Inexact Dual-Primal Isogeometric Tearing and Interconnecting Methods

C. Hofer, U. Langer, S. Takacs

RICAM-Report 2017-16

Inexact Dual-Primal Isogeometric Tearing and Interconnecting Methods

Christoph Hofer[1], Ulrich Langer[1,2] and Stefan Takacs[2]

[1] Johannes Kepler University (JKU), Altenbergerstr. 69, A-4040 Linz, Austria,
christoph.hofer@jku.at, ulrich.langer@jku.at

[2] Austrian Academy of Sciences, RICAM, Altenbergerstr. 69, A-4040 Linz, Austria,
ulrich.langer@ricam.oeaw.ac.at, stefan.takacs@ricam.oeaw.ac.at

Abstract. In this paper, we investigate inexact variants of dual-primal isogeometric tearing and interconnecting methods for solving large-scale systems of linear equations arising from Galerkin isogeometric discretizations of elliptic boundary value problems. The considered methods are extensions of standard finite element tearing and interconnecting methods to isogeometric analysis. The algorithms are implemented by means of energy minimizing primal subspaces. We discuss the replacement of local sparse direct solvers by iterative methods, particularly, multigrid solvers. We investigate the incorporation of these iterative solvers into different formulations of the algorithm. Finally, we present numerical examples comparing the performance of these inexact versions.

Key words: Elliptic diffusion problems, Isogeometric analysis, IETI-DP, Inexact solvers, Multigrid

1 Introduction

Isogeometric Analysis (IgA) is a novel methodology for the numerical solution of partial differential equations (PDEs). IgA was first introduced by Hughes, Cottrell and Bazilevs in [7], see also the survey article [1]. In IgA, for both the representation of the geometry and the approximation of the solution, spline-based spaces are chosen. The most common choices are B-Splines, Non Uniform Rational B-Splines (NURBS), T-Splines, Truncated Hierarchical B-Splines (THB-Splines), see [1] and references therein. One of the strengths of IgA consists in its capability of creating high-order smooth function spaces, while keeping the number of degrees of freedom small. Originally, IgA was formulated by means of one *global geometry mapping*, which restricts the method to simple domains being topologically equivalent to the unit square or the unit cube. More complicated domains are represented by decomposing them into such simple domains, called *patches* or *subdomains*. In such a *multi-patch* setting, each of the patches has its own geometry mapping, and all of the patches can be discretized by the use of different spline spaces.

We are interested in fast solvers for linear systems arising from the discretization of elliptic PDEs by means of IgA. We investigate non-overlapping domain decomposition (DD) methods of the dual-primal tearing and interconnecting type. These methods are closely related to the Balancing Domain Decomposition by Constraints (BDDC) methods, see [16,13] and references therein. The version based on a conforming Galerkin (cG) discretization, called dual-primal isogeometric tearing and interconnecting (IETI-DP) method, was first introduced in [10]. The related IgA BDDC method was analyzed in [2]. Typically, the local problems are solved using a sparse Cholesky factorization. However, especially in IgA, one may run out of memory for big problems. A remedy would be to use inexact solvers for the local subproblems, as introduced in [9]. The aim of this work is to investigate how local sparse direct solvers can be replaced by inexact methods, like multigrid

(MG). This leads to several different variants of the IETI-DP algorithm, each with its own advantages and disadvantages.

In the present paper, we consider the following weak formulation of a second-order elliptic boundary value problem (BVP) in a bounded Lipschitz domain $\Omega \subset \mathbb{R}^d$, with $d \in \{2,3\}$, as model problem: Find $u \in V_0 := H_0^1(\Omega)$ such that

$$a(u,v) = \langle F, v\rangle \quad \forall v \in V_0. \tag{1}$$

The bilinear form $a(\cdot,\cdot) : V_0 \times V_0 \to \mathbb{R}$ and the linear form $\langle F, \cdot\rangle : V_0 \to \mathbb{R}$ are given by

$$a(u,v) := \int_\Omega \nabla u \cdot \nabla v \, dx \quad \text{and} \quad \langle F, v\rangle := \int_\Omega f v \, dx,$$

respectively. We assume that the given right hand side function f is sufficiently smooth.

2 Isogeometric Analysis and IETI-DP

On the unit interval, for any spline degree p and number of basis functions M, we define the one dimensional B-Spline basis $(\widehat{N}_{i,p})_{i=1}^M$ via the Cox-De Boor's algorithm, cf. [1]. On the *parameter domain* $\widehat{\Omega} := (0,1)^d$, a multivariate basis is realized by the tensor product of such univariate bases functions, again denoted by $\widehat{N}_{i,p}$, where $i = (i_1, \ldots, i_d) \in \mathcal{I} := \{1,\ldots,M_1\} \times \ldots \times \{1,\ldots,M_d\}$ and $p = (p_1,\ldots,p_d)$ are multi-indices.

In standard (single-patch) IgA, the *physical domain* Ω is given as the image of the parameter domain under the *geometrical mapping* $G : \widehat{\Omega} \to \mathbb{R}^d$, defined by $G(\xi) := \sum_{i\in\mathcal{I}} P_i \widehat{N}_{i,p}(\xi)$, with the control points $P_i \in \mathbb{R}^d$, $i \in \mathcal{I}$. In a multi-patch setting, the domain Ω (*multipatch domain*) is decomposed into non-overlapping patches $\Omega^{(k)}$, $k = 1,\ldots,N$, such that $\overline{\Omega} := \bigcup_{k=1}^N \overline{\Omega}^{(k)}$. Each patch $\Omega^{(k)} := G^{(k)}(\widehat{\Omega})$ is represented by its own geometrical mapping. We call $\Gamma := \bigcup_{k>l} \partial\Omega^{(k)} \cap \partial\Omega^{(l)}$ the *interface*, and denote its restriction to one of the patches $\Omega^{(k)}$ by $\Gamma^{(k)} := \Gamma \cap \partial\Omega^{(k)}$. Here and in what follows, the superscript (k) denotes the restriction of the underlying symbol to the patch $\Omega^{(k)}$.

We use the B-Splines not only for defining the geometry, but also for representing the approximate solution of the BVP. Once the basis functions are defined on the parameter domain $\widehat{\Omega}$, we define the bases on the physical domain $\Omega^{(k)}$ via the standard pull-back principle, and obtain the basis functions $N_{i,p} := \widehat{N}_{i,p} \circ G^{-1}$.

The main idea of IETI-DP is to decouple the patches by tearing the interface unknowns which introduces additional degrees of freedom (dofs). We denote the resulting space by V_h. Then, continuity is again enforced using Lagrange multipliers λ. Doing so, the local subproblems on each patch are essentially pure Neumann problems (at least for interior patches). Therefore, they have a kernel consisting of the constant functions in our case. So, a Schur complement formulation is not possible. In order to overcome this problem, certain continuity conditions are enforced strongly, i.e., by incorporating into the space V_h, (*strongly enforced continuity conditions*) which yields the smaller space $\widetilde{V}_h$. There, we formulate the following problem. Find $(u,\lambda) \in \widetilde{V}_h \times \Lambda$ such that

$$\begin{bmatrix} \widetilde{K} & \widetilde{B}^T \\ \widetilde{B} & 0 \end{bmatrix} \begin{bmatrix} u \\ \lambda \end{bmatrix} = \begin{bmatrix} \widetilde{f} \\ 0 \end{bmatrix}, \tag{2}$$

where $\widetilde{K}$ is the stiffness matrix, $\widetilde{B}$ the jump operator, and $\widetilde{f}$ the right hand side, all in $\widetilde{V}_h$.

As next step, we split V_h into interior dofs and interface dofs, which yields an interface space W. By splitting $\widetilde{V}_h$ analogously, we obtain the space $\widetilde{W}$. Based on this splitting, we formulate the problem using the Schur complement of the stiffness matrix K in V_h with respect to the interface dofs: $S := K_{BB} - K_{BI}(K_{II})^{-1}K_{IB}$, where the subindices B and I denote the boundary and interior dofs, respectively. The restriction of S into $\widetilde{W}$ is denoted by $\widetilde{S}$, which yields the saddle-point formulation of the problem: Find $(w, \lambda) \in \widetilde{W} \times \Lambda$ such that

$$\begin{bmatrix} \widetilde{S} & \widetilde{B}^T \\ \widetilde{B} & 0 \end{bmatrix} \begin{bmatrix} w \\ \lambda \end{bmatrix} = \begin{bmatrix} \widetilde{g} \\ 0 \end{bmatrix}, \tag{3}$$

where $\widetilde{g} := \widetilde{I}^T(f_B - K_{BI}(K_{II})^{-1}f_I)$ and $\widetilde{I}$ is the canonical embedding of $\widetilde{W}$ in W.

We denote the subspace of $\widetilde{W}$ satisfying the strongly enforced continuity conditions homogeneously by W_Δ and the S-orthogonal complement by W_Π. In the literature, our choice of W_Π is often called *energy minimizing primal subspace.* Finally, we can define the Schur complement F of the saddle-point problem (3), and obtain the problem: Find $\lambda \in U$ such that

$$F\lambda = d, \tag{4}$$

where $F := \widetilde{B}\widetilde{S}^{-1}\widetilde{B}^T$ and $d := \widetilde{B}\widetilde{S}^{-1}\widetilde{g}$.

Equation (4) is solved by means of the conjugate gradient (CG) algorithm using the scaled Dirichlet preconditioner $M_{sD}^{-1} := B_D S B_D^T$, where B_D is a scaled version of the jump operator B on V_h. Note that we can approximate $\widetilde{S}^{-1}$ because $\widetilde{S}$ can be represented (by reordering of the dofs) as a block diagonal matrix, consisting of matrices $S_{\Delta\Delta}^{(k)}$ for each patch and the matrix $S_{\Pi\Pi}$. For a summary of the algorithm and a more detailed explanation, we refer, e.g., to [16,13,5] and references therein.

3 Incorporating Multigrid in IETI-DP

We investigate different possibilities to incorporate a multigrid solver into the IETI-DP algorithm. The application of the IETI-DP algorithm requires the solution of linear systems at certain places. Two types of local problems are involved: *Dirichlet problems* and *Neumann problems.*

3.1 Local Dirichlet problems

We have to solve linear systems with system matrix $K_{II}^{(k)}$ in the application of S in the preconditioner and when calculating the right hand side $\widetilde{g}$. These linear systems are Dirichlet problems. (They would have Neumann boundary conditions only if the patch boundary contribute to the Neumann boundary of the whole domain.) The right hand side $\widetilde{g}$ has to be computed very accurately, i.e., at least up to discretization error. However, for the preconditioner, a few MG V-cycles are usually enough, since we only have to ensure the spectral equivalence of the inexact scaled Dirichlet preconditioner to the exact one, cf. [8] and references therein.

3.2 Local Neumann problems

The second class of local problems are Neumann problems. They appear in the construction of the S-orthogonal basis for W_Π and in the application of $S_{\Delta\Delta}$. Let us first investigate the construction of the basis $\{\phi_j^{(k)}\}_j$ for $W_\Pi^{(k)}$. Since we look for a nodal basis, which is S-orthogonal, we have to solve the following linear system

$$\begin{bmatrix} S^{(k)} & {C^{(k)}}^T \\ C^{(k)} & 0 \end{bmatrix} \begin{bmatrix} \phi_j^{(k)} \\ \mu_j^{(k)} \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{e}_j^{(k)} \end{bmatrix}, \quad \forall j \in \{1, \ldots, n_\Pi^{(k)}\}, \tag{5}$$

where $\boldsymbol{e}_j^{(k)} \in \mathbb{R}^{n_\Pi^{(k)}}$ is the j-th unit vector and the matrix $C^{(k)}$ realizes the $n_\Pi^{(k)}$ strongly enforced continuity conditions contributing to the patch $\Omega^{(k)}$. This system has to be solved for $n_\Pi^{(k)}$ right hand sides, which is an advantage for direct solvers over iterative solvers because the expensive factorization must be computed only once. Instead of solving (5) directly, we use the approach proposed in [13], solve

$$\begin{bmatrix} K^{(k)} & {C^{(k)}}^T \\ C^{(k)} & 0 \end{bmatrix} \begin{bmatrix} \overline{\phi}_j^{(k)} \\ \mu_j^{(k)} \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{e}_j^{(k)} \end{bmatrix}, \quad \forall j \in \{1, \ldots, n_\Pi^{(k)}\}, \tag{6}$$

and obtain the desired basis functions by $\phi_j = \overline{\phi}_j|_{\Gamma^{(k)}}$. Note that $\{\overline{\phi}_j^{(k)}\}_j$ is a K-orthogonal basis. If the patch $\Omega^{(k)}$ does not touch the boundary $\partial\Omega$, the upper left block becomes semi-definite due to the presence of a kernel. We are looking for a way to use the CG algorithm. As long as there is no kernel, i.e., where $\partial\Omega^{(k)} \cap \partial\Omega \neq \emptyset$, one straightforward way would be to use the Bramble-Pasciak conjugate gradient (BPCG) algorithm or one of its variations, see [3,15]. However, these iterative methods require that the upper left block is positive definite. The remedy is a special preconditioner and a non standard inner product for the CG algorithm, leading to the *Schöberl-Zulehner* (SZ) preconditioner, see [14]. An alternative approach would be to use the MinRes method with a block diagonal preconditioner, for which our experiments indicated a larger number of iterations.

The SZ preconditioner for (6) requires preconditioners $\hat{K}^{(k)}$ and $\hat{H}^{(k)}$ for the upper left block $K^{(k)}$ and its inexact Schur complement $H^{(k)} := C^{(k)}(\hat{K}^{(k)})^{-1}{C^{(k)}}^T$, respectively. The preconditioner $K^{(k)}$ shall be realized by a few MG V-cycles. It is required that $\hat{K}^{(k)} > K^{(k)}$, which implies that $\hat{K}^{(k)}$ has to be positive definite. In order to handle also the case where $K^{(k)}$ is singular, we need to set up MG based on a regularized matrix $K_M^{(k)} := K^{(k)} + \alpha\widehat{M}^{(k)}$, where α is chosen to be 10^{-2} and $\widehat{M}^{(k)}$ is the mass matrix on the parameter domain. Note, we can exploit the tensor product structure to efficiently assemble the mass matrix $\widehat{M}^{(k)}$. Finally, this provides us with an appropriate preconditioner $\hat{K}^{(k)}$ for $K^{(k)}$. Secondly, the SZ preconditioner requires that $\hat{H}^{(k)} < H^{(k)}$. Since in our case the number of rows of $C^{(k)}$ is given by $n_\Pi^{(k)}$, a small number that does not change during refinement, we calculate the inexact Schur complement exactly. This can be performed by applying $(\hat{K}^{(k)})^{-1}$ to $n_\Pi^{(k)}$ vectors. Finally, by a suitable scaling, e.g., $\hat{H}^{(k)} := 0.99H^{(k)}$, we obtain the desired matrix inequality. Having the preconditioners $\hat{K}^{(k)}$ and $\hat{H}^{(k)}$, we apply CG with the SZ preconditioner to construct the basis for $W_\Pi^{(k)}$.

The second type of Neumann problem appears in the application of F. We look for a solution of the system $S_{\Delta\Delta}^{(k)} w_\Delta^{(k)} = f_\Delta^{(k)}$, which can be written as

$$\begin{bmatrix} S^{(k)} & {C^{(k)}}^T \\ C^{(k)} & 0 \end{bmatrix} \begin{bmatrix} w_\Delta^{(k)} \\ \mu^{(k)} \end{bmatrix} = \begin{bmatrix} f^{(k)} \\ 0 \end{bmatrix}. \tag{7}$$

Certainly, one can use the same method as above. However, we can utilize the fact that we search for a minimizer of $\frac{1}{2}(S^{(k)}w^{(k)}, w^{(k)}) - (w^{(k)}, f^{(k)})$ in the subspace given by $C^{(k)}w^{(k)} = 0$. This solution can be computed by first solving the unconstrained problem and projecting the minimizer into the subspace using a energy-minimizing projection. The projection is trivial because the decomposition of $\widetilde{W}$ into W_Π and W_Δ is S-orthogonal.

Note that the CG algorithm, when applied to a positive semidefinite matrix, stays in the factor space with respect to the kernel and computes one of the minimizers. The solution of the constrained minimization problem is, as outlined above, obtained by applying the projection. As long as the number of CG iterations is not too large, numerical instabilities are not observed when applying CG to a positive semidefinite problem.

The S-orthogonal basis has to be computed very accurate in order to maintain the orthogonality. Because the equation $S_{\Delta\Delta}^{(k)}w_\Delta^{(k)} = f_\Delta^{(k)}$ appears in the system matrix F, its solution also requires an accuracy of at least the discretization error.

3.3 Variants of inexact formulations

From the discussion above, we deduce four (reasonable) combinations of the IETI-DP method with direct solvers and MG.

(D-D) This is the classical IETI-DP method, where we use direct solvers everywhere.

(D-MG) We use MG in the scaled preconditioner for the solution of the local Dirichlet problems and the transformation of the right hand side, see Section 3.1. As already mentioned, the required accuracy for computing $\widetilde{g}$ has to be of the order of discretization error, whereas for the preconditioner, a few V-cycles are enough.

(MG-MG) We use MG for all patch local problems, i.e., the local Dirichlet and Neumann problems. This implies that also the calculation of the basis for W_Δ is performed by means of MG, which turns out to be very costly. Moreover, for each application of F, we have to solve a local Neumann problem in W_Δ with the accuracy in the order of the discretization error.

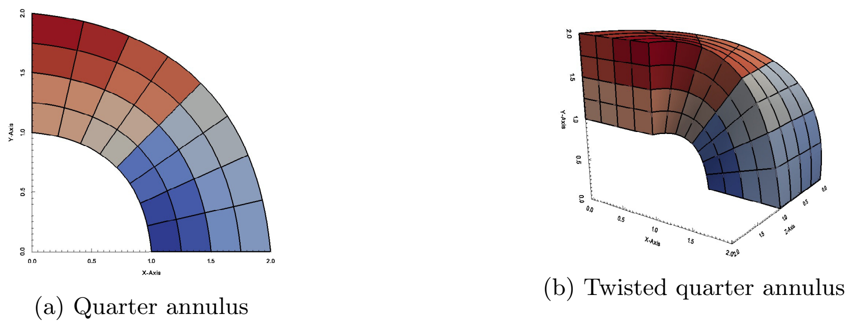
(MG-MG-S) To overcome the efficiency problem of applying MG at each iteration up to a small precision, we use the saddle point formulation instead of F. On the one hand, at each iteration step we only have to apply a given matrix instead of solving a linear system. On the other hand, we now have to deal with a saddle point problem. Moreover, the iteration is not only applied to the interface dofs, but also to the dofs in the whole domain.

We will always assume that the considered multipatch domain has only a moderate number of patches, such that the coarse problem can still be handled by a direct solver. For extensions to inexact version for the coarse problem, we refer to, e.g., [9].

For the first three methods, we use the CG method to solve $F\lambda = d$ as outer iteration. For the last setting (MG-MG-S), we have to deal with the saddle point problem (2), which we solve using the BPCG method. The building blocks for this method are a preconditioner $\hat{\widetilde{K}}$ for $\widetilde{K}$ and $\hat{F}$ for the Schur complement F. The construction of $\hat{\widetilde{K}}$ follows the same steps as in the previous section, but we only apply a few MG V-cycles. Concerning $\hat{F}$, a good choice is the scaled Dirichlet preconditioner M_{sD}^{-1}, cf. [9].

4 Numerical Experiments

We solve the model problem (1) on a two and a three dimensional computational domain. In the two dimensional case, we use the quarter annulus divided into $32 = 8 \times 4$ patches, as illustrated in Figure 2(a). The three dimensional domain is the twisted quarter annulus, decomposed into $128 = 4 \times 4 \times 8$ patches as presented in Figure 2(b).



(a) Quarter annulus

(b) Twisted quarter annulus

Fig. 2: Illustration of the two and three dimensional computational domain.

As strongly enforced continuity conditions, we have chosen the continuity of the vertex values and the edge averages for the two dimensional example, and the continuity of the edge averages for the three dimensional example.

For the examples with polynomial degree $p = 2$, we use a standard MG method based on a hierarchy of nested grids keeping p fixed and use a standard Gauss Seidel (GS) smoother. For the examples with higher polynomial degree ($p = 4$ or 7), we have used $p = 1$ on all grid levels but the finest grid. This does not yield nested spaces. Thus, we cannot use the canonical embedding and restriction. Instead, we use L^2-projections to realize them. On the finest grid, we use a MG smoother suitable for high-order IgA, namely a variant of the subspace-corrected mass smoother proposed and analyzed in [6]. For this smoother, it was shown that a resulting MG method is robust with respect to both the grid size and the polynomial degree. However, for $p = 1$ or 2, standard approaches are more efficient. Thus, we again use this smoother only for the finest level, while for all other grid levels we use standard GS smoothers. To archive better results, we have modified the subspace-corrected mass smoother by incorporating a rank-one approximation of the geometry transformation.

For the outer CG or BPCG iteration, we use a zero initial guess, and the reduction of the initial residual by the factor 10^{-6} as stopping criterion. The local problems related to the calculation of the S-orthogonal basis are solved up to a tolerance of 10^{-12}. In case of the (MG-MG) version, the local Neumann problems (7) in W_Δ are solved up to a relative error of 10^{-10}. The number of MG cycles in the preconditioner is fixed. For the local Dirichlet problems in the scaled Dirichlet preconditioner, we use 2 V-cycles. The local Neumann problems, which appear in the preconditioner of the (MG-MG-S) version, are approximately solved by 3 V-cycles. In the following, we report on the number of CG iterations to solve (4) and BP-CG iterations for (2) and the total time, which includes the assembling, the IETI-DP setup and solving phase.

$p = 2$	**D-D**		**MG-D**		**MG-MG**		**MG-MG-S**		$p = 7$	**D-D**		**MG-D**		**MG-MG**		**MG-MG-S**	
Dofs	It.	Time	It.	Time	It.	Time	It.	Time	Dofs	It.	Time	It.	Time	It.	Time	It.	Time
134421	9	9.5	9	7.8	9	12.5	14	14.4	45753	10	25.7	10	26.7	10	56.7	14	53.5
530965	10	45.4	10	37.0	10	54.4	15	90.1	155961	11	108	11	110	11	225	15	211
2110485	11	224	11	172	11	272	16	568	572985	12	498	12	495	12	1048	17	1013
8415253	11	1005	11	762	11	1181	15	3394	2193465	13	2384	13	2265	14	4427	18	4344
33607701	OoM		OoM		13	5070	OoM		8580153	OoM		OoM		15	18484	20	19958

Table 1: Number of outer iterations and timings for the four different formulations using the quarter annulus, see Figure 2(a). GS smoother is used for $p = 2$ and p-robust subspace corrected mass smoother for $p = 7$.

$p = 2$	**D-D**		**MG-D**		**MG-MG**		**MG-MG-S**		$p = 4$	**D-D**		**MG-D**		**MG-MG**		**MG-MG-S**	
Dofs	It.	Time	It.	Time	It.	Time	It.	Time	Dofs	It.	Time	It.	Time	It.	Time	It.	Time
14079	11	2.6	11	2.5	11	7.6	25	7.2	40095	13	29.5	13	32.8	13	112	23	104
86975	12	19.3	12	19.1	12	59.1	26	59.1	160863	15	234	15	254	15	659	28	633
606015	14	213	14	197	14	484	30	616	849375	16	2237	17	2356	17	5403	32	5298
4513343	OoM		16	2764	16	5244	35	11657	5390559	OoM		OoM		19	45243	37	52831

Table 2: Number of outer iterations and timings for the four different formulations using the twisted quarter annulus, see Figure 2(a). GS smoother is used for $p = 2$ and p-robust subspace corrected mass smoother for $p = 4$.

The algorithm is realized with the open source C++ library G+Smo [12], which uses the linear algebra facilities of the Eigen library [4]. We utilize the PARDISO 5.0.0 Solver [11] for performing the LU factorizations.[1]

In Table 1, we summarize the results for the two dimensional domain for $p = 2$ and 7. We observe that replacing the direct solver in the preconditioner with two MG V-cycles does not change the number of outer iterations. Moreover, going from the Schur complement to the saddle point formulation and using BPCG there, leads only to a minor increase in the number of outer iterations. In all cases, the logarithmic dependence of the condition number on h is preserved. The advantage of the formulation using only MG, especially (MG-MG), is its smaller memory footprint, therefore, the possibility of solving larger systems. However, the setting with the best performance is (MG-D). Concluding, for small polynomial degrees and using the GS smoother, (MG-MG) gives reasonable trade off between performance and memory usage and for larger polynomial degrees, this setting can be still recommended if memory consumption is an issue.

In the case $p = 2$, for the inner iterations, we have observed that the CG needed on average 8 iterations to compute $\widetilde{g}$, the calculation of the S-orthogonal basis needed on average 14 iterations and the solution of (7) required on average 10 iterations. For the second case, $p = 7$, we needed 9 iterations to compute $\widetilde{g}$, 13 iterations for the calculation of the S-orthogonal basis and 10 iterations for the solutions of (7). Here and in what follows, we have taken the average over the patches, the individual levels and the individual steps of the outer iteration. We mention that the number of inner iterations was only varying slightly.

In Table 2, we summarize the results for the three dimensional domain and for $p = 2$ and 4. We observe that replacing the direct solver in the preconditioner with two MG V-cycles does not change the number of outer iterations. We further observe that the results behave similar to the one of the two dimensional case. However, the number of iterations almost doubled when using BPCG for (MG-MG-S). In all cases, the logarithmic dependence of

[1] Our code is compiled with the `gcc 4.8.3` compiler with optimization flag `-O3`. The results are obtain on the RADON1 cluster at Linz. We use a single core of a node, equipped with 2x Xeon E5-2630v3 “Haswell” CPU (8 Cores, 2.4Ghz, 20MB Cache) and 128 GB RAM.

the condition number on h is preserved. The advantage of the formulation using only MG, especially (MG-MG), is its smaller memory footprint, therefore the possibility of solving larger systems. The best performance is obtained sometimes by (D-D) and sometimes by (MG-D), where both approaches are comparable.

Concerning the inner iterations, for $p = 2$, we need on average 15 CG iterations to compute $\widetilde{g}$, 22 CG iterations to build up each S-orthogonal basis function, and 18 CG iterations to solve (7). In the case of $p = 4$, we needed on average only 10 iterations to compute $\widetilde{g}$, 14 iterations for the construction of the S-orthogonal basis functions, and 11 iterations for solving (7).

Acknowledgments

This work was supported by the Austrian Science Fund (FWF) under the grant W1214, project DK4. This support is gratefully acknowledged.

References

1. L. Beirão da Veiga, A. Buffa, G. Sangalli, and R. Vázquez. Mathematical analysis of variational isogeometric methods. *Acta Numerica*, 23:157–287, 2014.
2. L. Beirão da Veiga, D. Cho, L. F. Pavarino, and S. Scacchi. BDDC preconditioners for isogeometric analysis. *Math. Models Methods Appl. Sci.*, 23(6):1099–1142, 2013.
3. J. H. Bramble and J. E. Pasciak. A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems. *Mathematics of Computation*, 50(181):1–17, 1988.
4. G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.
5. C. Hofer and U. Langer. Dual-primal isogeometric tearing and interconnecting solvers for multipatch dG-IgA equations. *Computer Methods in Applied Mechanics and Engineering*, 316:2 – 21, 2017.
6. C. Hofreither and S. Takacs. Robust multigrid for isogeometric analysis based on stable splittings of spline spaces. RICAM-Report 27, Johann Radon Institute for Computational and Applied Mathematics, Austrian Academy of Sciences, 2016. also available as arXiv preprint arXiv:1607.05035.
7. T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput. Methods Appl. Mech. Engrg.*, 194:4135–4195, 2005.
8. A. Klawonn, M. Lanser, and O. Rheinbach. *A Highly Scalable Implementation of Inexact Nonlinear FETI-DP Without Sparse Direct Solvers*, pages 255–264. Springer International Publishing, Cham, 2016.
9. A. Klawonn and O. Rheinbach. Inexact FETI-DP methods. *Int. J. Numer. Methods Eng.*, 69(2):284–307, 2007.
10. S. Kleiss, C. Pechstein, B. Jüttler, and S. Tomar. IETI–isogeometric tearing and interconnecting. *Computer Methods in Applied Mechanics and Engineering*, 247:201–215, 2012.
11. A. Kuzmin, M. Luisier, and O. Schenk. Fast methods for computing selected elements of the greens function in massively parallel nanoelectronic device simulations. In F. Wolf, B. Mohr, and D. Mey, editors, *Euro-Par 2013 Parallel Processing*, volume 8097 of *Lecture Notes in Computer Science*, pages 533–544. Springer Berlin Heidelberg, 2013.
12. A. Mantzaflaris, C. Hofer, S. Takacs, et al. G+Smo (Geometry plus Simulation modules) v0.8.1. http://gs.jku.at/gismo, 2015.
13. C. Pechstein. *Finite and boundary element tearing and interconnecting solvers for multiscale problems.* Berlin: Springer, 2013.
14. J. Schöberl and W. Zulehner. Symmetric indefinite preconditioners for saddle point problems with applications to pde-constrained optimization problems. *SIAM Journal on Matrix Analysis and Applications*, 29(3):752–773, 2007.
15. M. Stoll and A. Wathen. Combination preconditioning and the Bramble–Pasciak $^+$ preconditioner. *SIAM Journal on Matrix Analysis and Applications*, 30(2):582–608, 2008.
16. A. Toselli and O. B. Widlund. *Domain decomposition methods – algorithms and theory.* Berlin: Springer, 2005.