (reduced) SVD. The time depends on the dimension of the interpolation space used and the required tolerance. Indeed, this space is independent on the refinement level of the isogeometric discretization, since we always approximated the weight functions with a very small tolerance (such as machine precision).

All experiments were conducted on a laptop with Intel Core i7 @ 2.70GHz processor having two cores and 6GB of RAM, running Linux. The method is implemented in C++ using the G+Smo library for isogeometric analysis[4].

6.1 Quarter annulus

A quarter annulus can be regarded as a line segment swept along a circular arc. Therefore we expect that the tensor rank is low. The SVD computation confirms that the tensor rank is exactly one, and all sub-dominant singular values are equal to zero. Consequently, the assembly of mass and stiffness on this domain using TD is almost as efficient as if we would treat a square domain, by hard-coding the identity matrix in place of $\det \nabla G$.

The domain is represented by linear components in the s direction and by quadratic polynomials in the t direction. Note that we assumed to have the same degree in both directions in the previous section. This can easily be generalized to non-uniform degrees, as in the present example. Therefore we have used 2 and 3 Gauss points respectively, i.e. a total of 6 points per element. For our experiments, we apply uniform $h-$refinement to obtain problem sizes from $6.6 \cdot 10^4$ up to $4.3 \cdot 10^9$ degrees of freedom (dofs).

First we consider the assembly of the mass matrix. The Jacobian determinant of the geometry mapping has rank one and is exactly represented in a spline basis of bidegree (4,2) on the (input) coarse mesh. Therefore no approximation is needed in the SVD step. This implies that the spline space in which the determinant lies has dimension just 15, which is also the size of the matrix that we shall apply SVD computation to. Figure 1 shows the domain and the control grid, as well as the two "skeleton functions" of the rank one tensor representation of the Jacobian determinant. The product of these components is the Jacobian determinant functional, which is shown in Figure 2.

We demonstrate the reduction of the required memory in Table 1. The required memory shrinks from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. In our examples, this translates into using some megabytes of memory instead of several gigabytes.

Table 2 shows the computing times and the speedup with respect to using Gauss quadrature for the mass matrix assembly. Some entries of the table (for high numbers of degrees of freedom) are left blank, since our program aborted, due to insufficient memory; the testing machine (a laptop) has 6GB of memory, which will not fit a matrix with 1G of elements, since this requires 8GB of memory, using double arithmetic. It should be noted that TD was applicable up to 16 levels of dyadic refinement (about $4.3 \cdot 10^9$ dof), in only a fifth of a second.

[4] *Geometry + Simulation Modules*, see http://www.gs.jku.at/gismo, also [10].

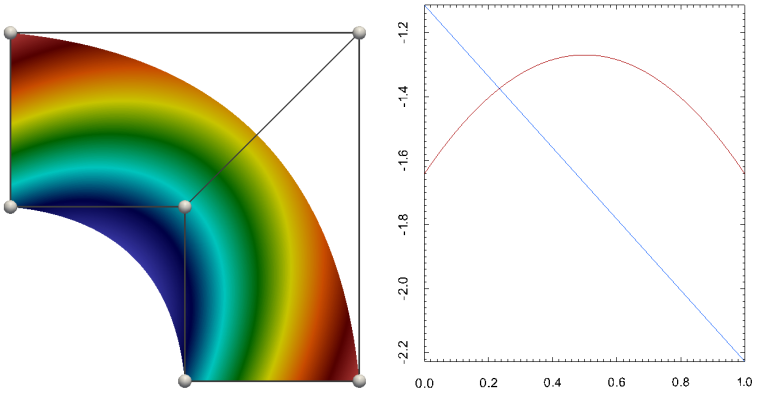

Fig. 1. Left: A B-spline quarter annulus domain. The color field is the value of the Jacobian determinant. Right: "Skeleton functions" of the Jacobian determinant. The linear function is associated with the first direction, while the curved one describes the determinant along the second parametric direction.

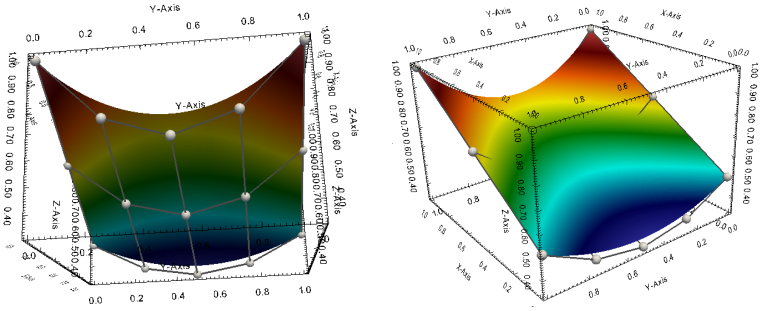

Fig. 2. Two views of the graph of the Jacobian determinant and its control grid over the parameter domain $[0,1]^2$. The degree is 2 in the first direction and 4 in the second one.

Certainly, at this point we are unable to compute the Kronecker product of the skeleton matrices, but given enough memory and processors this is a task that could be performed easily in parallel.

We now turn our attention to the stiffness matrix. We find that the weight functions $q_{k,\ell}$ are also represented by rank one tensor approximations. However, a sufficiently good approximation of these weight functions requires some steps of $h-$refinement. In the experiments of Figure 3, we used a 266K (515×517) grid for the interpolation of these weight functions. This was sufficient to approximate the four functions $q_{\ell,k}$ up to machine precision. Overall, the stiffness matrix has rank four.

Table 3 shows the computational time and reports the speedup for the stiffness matrix assembly. Note that at (dyadic) refinement level 13 we are dealing with around 67.1 million degrees of freedom, and the number of non-zeros in the global matrix reaches 1 billion. When using double precision, this requires

Table 1. Quarter annulus: memory statistics for the (rank one) mass matrix for high levels of dyadic $h-$refinement. The second column shows the number of degrees of freedom per coordinate (tensor) direction. The third column contains the number of non-zero (nz) entries in each factor matrix of the Kronecker product in (9). The non-zero entries of the mass matrix are shown in the last column. The factor matrices in the decomposition of the stiffness matrix use four times as many non-zero elements.

$h-$ref. level	no. of dof	mass matrix TD nz	mass matrix nz
8	257×258	$769 + 1,284$	987K
9	513×514	$1,537 + 2,564$	3.9M
10	$1,025 \times 1,026$	$3,073 + 5,124$	15.7M
11	$2,049 \times 2,050$	$6,145 + 10,244$	62.9M
12	$4,097 \times 4,098$	$12,289 + 20,484$	251.7M
13	$8,193 \times 8,194$	$24,577 + 40,964$	1G
14	$16,385 \times 16,386$	$49,153 + 81,924$	4G
15	$32,769 \times 32,770$	$98,305 + 32,770$	16G
16	$65,537 \times 65,538$	$196,609 + 327,684$	64G

Table 2. Computation times (in seconds) for the mass matrix assembly of the quarter annulus. The last two columns report the speedup (time ratio).

$h-$ref. level	FG	TD	KP	FG/TD	FG/ (TD+KP)
8	0.30	$9 \cdot 10^{-4}$	0.017	339	16.87
9	1.22	$2 \cdot 10^{-3}$	0.07	610	17.9
10	4.82	$5 \cdot 10^{-3}$	0.23	947	20.9
11	18.94	$7 \cdot 10^{-3}$	0.79	2,600	23.6
12	76.21	$1.5 \cdot 10^{-2}$	3.23	5,154	23.4
13	-	$4.1 \cdot 10^{-2}$	-	-	-
14	-	$6.5 \cdot 10^{-2}$	-	-	-
15	-	0.1	-	-	-
16	-	0.18	-	-	-

at least 8GB of memory, and was not feasible on our test machine (laptop). Consequently the columns FG and KP are filled up to refinement level 12.

One may observe that computing times are mostly determined by the memory required for the problem. In particular, note that computing e.g. the mass matrix using FG with 8 levels of refinement implies the computation of a matrix of around 526K elements (taking into account symmetry). This took 0.3 seconds, which is quite similar to using TD with 16 levels of refinement for computing (in 0.18 seconds) a decomposition of a (much bigger) matrix with comparable amount of non-zero elements.

The annulus example has also been used as a benchmark for the IIL method. As observed in [18, Sect. 7.3], the speedup of the stiffness matrix assembly with respect to the baseline FG method is independent of h, that is, the computation time for both FG and IIL is linear with respect to the number of degrees of freedom. We did not include the comparison with IIL in Table 3, since its per-

Table 3. Computation times (in seconds) and speedup for the stiffness matrix assembly of the quarter annulus.

$h-$ref. level	FG	TD	KP	FG/TD	FG/ (TD+KP)
8	0.36	0.019	0.016	19.02	10.2
9	1.45	0.083	0.072	17.40	9.3
10	5.84	0.085	0.28	68.86	15.8
11	23.59	0.089	1.09	264	19.9
12	96.50	0.099	4.56	979	20.7
13	-	0.11	-	-	-
14	-	0.14	-	-	-
15	-	0.21	-	-	-
16	-	0.34	-	-	-

formance is quite similar to FG for low degrees (e.g., it is approximately twice as fast as FG for bi-quadratic discretizations, cf. [18, Fig. 10]).

6.2 Yeti footprint

We now turn to a more challenging benchmark. The domain is a "footprint" domain parameterized by 21 patches (Figure 3) introduced in [16]. All the patches have a number of interior knots per direction, ranging from one to three.

One question is whether the rank of each patch (i.e., the rank of the tensor approximation) stabilizes to a small value. To test this, we set a tolerance of 10^{-10} and compute the rank of $\det \nabla G$ and $q_{k,\ell}$ for all mappings G of the different patches and several levels of $h-$refinement. In all cases, the rank stabilized (i.e., it stayed constant under $h-$refinement) to a value between 4 and 7 for $\det \nabla G$ and to values ranging from 14 to 22 for $q_{k,\ell}$. The exact numbers are shown in Figure 3(left).

Clearly, the full multi-patch (mass or) stiffness matrix is a block-structured matrix where the blocks are the patch-wise matrices. Therefore, we now choose one patch (marked in Figure 3 left) in order to perform our benchmark computation. Figure 4 provides a closer look at the selected patch and its Jacobian determinant.

In Table 4 we report computing times and memory requirements for the mass matrix assembly for the selected patch. The last column shows the memory savings that can be obtained by using a tensor representation, instead of assembling the global mass matrix.

We note that the mass matrix on this patch has rank 7, therefore memory requirements are seven times more than the annulus case, but the percentage shows that this is still minor compared to the size of the full matrix. The timings confirm that also in this case a substantial speedup is possible. In particular, in the last row we assembled a (tensor decomposed) mass matrix for 17.2 billion degrees of freedom, and this took one second. This suggests that in less than

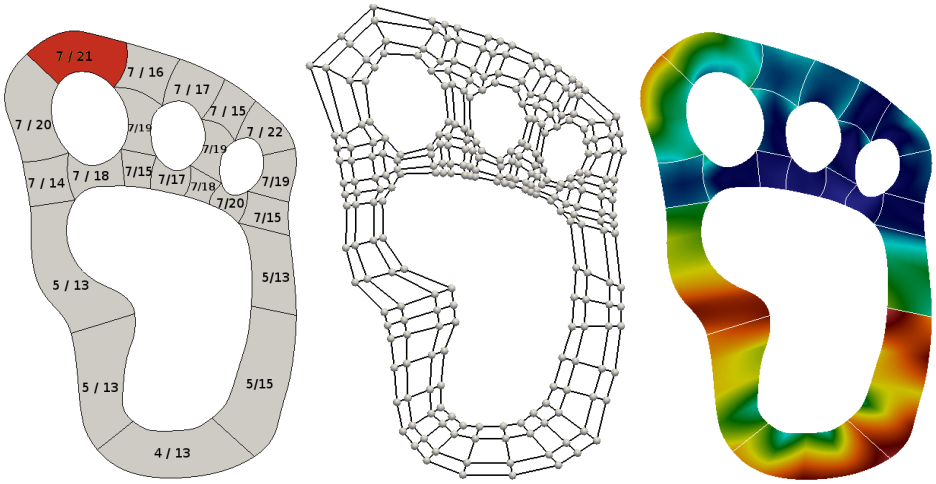

Fig. 3. The yeti footprint domain, parameterized by 21 quadratic B-spline patches. Patch segmentation (left), control grids (middle) and the (absolute) Jacobian determinant are shown. On the left picture $\operatorname{rank}(\det \nabla G)$ and $\max(\operatorname{rank}(q_{k,\ell}),\, k,\ell \in \{1,2\})$ are given.

half a minute, the full mass matrix of all 21 patches can be computed with the same refinement level.

Table 5 reports on the stiffness matrix assembly for the same patch. Each weight $q_{k,\ell}$ has a rank 21 tensor approximation, leading to a total rank of 84 for the tensor form of the stiffness matrix. This increase with respect to the mass matrix is reflected in the computation time and the number of non-zero elements, i.e., in the memory that is required. Nevertheless, the percentage of memory required is still around 3% of the full matrix for one million degrees of freedom, and is reduced to half at every refinement level. The Kronecker product computation becomes slower since now we need to sum up $R = 84$ contributions to compute one entry in the Kronecker product. Nevertheless, we still get a speedup factor 1.52 compared to FG. The speedup observed if we refrain from computing the KP (sixth column) is still quite important and scales nicely with refinement. Finally, from the last row of the table we may estimate that about three minutes' time would suffice for assembling a (decomposed) giant stiffness matrix of 361 billion degrees of freedom on a laptop.

7 Conclusion

We explored the use of low rank tensor approximation for performing the assembly task of the isogeometric mass and stiffness matrices. Clearly, the method is general and applicable to other types of isogeometric matrices, arising in PDE discretizations.

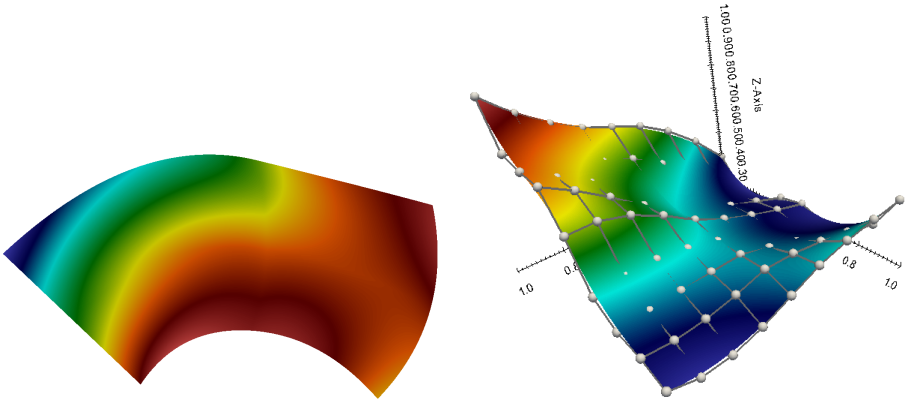

Fig. 4. The Jacobian determinant values of the selected patch (Fig. 3) on the domain (left). The graph and control net in parameter domain (right).

Table 4. Computing times (in seconds), speedup and memory requirements for the mass matrix assembly of the selected patch of the yeti footprint. The levels of (dyadic) refinement vary from 9 to 16.

no. of dof	FG	TD	KP	FG/TD	FG/ (TD+KP)	Mem.(%)
1.0M	7.96	0.015	0.60	544	13.0	0.273
4.2M	32.50	0.025	2.34	1289.14	13.7	0.137
16.8M	129.83	0.053	11.33	2461.57	11.4	0.068
67.1M	-	0.091	-	-	-	0.034
268M	-	0.148	-	-	-	0.017
1.1G	-	0.262	-	-	-	0.009
4.2G	-	0.496	-	-	-	0.004
17.2G	-	0.997	-	-	-	0.002

Our results show that the use of tensor methods in IGA possesses a great potential. The main advantage is the substantial memory reduction that is obtained by creating low rank approximations of large isogeometric matrices. This observation can be further exploited if one adopts the Kronecker representation both for storing matrices and for applying iterative solvers or other operations. Indeed, our benchmarks show that impressive speedups are possible if one avoids the evaluation of the full matrices.

As another advantage, using tensor methods allows to fully exploit the tensor-product structure of the B-spline basis to apply efficient quadrature, also for low polynomial degrees. This has been a long-standing challenge in isogeometric analysis, since the increased support and degree of the basis functions result in a rapidly growing quadrature grid, and affects dramatically the computation times. By reducing the problem to 1D components, the number of evaluations

Table 5. Computing times (in seconds), speedup and memory requirements for the stiffness matrix assembly of the selected patch of the yeti footprint. The levels of (dyadic) refinement vary from 9 to 16.

no. of dof	FG	TD	KP	FG/TD	FG/ (TD+KP)	Mem.(%)
1.0M	9.94	0.073	7.25	136.75	1.36	3.33
4.2M	40.45	0.14	29.1	298.97	1.39	1.66
16.8M	164.4	0.27	108.1	614.38	1.52	0.83
67.1M	-	0.53	-	-	-	0.41
268M	-	1.06	-	-	-	0.21
1.1G	-	2.17	-	-	-	0.10
4.2G	-	4.46	-	-	-	0.05
17.2G	-	8.85	-	-	-	0.02

and quadrature points increase logarithmically with the number of degrees of freedom.

Future research will be devoted to the use of approximate SVD computation and to the extension to the trivariate case.

Acknowledgement. This research was supported by the National Research Network "Geometry + Simulation" (NFN S117), funded by the Austrian Science Fund (FWF).

References

1. P. Antolin, A. Buffa, F. Calabrò, M. Martinelli, and G. Sangalli. Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization. *Comp. Meth. Appl. Mech. Engrg.*, 285:817–828, 2015.
2. F. Auricchio, F. Calabrò, T. Hughes, A. Reali, and G. Sangalli. A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis. *Comp. Meth. Appl. Mech. Engrg.*, 249-252:15–27, 2012.
3. Y. Bazilevs, L. Beirão da Veiga, J. A. Cottrell, T. J. R. Hughes, and G. Sangalli. Isogeometric analysis: Approximation, stability and error estimates for h-refined meshes. *Math. Models Methods Appl. Sci.*, 16(7):1031–1090, 2006.
4. F. Calabrò, C. Manni, and F. Pitolli. Computation of quadrature rules for integration with respect to refinable functions on assigned nodes. *Applied Numerical Mathematics*, 90:168–189, 2015.
5. J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. John Wiley & Sons, Chichester, England, 2009.
6. C. De Boor. *A practical guide to splines*. Applied mathematical sciences. Springer, Berlin, 2001.
7. W. Hackbusch. *Tensor spaces and numerical tensor calculus*. Springer, Berlin, 2012.
8. T. Hughes, J. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comp. Meth. Appl. Mech. Engrg.*, 194(39–41):4135–4195, 2005.

9. T. Hughes, A. Reali, and G. Sangalli. Efficient quadrature for NURBS-based isogeometric analysis. *Comp. Meth. Appl. Mech. Engrg.*, 199(5 – 8):301–313, 2010.
10. B. Jüttler, U. Langer, A. Mantzaflaris, S. E. Moore, and W. Zulehner. Geometry + Simulation Modules: Implementing Isogeometric Analysis. *PAMM*, 14(1):961–962, 2014.
11. A. Karatarakis, P. Karakitsios, and M. Papadrakakis. Computation of the isogeometric analysis stiffness matrix on GPU. In M. Papadrakakis, M. Kojic, and I. Tuncer, editors, *Proc. of the 3rd South-East European Conference on Computational Mechanics (SEECCM)*, 2013. http://www.eccomasproceedings.org/cs2013.
12. A. Karatarakis, P. Karakitsios, and M. Papadrakakis. GPU accelerated computation of the isogeometric analysis stiffness matrix. *Comp. Meth. Appl. Mech. Engrg.*, 269:334 – 355, 2014.
13. B. N. Khoromskij. $\mathcal{O}(d \log n)$–Quantics approximation of N–d tensors in high-dimensional numerical modeling. *Constr. Appr.*, 34(2):257–280, 2011.
14. B. N. Khoromskij. Tensor-structured numerical methods in scientific computing: survey on recent advances. *Chemometr. Intell. Lab. Syst.*, 110(1):1–19, 2012.
15. B. N. Khoromskij. Tensor Numerical Methods for Multidimensional PDEs: Theoretical Analysis and Initial Applications. In *Proc. ESAIM*, pages 1–28, 2014.
16. S. Kleiss, C. Pechstein, B. Jüttler, and S. Tomar. IETI – isogeometric tearing and interconnecting. *Comp. Meth. Appl. Mech. Engrg.*, 247-248:201–215, 2012.
17. A. Mantzaflaris and B. Jüttler. Exploring matrix generation strategies in isogeometric analysis. In M. Floater et al., editors, *Mathematical Methods for Curves and Surfaces*, volume 8177 of *Lecture Notes in Computer Science*, pages 364–382. Springer, 2014.
18. A. Mantzaflaris and B. Jüttler. Integration by interpolation and look-up for Galerkin-based isogeometric analysis. *Comp. Methods Appl. Mech. Engrg.*, 284:373 – 400, 2015. Isogeometric Analysis Special Issue.
19. B. Patzák and D. Rypl. Study of computational efficiency of numerical quadrature schemes in the isogeometric analysis. In *Proc. of the 18th Int'l Conf. Engineering Mechanics*, EM'12, pages 1135–1143, 2012.
20. D. Schillinger, J. Evans, A. Reali, M. Scott, and T. Hughes. Isogeometric collocation: Cost comparison with Galerkin methods and extension to adaptive hierarchical NURBS discretizations. *Comp. Meth. Appl. Mech. Engrg.*, 267:170–232, 2013.
21. D. Schillinger, S. Hossain, and T. Hughes. Reduced Bézier element quadrature rules for quadratic and cubic splines in isogeometric analysis. *Comp. Meth. Appl. Mech. Engrg.*, 277(0):1 – 45, 2014.