A Three-Level Extension for Fast and Robust Overlapping Schwarz (FROSch) Preconditioners with Reduced Dimensional Coarse Space

Heinlein, Alexander; Klawonn, Axel; Rheinbach, Oliver; Röver, Friederike

DOI
10.1007/978-3-030-95025-5_54

Publication date
2022

Document Version
Final published version

Published in
Domain Decomposition Methods in Science and Engineering XXVI

A Three-Level Extension for Fast and Robust Overlapping Schwarz (FROSch) Preconditioners with Reduced Dimensional Coarse Space

Alexander Heinlein, Axel Klawonn, Oliver Rheinbach, and Friederike Röver

1 Fast and Robust Overlapping Schwarz preconditioners

The Fast and Robust Overlapping Schwarz framework [7, 8], which is part of the Trilinos Software library [18], contains a parallel implementation of the *generalized Dryja–Smith–Widlund* (GDSW) preconditioner. The GDSW preconditioner is a two-level overlapping Schwarz domain decomposition preconditioner [17] with an energy minimizing coarse space [3, 4]. It is constructed based on a domain decomposition of the computational domain Ω into N nonoverlapping subdomains $\{\Omega_i\}_{i=1,\dots,N}$. These are then extended by k layers of elements, resulting in a corresponding overlapping domain decomposition $\{\Omega_i'\}_{i=1,\dots,N}$. The two-level GDSW preconditioner can then be written as

$$M_{\text{GDSW}}^{-1} = \underbrace{\Phi K_0^{-1} \Phi^T}_{\text{coarse level}} + \underbrace{\sum_{i=1}^{N} R_i^T K_i^{-1} R_i}_{\text{first level}}, \tag{1}$$

where Φ contains the coarse basis functions. Contrary to the classical approach, where the coarse basis functions are chosen as nodal finite element functions on

Alexander Heinlein
Delft University of Technology, Faculty of Electrical Engineering Mathematics & Computer Science, Delft Institute of Applied Mathematics, Mekelweg 4, 72628 CD Delft, Netherlands,
e-mail: a.heinlein@tudelft.nl

Axel Klawonn
Department Mathematik/Informatik, Universität zu Köln, Weyertal 86-90, 50923 Köln, Germany,
e-mail: axel.klawonn@uni-koeln.de
and
Center for Data and Simulation Science, University of Cologne.
https://www.cds.uni-koeln.de

Oliver Rheinbach and Friederike Röver
Institut für Numerische Mathematik und Optimierung, Technische Universität Bergakademie Freiberg, Akademiestr. 6, 09599 Freiberg, Germany,
e-mail: {oliver.rheinbach, friederike.roever}@math.tu-freiberg.de.

S. C. Brenner et al. (eds.), *Domain Decomposition Methods in Science and Engineering XXVI*, Lecture Notes in Computational Science and Engineering 145,
https://doi.org/10.1007/978-3-030-95025-5_54

a coarse triangulation, for the GDSW preconditioner, these are chosen as discrete harmonic extensions of certain interface functions Φ_Γ to the interior of each subdomain. In particular, the functions Φ_Γ are restrictions of the null space of the global Neumann matrix to the vertices, edges, and faces, which form a nonoverlapping decomposition of the domain decomposition interface. The matrix $K_0 = \Phi^T K \Phi$ is the coarse matrix and the matrices $K_i = R_i K R_i^T$, $i = 1, \ldots, N$, correspond to the overlapping subdomain problems on the first level. The local subspaces corresponding to the overlapping subdomains are denoted as $V_1, \ldots, V_N$, and the GDSW coarse space is denoted by V_0. For scalar elliptic problems, the condition number is bounded by

$$\kappa(M_{\mathrm{GDSW}}^{-1} K) \leq C \left(1 + \frac{H}{\delta}\right) \left(1 + \log\left(\frac{H}{h}\right)\right)^2, \tag{2}$$

where C is a constant independent of the finite element size h, the size H of the nonoverlapping subdomains, and the width of the overlap $\delta = kh$; see [3]. The GDSW coarse space can be constructed in an algebraic fashion, i.e., without geometric information. For a further reduction of the coarse space, the FROSch framework provides an implementation of a reduced dimensional coarse space (RGDSW) [11]. For the reduced dimensional GDSW coarse space, the basis functions are constructed from nodal interface functions. Two options are currently available in FROSch: a fully algebraic version *(Option 1)* [5, 11], where the interface values are defined through the number of adjacent vertices, or the less algebraic version *(Option 2.2)* [5, 11], where the interface values are defined through the distance to the adjacent vertices; cf. [5, 11]. In general, the two options result in different partitions of unity. The interior values of each subdomain are determined as in the classical GDSW approach.

2 Three-level extension

For a large number of subdomains, the coarse problem of the two-level (R)GDSW preconditioners may become too large to be solved by a sparse direct solver. As in the three-level BDDC methods [19], we can resolve this by applying the GDSW preconditioner recursively to the coarse problem [9, 10]. This technique can be extended to a multi-level version, as in multi-level BDDC [1, 16] (which compete with inexact FETI-DP methods [13]), multilevel Schwarz methods [14, 15], or multigrid methods. We only discuss the three-level extension in this paper.

To apply the (R)GDSW preconditioner to the coarse problem, we need to define an additional layer of decomposition. We therefore decompose the domain into non-overlapping subregions Ω_{i0} of diameter H_c, whereas each subregion is a union of subdomains. To obtain overlapping subregions Ω'_{i0}, we extend each subregion by recursively adding layers of subdomains, as we do with finite elements on the subdomain level; see Figure 1. We denote the subregion overlap by Δ. The notation on the subdomain level is kept consistent with the two-level method.

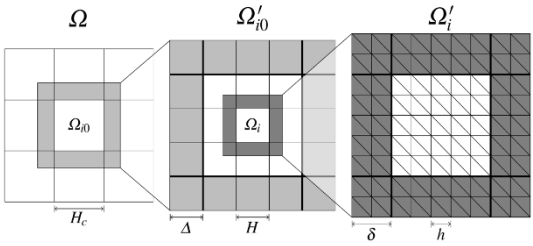

Fig. 1: Structured decomposition of an exemplary two-dimensional computational domain Ω into nonoverlapping subregions Ω_{i0}(left), a zoom into one overlapping subregion Ω'_{i0} consisting of subdomains Ω_i (middle), and a zoom into one overlapping subdomain Ω'_i (right). Each level of zoom corresponds to one level of the preconditioner; image from [9].

We define the three-level GDSW preconditioner [9, 10] by

$$M^{-1}_{\text{GDSW}-3\text{L}} = \Phi\Big(\overbrace{\Phi_0 K_{00}^{-1}\Phi_0^T}^{\text{third level}} + \overbrace{\sum_{i=1}^{N_0} R_{i0}^T K_{i0}^{-1} R_{i0}}^{\text{second level}}\Big)\Phi^T + \overbrace{\sum_{j=1}^{N} R_j^T K_j^{-1} R_j}^{\text{first level}}, \qquad (3)$$

where the first level and the matrices Φ are defined as in the two-level method and where $K_{00} = \Phi_0^T K_0 \Phi_0$ and $K_{i0} = R_{i0} K_0 R_{i0}^T$. The restriction operators, restricting to the overlapping subregions Ω'_{i0}, are defined as $R_{i0} : V^0 \to V_i^0 := V^0(\Omega'_{i0})$ for $i = 1, ..., N_0$. The respective coarse space is denoted as V_{00} and spanned by the coarse basis functions Φ_0.

3 Implementation

The Fast and Robust Overlapping Schwarz (FROSch) framework [7, 8] is part of the package ShyLU from the Trilinos software library [18]. It contains parallel implementations of the GDSW and RGDSW preconditioners based on the Trilinos linear algebra interface Xpetra; it enables the use of both Trilinos linear packages Epetra and Tpetra. To test the three-level extension to the FROSch implementation, we considered a linear elasticity model problem on the unit cube $[0, 1]^3$ with homogenous Dirichlet boundary condition on $\partial\Omega$. We use piecewise trilinear finite elements and a structured decomposition of the computational domain. To assemble the stiffness matrix we apply the Trilinos package Galeri. Here, each process owns the same number of rows of stiffness matrix resulting in different subdomain sizes. We use a generic right-hand side vector in which each entry is set to one. If the coarse space is constructed as described in Section 1, the columns of the matrix Φ will be a generating set of the coarse space. However, for our model problem, the columns will not be linearly independent and, hence, not form a basis of the coarse space. This is because the restriction of the six-dimensional null space, consisting of

translations and linearized rotations, to an interface component may yield linearly dependent vectors. For instance, the restriction of the null space to a single vertex yields only a three dimensional space. In order to make sure that the coarse matrix K_0 is invertible, we have to deal with this in our implementation. In particular, before building K_0, we replace linearly dependent coarse functions by null vectors until all other basis functions are linearly independent; in order to identify linear dependencies, we perform local orthonormalization using LAPACK's SGEQRF routine for computing a QR factorization using Householder transformations. This procedure yields zero rows and columns in K_0. Therefore, in order to make K_0 invertible, we finally replace those rows and columns by the corresponding unit vectors, leaving a one on the diagonal and zeros otherwise. This also has the nice side effect that the size of the coarse matrix is always the number of interface components times the dimension of the null space. The coarse level is decomposed into subregions in an unstructured way using the Parallel Hypergraph and Graph Partitioning (PGH) from the Trilinos package Zoltan2 [20]; see also [12]. As a Krylov iteration method, we apply the preconditioned conjugate gradient method (PCG) provided by the Trilinos package Belos (BelosPseudoBlockCG). The implementation offers a condition number estimate using the tridiagonal matrix constructed in the Lanczos process. We use the relative stopping criterion $\|r^k\|_2/\|r^0\|_2 \leq 10^{-6}$, where r^k is the residual in the k-th iteration step and r^0 is the initial residual. For all tests, we chose $20^3 * 3$ rows of the stiffness matrix for each process and approximately 8^3 subdomains per nonoverlapping subregion. The overlap is obtained by extending each subdomain by one layer of elements and by extending each subregion by one layer of subdomains. We performed all numerical tests on the GCS supercomputer SuperMUC-NG. The INTEL 19.0 compiler is used. The sparse linear subproblems arising in the preconditioner are solved using the sparse direct linear solver PardisoMKL [2].

4 Weak parallel scalability results for the three-Level extension

In this section, we focus on weak parallel scalability results for the three-level GDSW preconditioner with a reduced dimensional coarse space. We always use *Option 1* to construct the coarse basis functions. In Trilinos the data is distributed among the processes via the map object. We use a repeatedly decomposed map to determine the interface Γ. This map can be passed as an input to the FROSch framework.

For our weak parallel scalability tests, we consider three different setups to determine the interface Γ, which result in different sizes and sparsity patterns for the coarse problem; see Figure 2. We either use the *Geometric Map*, which is constructed from the structured non-overlapping domain decomposition on the first level, or the *Algebraic Map* [6], which is built algebraically from the uniquely decomposed row map of the input matrix. In particular, the interfaces and hence the vertices, edges, and faces may differ slightly for the two different maps; this effect may be more pronounced for unstructured domain decompositions.

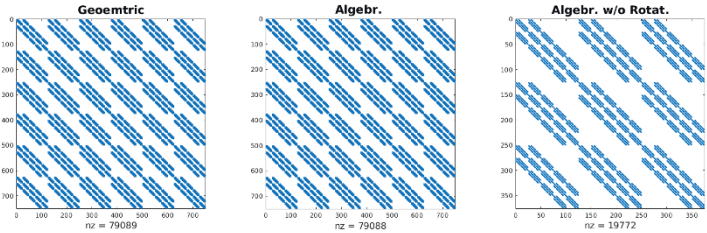

Fig. 2: Sparsity of the coarse matrix K_0 for our linear elasticity model problem in three dimensions with 216 subdomains; using the *Geometric Map* (left) and the *Algebraic Map*, with rotations (middle) and without linearized rotations (right). The subdomain size is chosen such that each process of the uniquely decomposed map owns 30^3 nodes.

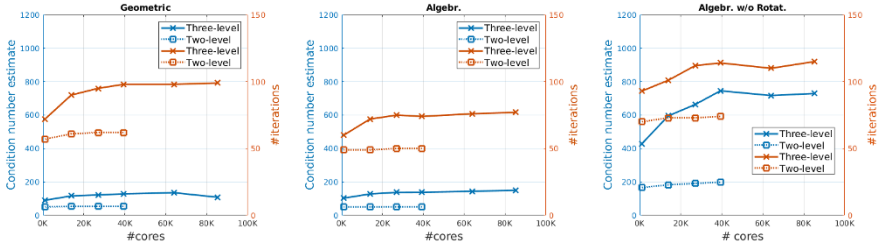

Fig. 3: Weak numerical scalability for the three- and two-level method with a reduced dimensional coarse space; see Table 1 for the data; using the *Geometric Map* and the *Algebraic Map* with and without rotations.

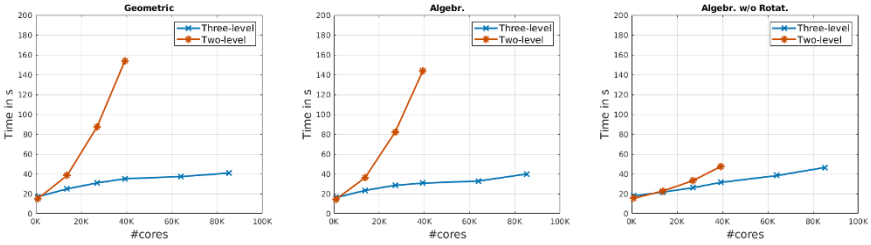

Fig. 4: Weak parallel scalability for the three- and two-level method with a reduced dimensional coarse space; see Table 1 for the data.

When using the *Algebraic Map*, we also consider the case where the rotations are neglected (*Algebr. w/o Rotat.*). In Figure 2, we only see minor differences in the sparsity pattern of K_0 using the *Geometric* and the *Algebraic Map*. For higher numbers of subdomains, the differences between these two approaches will be more visible: for our largest test case with 85 184 subdomains, we have 539 460 as a maximum nonzero entries per core in K_0 for the *Geometric Map*; this compares to 578 340 maximum nonzeros per core for the *Algebraic Map*. For all input maps, the

# Subd.	# Subr.	# Dofs	Map	Two-level $\kappa(M^{-1}K)$	Two-level Iter	Two-level Solver Time	Three-level $\kappa(M^{-1}K)$	Three-level iter	Three-level Solver Time
1 000	4	$2.4 \cdot 10^7$	Geom.	51.45	57	15.11s	90.46	72	16.99s
			Algebr.	50.73	49	14.54s	103.02	60	16.12s
			Algebr. w/o Rotat.	166.68	70	15.48s	429.05	93	17.91s
13 824	27	$3.3 \cdot 10^8$	Geom.	53.61	61	38.40s	116.19	90	24.89s
			Algebr.	51.08	49	36.39s	127.91	72	23.38s
			Algebr. w/o Rotat.	182.46	73	22.75s	594.97	101	21.58s
27 000	64	$6.5 \cdot 10^8$	Geom.	53.77	62	87.28s	122.18	95	30.87s
			Algebr.	51.12	50	82.01s	137.42	75	28.46s
			Algebr. w/o Rotat.	191.12	73	33.17s	663.44	112	26.02s
39 304	125	$9.4 \cdot 10^8$	Geom.	53.82	62	153.88s	128.39	98	35.12s
			Algebr.	51.12	50	144.01s	137.96	74	30.63s
			Algebr. w/o Rotat.	198.05	74	47.48s	745.26	114	31.40s
64 000	216	$1.5 \cdot 10^9$	Geom.	-	-	-	135.58	98	37.29s
			Algebr.	-	-	-	143.87	76	32.81s
			Algebr. w/o Rotat.	-	-	-	717.06	110	38.24s
85 184	275	$2.0 \cdot 10^9$	Geom.	-	-	-	108.49	99	40.80s
			Algebr.	-	-	-	150.37	77	39.87s
			Algebr. w/o Rotat.	-	-	-	729.14	115	46.45s

Table 1: Data corresponding to Figure 3 and 4. By *Iter*, we denote the number of PCG iterations, and κ is the condition number of the preconditioned operator. *Solver Time* is the time to build the preconditioner and to perform the Krylov iterations; see also Figure 3 and 4. The subdomain size is chosen such that each process of the uniquely decomposed map owns 20^3 nodes. We have $H_c/H \approx 8$. One layer of finite elements respectively one layer of subdomains is chosen as the overlap for each level.

two- and the three-level method are numerically scalable, whereas the *Algebraic Map without Rotations* yields the highest iteration counts and condition number estimates; cf. Figure 3 and Table 1. Replacing the direct solver for the coarse problem (used in the two-level method) by the application of the RGDSW preconditioner for the three-level method generally results in higher condition number estimates and iteration counts.

However, the three-level extension of the FROSch framework shows a better parallel weak scalability than the two-level method; cf Figure 4 and Table 1. The *Solver Time* is the time to build the preconditioner and to perform the Krylov iterations. The time includes the factorization and forward backward substitution for the sparse direct solvers. For the three-level method the time for the unstructured decomposition of the coarse problem is also included. For all test settings, the three-level method is faster for 13 824 and more cores. Moreover, at 39 304 cores the three-level method is faster by more than a factor of four: Using the three-level method, we obtain a *Solver Time* of 35.12 *s* using the *Geometric Map* and 30.63 *s* using *Algebraic Map*. This compares to a *Solver Time* of 153.88 *s* for the *Geometric Map* and 144.01 *s* for the *Algebraic Map* in the two-level method. Using *Algebraic Map without Rotation* results in a smaller coarse problem, making the two-level methods more competitive. Here, the three-level method (*Solver Time* 31.40 *s*) is still faster by a factor of 1.5 than the two-level method (*Solver Time* 47.48 *s*). As the results are clear, we did not perform tests beyond the 39 304 cores for the two-level method. To illustrate the strong influence of the size of the coarse problem on the

preconditioner time, we consider the test case of 39 304 cores in Table 2. For this test case, the solution of coarse problem K_0 in the *Geometric Map* setup takes 78% (120.19 *s*) of the total *Solver Time* (153.88 *s*).

# Subd.	# Subr.	# Dofs	Map	Two-level method Size K_0	K_0 Solve Time	Three-level method Size K_{00}	K_{00} Solve Time
1 000	4	$2.4 \cdot 10^7$	Geom.	4 374	0.24s	6	<1e-5s
			Algebr.	4 374	0.22s	6	<1e-5s
			Algebr. w/o Rotat.	2 184	0.08s	3	<1e-5s
13 824	27	$3.3 \cdot 10^8$	Geom.	73 002	12.03s	366	0.01s
			Algebr.	73 002	12.04s	366	0.01s
			Algebr. w/o Rotat.	36 501	2.02s	174	0.003s
27 000	64	$6.5 \cdot 10^8$	Geom.	146 334	59.38s	1 056	0.08s
			Algebr.	146 334	45.75s	1 116	0.08s
			Algebr. w/o Rotat.	73 167	11.69s	546	0.04s
39 304	125	$9.4 \cdot 10^8$	Geom.	215 622	120.19s	2 508	0.29s
			Algebr.	215 622	114.06s	2 556	0.25s
			Algebr. w/o Rotat.	107 811	22.14s	1 290	0.11s
64 000	216	$1.5 \cdot 10^9$	Geom.	355 914	-	4 980	0.81s
			Algebr.	355 914	-	4 938	0.63s
			Algebr. w/o Rotat.	177 957	-	2 319	0.21s
85 184	275	$2.0 \cdot 10^9$	Geom.	477 042	-	6 432	0.63s
			Algebr.	477 042	-	6 660	0.72s
			Algebr. w/o Rotat.	238 521	-	3 222	0.16s

Table 2: Cost for solving the problem on the coarsest level. *Solve Coarse Problem Time* includes the time of the factorization of the problem as well as the forward and backward substitution in the Krylov iterations.

For this test case, the solution of coarse problem K_0 in the *Geometric Map* setup takes 78% (120.19 *s*) of the total *Solver Time* (153.88 *s*). This time compares to less than a second (0.29 *s*) to solve the coarse problem corresponding to K_{00} in the three-level method. Similar results are obtained for the *Algebraic Map* where 114.06 *s* for the two-level method compare with 0.25 *s* for the three-level method. For the *Algebraic Map without Rotations* the size of the coarse problem is reduced by a factor of two; cf. Table 2. Therefore, the cost of the coarse problem reduces to 22.14 *s* for the two-level method, which compares to 0.11 *s* for the three-level method. Although the *Algebraic Map* has the largest coarse problem size (see Table 2) this is consistently the fastest setup of the three-level method. The stronger connectivity given by this coarse problem improves the iteration count and therefore decreases the *Solver Time*. Resulting in the highest number of iterations (cf. Table 1) the *Algebraic Map without rotations* is the slowest test case.

Acknowledgements The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for providing computing time on the GCS Supercomputer SuperMUC-NG at Leibniz Supercomputing Centre (www.lrz.de).

The author acknowledge computing time on the Compute Cluster of the Fakultät für Mathematik und Informatik of Technische Universität Freiberg DFG project number 397252409), operated by the university computing center (URZ).

The third and fourth author would like to acknowledge funding by the Deutsche Forschungsgemeinschaft (DFG) under the DFG project number 441509557 within the DFG SPP 2256.

References

1. S. Badia, A.F. Martín and J. Principe. Multilevel balancing domain decomposition at extreme scales. *SIAM J. Sci. Comput.* **38**(1), C22–C52 (2016).
2. M. Bollhöfer, O. Schenk, R. Janalik, S. Hamm and K. Gullapalli. State-of-the-art sparse direct solvers. In: A. Grama and A.H. Sameh, editors, *Parallel Algorithms in Computational Science and Engineering*, pp. 3–33, Springer, Cham (2020).
3. C.R. Dohrmann, A. Klawonn and O.B. Widlund. Domain decomposition for less regular subdomains: overlapping Schwarz in two dimensions. *SIAM J. Numer. Anal.* **46**(4), 2153–2168 (2008).
4. C.R. Dohrmann, A. Klawonn and O.B. Widlund. A family of energy minimizing coarse spaces for overlapping Schwarz preconditioners. In: *Domain decomposition methods in science and engineering XVII*, volume 60 of LNCSE, pp. 247–254, Springer, Berlin (2008).
5. C.R. Dohrmann and O.B. Widlund. On the design of small coarse spaces for domain decomposition algorithms. *SIAM J. Sci. Comput.* **39**(4), A1466–A1488 (2017).
6. A. Heinlein, C. Hochmuth and A. Klawonn. Fully algebraic two-level overlapping Schwarz preconditioners for elasticity problems. In: F.J. Vermolen and C. Vuik, editors, *Numerical Mathematics and Advanced Applications ENUMATH 2019*, pp. 531–539, Springer, Cham (2021).
7. A. Heinlein, A. Klawonn, S. Rajamanickam and O. Rheinbach. FROSch: A fast and robust overlapping Schwarz domain decomposition preconditioner based on Xpetra in Trilinos. In: *Domain Decomposition Methods in Science and Engineering XXV*, pp. 176–184, Springer, Cham (2020).
8. A. Heinlein, A. Klawonn and O. Rheinbach. A parallel implementation of a two-level overlapping Schwarz method with energy-minimizing coarse space based on Trilinos. *SIAM J. Sci. Comput.* **38**(6), C713–C747 (2016).
9. A. Heinlein, A. Klawonn, O. Rheinbach and F. Röver. A three-level extension of the GDSW overlapping Schwarz preconditioner in two dimensions. In: *Advanced Finite Element Methods with Applications: Selected Papers from the 30th Chemnitz Finite Element Symposium 2017*, pp. 187–204, Springer, Cham (2019).
10. A. Heinlein, A. Klawonn, O. Rheinbach and F. Röver. A three-level extension of the GDSW overlapping Schwarz preconditioner in three dimensions. In: *Domain Decomposition Methods in Science and Engineering XXV*, pp. 185–192, Springer, Cham (2020).
11. A. Heinlein, A. Klawonn, O. Rheinbach and O.B. Widlund. Improving the parallel performance of overlapping Schwarz methods by using a smaller energy minimizing coarse space. In: *Domain Decomposition Methods in Science and Engineering XXIV*, pp. 383–392, Springer, Cham (2018).
12. A. Heinlein, O. Rheinbach and F. Röver. Choosing the subregions in three-level FROSch preconditioners. In: *14th WCCM-ECCOMAS Congress 2020*, volume 700 (2021).
13. A. Klawonn and O. Rheinbach. Inexact FETI-DP methods. *IJNME* **69**(2), 284–307 (2007).
14. F. Kong and X.-C. Cai. A highly scalable multilevel Schwarz method with boundary geometry preserving coarse spaces for 3D elasticity problems on domains with complex geometry. *SIAM J. Sci. Comput.* **38**(2), C73–C95 (2016).
15. S. Scacchi. A hybrid multilevel Schwarz method for the bidomain model. *Comput. Methods Appl. Mech. Engrg.* **197**(45-48), 4051–4061 (2008).

16. J. Šístek, B. Sousedík, J. Mandel and P. Burda. Parallel implementation of multilevel BDDC. In: A. Cangiani, R. Davidchack, E. Georgoulis, A. Gorban, J. Levesley and M. Tretyakov, editors, *Numerical mathematics and advanced applications 2011*, pp. 681–689, Springer, Berlin, Heidelberg (2013).
17. A. Toselli and O. Widlund. *Domain decomposition methods – algorithms and theory*. Volume 34 of Springer Series in Computational Mathematics. Springer-Verlag, Berlin (2005).
18. Trilinos public git repository. Web, 2018. https://github.com/trilinos/trilinos.
19. X. Tu. Three-level BDDC in two dimensions. *IJNME* **69**, 33–59 (2007).
20. The Zoltan2 Project Team. https://trilinos.github.io/zoltan2.html.