Application of Multilevel BDDC to the Problem of Pressure in Simulations of Incompressible Flow

Martin Hanek and Jakub Šístek

1 Introduction

We deal with the numerical solution of problems of incompressible flows and investigate the applicability of the Balancing Domain Decomposition by Constraints (BDDC) by [2] for solving the arising linear systems. In [5], we extended the multilevel version of BDDC ([12, 7]) to nonsymmetric problems arising from steady problems described by the Navier-Stokes equations. In the present contribution, we are interested in solving unsteady problems, for which we employ the pressure-correction operator-splitting scheme (see e.g. the overview paper by [4]). It presents a very efficient approach for solving the problem by transforming the coupled Navier-Stokes equations into a sequence of a scalar convection-diffusion problem for each velocity component, a Poisson problem for pressure (corrector), and an L_2-projection problem in each time step.

In [10], we studied efficient solution techniques for the arising systems based on Krylov subspace methods with one-level domain decomposition (DD) preconditioners from the `PETSc` library. A conclusion of the study was that while these relatively simple preconditioners work well for the nonsymmetric problems for velocities and the L_2-projection problem, the known dependence of one-level DD methods on the number of subdomains made the pressure Poisson problem increasingly difficult for a solution with growing problem size, eventually becoming the bottleneck of the simulations.

Martin Hanek
Institute of Mathematics of the Czech Academy of Sciences, Žitná 25, Prague, Czech Republic
Czech Technical University in Prague, Technická 4, Prague, Czech Republic e-mail: `martin.hanek@fs.cvut.cz`

Jakub Šístek
Institute of Mathematics of the Czech Academy of Sciences, Žitná 25, Prague, Czech Republic
e-mail: `sistek@math.cas.cz`

In this paper, we want to investigate the applicability of several variants of BDDC for the problem of pressure corrector. As long as the mesh is not changed, the matrix and the preconditioner are set up just once for all time steps. This makes it interesting to use variants of BDDC with more expensive setup, saving the number of iterations in each time step, such as the BDDC method with the adaptive selection of constraints by [8], and its combination with the multilevel extension [11] implemented in our `BDDCML` library.

Another strategy worth investigating for sequences of algebraic problems is a recycling of the Krylov subspace across time steps, proposed e.g. by [3]. It has been shown in the literature that if the differences of the successive right-hand sides are not large, after expanding the new right-hand side in the pre-existing Krylov basis, one may require only very few or even no additional iterations for convergence to the full accuracy. Hence, it is another aim of this paper to investigate the benefits of the approach by [3] to the present problem.

2 The Pressure-Correction Method

We consider a domain $\Omega \subset \mathbb{R}^3$ with its boundary Γ consisting of three disjoint parts Γ_S, Γ_∞, and Γ_O, $\Gamma = \Gamma_S \cup \Gamma_\infty \cup \Gamma_O$. Part Γ_S is the interface between fluid and the rigid body, Γ_∞ is the inflow free-stream boundary, and Γ_O is the outflow boundary. The flow is governed by the Navier-Stokes equations of an incompressible viscous fluid,

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p = \boldsymbol{0} \quad &\text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} = 0 \quad &\text{in } \Omega, \end{aligned} \tag{1}$$

where $\boldsymbol{u}$ is the velocity vector of the fluid, t denotes time, ν is the kinematic viscosity of the fluid and p is the kinematic pressure. System (1) is complemented by the following initial and boundary conditions: $\boldsymbol{u}(t = 0, \boldsymbol{x}) = \boldsymbol{0}$ in Ω, $\boldsymbol{u}(t, \boldsymbol{x}) = \boldsymbol{u}_\infty$ on Γ_∞, $\boldsymbol{u}(t, \boldsymbol{x}) = \boldsymbol{0}$ on Γ_S, and $-\nu(\nabla \boldsymbol{u})\boldsymbol{n} + p\boldsymbol{n} = \boldsymbol{0}$ on Γ_O, with $\boldsymbol{n}$ being the unit outer normal vector of Γ.

System (1) can be efficiently solved with a pressure-correction method. In particular, we use the incremental pressure-correction method in the *rotational form* discussed by [4]. Details of our implementation can be found in [10].

In this approach, we first define the pressure increment (corrector) $\psi^{n+1} = p^{n+1} - p^n + \nu \nabla \cdot \boldsymbol{u}^{n+1}$. In order to compute the velocity and pressure fields $(\boldsymbol{u}^{n+1}, p^{n+1})$ at time t^{n+1}, three subproblems are subsequently solved.

1. The velocity field $\boldsymbol{u}^{n+1}$ is obtained by solving the convection-diffusion problem for each component of velocity

$$\frac{1}{\Delta t}\boldsymbol{u}^{n+1} + (\boldsymbol{u}^n \cdot \nabla)\boldsymbol{u}^{n+1} - \nu \Delta \boldsymbol{u}^{n+1} = \frac{1}{\Delta t}\boldsymbol{u}^n - \nabla(p^n + \psi^n) \quad \text{in } \Omega \tag{2}$$

for $\boldsymbol{u}^{n+1} = \boldsymbol{u}_\infty$ on Γ_∞, $\boldsymbol{u}^{n+1} = \boldsymbol{0}$ on Γ_S, and $\nu(\nabla \boldsymbol{u}^{n+1})\boldsymbol{n} = p^n \boldsymbol{n}$ on Γ_O.

2. Next, the pressure corrector ψ^{n+1} is obtained by solving the Poisson problem

$$-\Delta\psi^{n+1} = -\frac{1}{\Delta t}\nabla\cdot\boldsymbol{u}^{n+1} \quad \text{in } \Omega \tag{3}$$

for $\frac{\partial\psi^{n+1}}{\partial\boldsymbol{n}} = 0$ on $\Gamma_\infty \cup \Gamma_S$ and $\psi^{n+1} = 0$ on Γ_O.

3. Finally, the pressure field p^{n+1} is updated with

$$p^{n+1} = p^n + \psi^{n+1} - \nu\nabla\cdot\boldsymbol{u}^{n+1}. \tag{4}$$

Problems (2), (3), and (4) are solved by the finite element method (FEM) using Taylor-Hood $Q_2 - Q_1$ hexahedral elements. In the resulting finite element mesh, there are $n_{\boldsymbol{u}}$ nodes with velocity unknowns and n_p nodes with pressure unknowns, with the ratio $n_{\boldsymbol{u}}/n_p$ being approximately 8.

For solving the algebraic problems arising from (2) and (4), we use the methods identified as optimal by [10]. In particular, the Generalized Minimal Residual method (GMRES) is used for solving problem (2), and the Conjugate Gradient (CG) method is used for problem (4). Block Jacobi preconditioner using ILU(0) on subdomains is used for both problems.

The main focus of this study is a scalable solution of the Poisson problem for pressure corrector (3). We apply different one-level domain decomposition preconditioners from the `PETSc`[1] library and compare them with several settings of the BDDC method from the `BDDCML`[2] library. Each preconditioner is combined with the CG method.

Problem (3) translates to an algebraic system with a discrete Laplacian matrix of size $n_p \times n_p$ which is symmetric and positive definite for $\Gamma_O \neq \emptyset$, i.e. a nonempty part with 'do-nothing' boundary condition. This is a well-studied case from the point of view of DD methods, which are very suitable solvers for this task.

For a fixed mesh, only the right-hand side of (3) differs in the sequence for the subsequent time steps. Hence, this problem offers large room for reusing information across all time steps. For example, one may afford a preconditioner with a more expensive setup if this leads to a lower number of iterations as long as each iteration does not get much more expensive. This is our motivation for experimenting also with the adaptive selection of constraints for BDDC.

3 Numerical results

We evaluate the strategies for solving (3) on the case of the flow past a sphere at Reynolds number 300. In our simulations, we consider two sizes of the problem mesh with the same geometry (see Fig. 1). The sphere diameter is 1 m, and the solution domain is a cylinder with the radius of 6 m and the length of 25 m. The

[1] `https://www.mcs.anl.gov/petsc` (version 3.10.4)

[2] `https://users.math.cas.cz/~sistek/software/bddcml.html` (version 2.6)

centre of the sphere lies on the cylinder axis and 5 m from the front wall of the cylinder. Far-field velocity of the fluid is $\boldsymbol{u}_\infty = (1, 0, 0)^\mathsf{T}$ m s^{-1}, and the kinematic viscosity is $\nu = 0.00333$ m^{2}s^{-1}, so that the Reynolds number defined as $\mathrm{Re} = \frac{|\boldsymbol{u}_\infty|\, d}{\nu}$ is equal to 300. The external boundary of the cylinder is considered as Γ_∞ except the rear face, which represents Γ_O with 'do-nothing' boundary condition. Zero Dirichlet boundary condition is prescribed on the surface of the sphere Γ_S.

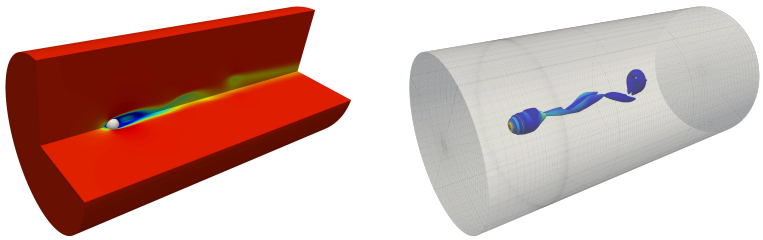

Fig. 1: Computational domain with meshing corresponding to Mesh 1, Reynolds number 300. Velocity magnitude (left) and vortical structures illustrated by isosurfaces of the average corotation ([6]) coloured by the magnitude of vorticity (right).

The computational Mesh 1 consists of 1.8 million unknowns for velocity in each component and 225 thousands for pressure, and it corresponds to the mesh used in [9]. Mesh 2 is obtained by doubling the number of elements in each direction. Hence, it has approx. 15 million unknowns for each velocity component and 1.9 millions for pressure. The meshes were created in the `Gmsh`[3] generator and divided into 16 and 128 subdomains, respectively, by the `METIS`[4] graph partitioner to maintain approximately the same size of subdomain problems (approx. 15 thousand unknowns per subdomain). The problems on Mesh 1 and Mesh 2 were solved using 16 and 128 CPU cores of the *Salomon* supercomputer at the IT4Innovations National Supercomputing Center in Ostrava, Czech Republic. The computational nodes of Salomon are equipped with two 12-core Intel Xeon E5-2680v3 2.5 GHz processors and 128 GB RAM.

This kind of simulations is usually performed for thousands of time steps. We formally employ the non-dimensional time $t' = \frac{t\,|\boldsymbol{u}_\infty|}{d}$, although for our setting of $|\boldsymbol{u}_\infty| = 1$ and $d = 1$, the values are the same as for the physical time t.

In particular, the simulation of 200 s on Mesh 1 performed with time-step size $\Delta t = 0.05$ results in 4000 time steps, while the simulation on Mesh 2 with time-step size $\Delta t = 0.025$ results in 8000 time steps. The different values of the time steps are motivated by an approximate preservation of the Courant number $\frac{|\boldsymbol{u}_\infty|\,\Delta t}{h}$ coupling the resolution in time and space.

Since our aim is to test the behaviour for different preconditioners, we compute only 30 time steps, and we report and compare the numbers of linear iterations and

[3] `https://gmsh.info/`

[4] `http://glaros.dtc.umn.edu/gkhome/metis/metis/overview`

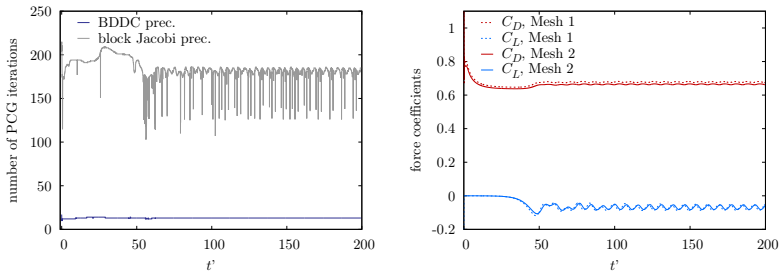

Fig. 2: Number of linear iterations in each time step (left), and values of drag (C_D) and lift (C_L) force coefficients during the whole simulation (right). Results for Mesh 1 are by [9] and for Mesh 2 from current simulations.

times for all time steps excluding the first time step. As you can see in the left part of Fig. 2, the number of iterations stays almost constant during the whole simulation for the BDDC method and within the same range for the block Jacobi method. This justifies using the first 30 iterations for our comparisons. We also compare the drag and lift force coefficients acting on the sphere with the results from [9] in Fig. 2. We have got a good agreement for the two resolutions.

In the first iteration, the setup of the preconditioner and the factorization of interior blocks of subdomain matrices are included. These operations are performed just once for all time steps. Hence, the number of linear iterations and the time of the whole solve (setup and iterations) is reported separately.

method	#its. min–max(avg.)	t./step [s]	#its. step 1	t. step 1 [s]	est. sim. [s]
block Jacobi + ILU(0)	63-169(108.3)	**0.16**	167	**0.25**	**640**
block Jacobi + ILU(1)	46-131(82.7)	0.25	130	0.37	1000
block Jacobi + ILU(2)	44-119(76.0)	0.46	118	0.61	1840
ASM–1 + ILU(0)	102-216(170.2)	0.36	209	0.43	1440
ASM–1 + ILU(1)	81-146(120.5)	0.48	140	0.54	1920
ASM–2 + ILU(0)	103-237(184.4)	0.48	230	0.57	1920
ASM–2 + ILU(1)	63-156(122.4)	0.60	148	0.67	2400
3-l. ad. BDDC + diag.	8-10(**9.6**)	0.32	**14**	75.74	1355
3-l. BDDC + diag.	10-12(11.5)	0.34	21	1.51	1361

Table 1: Mesh 1: Comparison of the number of linear iterations (minimum–maximum(average) across all time steps), average time for solving one time step, values for the first step, and estimated time of all time steps computed as the average time per step × 4000 + time for step 1. Here 'diag.' means scaling by diagonal entries of subdomain matrices, '2-l.' and '3-l.' stand for 2-level and 3-level variants of the BDDC method, respectively, and 'ad.' denotes the adaptive version of BDDC.

The results of our simulations are summarized in Tables 1 and 2. The tested preconditioners include block Jacobi and Additive Schwarz methods (ASM) from PETSc. For ASM, we compare one and two layers of overlap (ASM–1 and ASM–2). On subdomains, the incomplete LU factorization with different levels of allowed fill-in (ILU(0), ILU(1), and ILU(2)) is considered. As for the BDDC options, we use

method	#its. min–max(avg.)	t./step [s]	#its. step 1	t. step 1 [s]	est. sim. [s]
block Jacobi + ILU(0)	461-623(498.6)	1.15	611	1.41	9201
block Jacobi + ILU(1)	115-260(185.9)	0.71	258	**0.98**	5580
block Jacobi + ILU(2)	107-238(175.1)	1.08	236	1.39	8641
ASM–1 + ILU(0)	401-569(433.9)	1.60	557	2.05	12802
ASM–1 + ILU(1)	216-309(241.6)	1.36	300	1.64	10881
ASM–2 + ILU(0)	385-575(428.3)	1.95	550	2.43	15602
ASM–2 + ILU(1)	218-301(240.5)	1.79	294	2.14	14322
3-l. ad. BDDC + diag., $r = 0$	14-19(15.6)	0.62	**19**	157.77	5118
3-l. ad. BDDC + diag., $r = 50$	11-13(12.3)	**0.44**	**19**	132.88	**3652**
3-l. ad. BDDC + diag., $r = 100$	11-13(11.9)	0.54	**19**	159.48	4479
3-l. ad. BDDC + diag., $r = 200$	7-12(**9.9**)	0.51	**19**	160.92	4241
3-l. BDDC + diag., $r = 0$	29-42(32.8)	1.18	42	2.86	9443
3-l. BDDC + diag., $r = 50$	17-23(20.4)	0.89	40	3.14	7123
3-l. BDDC + diag., $r = 100$	17-20(18.7)	0.75	40	2.83	6003
3-l. BDDC + diag., $r = 200$	14-19(15.6)	0.81	40	2.92	6483
3-l. BDDC + arith., $r = 0$	14-19(17.1)	0.59	30	4.87	4724
3-l. BDDC + arith., $r = 50$	14-18(17)	0.55	29	4.07	4404
3-l. BDDC + arith., $r = 100$	14-17(15.9)	0.55	30	5.00	4405
3-l. BDDC + arith., $r = 200$	14-17(15.7)	0.55	30	5.04	4405
3-l. BDDC + diag., $r = 50$	17-25(22.6)	0.71	40	5.31	5685
2-l. ad. BDDC + diag., $r = 0$	14-19(15.2)	0.81	**19**	157.72	6638
2-l. ad. BDDC + diag., $r = 50$	11-13(12.5)	0.68	**19**	158.83	5599
2-l. BDDC + arith., $r = 50$	13-18(16.5)	0.80	27	4.27	6404
2-l. BDDC + diag., $r = 50$	17-13(20.6)	1.01	38	4.80	8085

Table 2: Mesh 2: Comparison of the number of linear iterations (minimum–maximum(average) across all time steps), average time for solving one time step, values for the first step, and estimated time of all time steps computed as the average time per step × 8000 + time for step 1. Here 'diag.' means scaling by diagonal entries of subdomain matrices, 'arith.' means scaling by arithmetic averaging, '2-l.' and '3-l.' stand for 2-level and 3-level variants of the BDDC method, respectively, and 'ad.' denotes the adaptive version of BDDC. Parameter r represents the maximum number of the stored Krylov basis vectors in recycling the Krylov subspaces.

several settings of the `BDDCML` library. Namely, we consider the 2- and 3-level BDDC methods, potentially with the adaptive selection of constraints for the coarse problem as in [11]. Two sequential instances of the MUMPS sparse direct solver (version 5.1.2, [1]) are used for each subdomain, namely a Cholesky LL^T decomposition of the block of unknows interior to the subdomain, and an LDL^T factorization of the saddle-point problems of BDDC (see [2] for details). In addition, a distributed memory instance of MUMPS is used for the final coarse problem.

The following coarse spaces are considered in the BDDC method. For the non-adaptive version, values at corners and arithmetic averages on each subdomain edge and face are taken as the continuous coarse degrees of freedom. In the adaptive case, a maximum of ten adaptive constraints is also considered on the faces.

We also compare results for two types of interface scaling, the standard one based on arithmetic averages (*arith*) and the one based on diagonal entries of the subdomain matrices (*diag*). In our computations, only the diagonal scaling is compatible with the adaptive BDDC method, while the arithmetic scaling gives better results for the non-adaptive version.

We also test several values of the number of stored Krylov basis vectors r in the approach to recycling the Krylov subspace by [3]. However, we observe little difference among the values of $r = 0$, $r = 50$, $r = 100$, and $r = 200$. We have chosen $r = 50$ for the other simulations with the BDDC method, which is the default for `BDDCML`. We have observed a larger improvement for reusing the solution from one time step as the starting approximation for the subsequent problem. This effect can be observed from the difference between the number of iterations in the first time step and their average number. The iterations are terminated when the relative norm of the residual gets below 10^{-6}.

An estimated cost of solving the pressure problem for all time steps (*est. sim.*) is also included in Tables 1 and 2. It is obtained as the time for the first step added to the average time per other steps multiplied by the number of time steps.

We can see that for the smaller problem, the most efficient method is the block Jacobi preconditioner with ILU(0) on subdomains, followed by the same preconditioner with ILU(1). The two configurations of the BDDC preconditioner are less efficient than these options.

However, for the larger problem, the most efficient method becomes BDDC with adaptive constraints, and also the non-adaptive 3-level BDDC method is more efficient than the one-level DD preconditioners, out of which the block Jacobi with ILU(1) requires the least time.

4 Conclusions

We have applied several variants of the BDDC method and one-level DD methods to the Poisson problem of pressure corrector within a solution of an unsteady problem of incompressible flow with two different meshes.

We have seen that while for a smaller problem, a simple one-level DD method (block Jacobi) provides the fastest solution, the adaptive BDDC method becomes advantageous for larger problems divided into more subdomains. Although the setup of the preconditioner is significantly more expensive, its price gets outweighed by the lower number of CG iterations required in each time step. In addition, recycling the Krylov subspace basis is also slightly beneficial for a reasonable size of the stored basis (50 vectors in our experiments).

The results are encouraging, and we can expect that for even larger problems divided into more subdomains, the adaptive-multilevel BDDC method will be even more beneficial. Confirming this expectation will be a subject of a future study as well as other selection strategies for a suitable recycling basis.

Acknowledgements This research was supported by the Czech Science Foundation through grant 20-01074S, by the Czech Academy of Sciences through RVO:67985840, and by the Czech Technical University in Prague through the student project *SGS19/154/OHK2/3T/12*. Computational time on the Salomon supercomputer has been provided thanks to the support of The Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing Center – LM2015070".

References

1. P.R. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures. *ACM Trans. Math. Softw.*, 45:2:1–2:26, 2019.
2. C. R. Dohrmann. A preconditioner for substructuring based on constrained energy minimization. *SIAM J. Sci. Comput.*, 25(1):246–258, 2003.
3. C. Farhat, L. Crivelli, and F. X. Roux. Extending substructure based iterative solvers to multiple load and repeated analyses. *Comput. Methods Appl. Mech. Engrg.*, 117:195–209, 1994.
4. J. L. Guermond, P. Minev, and J. Shen. An overview of projection methods for incompressible flow. *Comput. Methods Appl. Mech. Engrg.*, 195:6011–6045, 2006.
5. M. Hanek, J. Šístek, and P. Burda. Multilevel BDDC for incompressible Navier-Stokes equations. *SIAM J. Sci. Comput.*, 42(6):C359–C383, 2020.
6. Václav Kolář, Jakub Šístek, Fehmi Cirak, and Pavel Moses. Average corotation of line segments near a point and vortex identification. *AIAA J.*, 51(11):2678–2694, 2013.
7. Jan Mandel, Bedřich Sousedík, and Clark R. Dohrmann. Multispace and multilevel BDDC. *Computing*, 83(2-3):55–85, 2008.
8. Jan Mandel and Bedřich Sousedík. Adaptive selection of face coarse degrees of freedom in the BDDC and the FETI-DP iterative substructuring methods. *Comput. Methods Appl. Mech. Engrg.*, 196(8):1389–1399, 2007.
9. J. Šístek. A parallel finite element solver for unsteady incompressible Navier-Stokes equations. In D. Šimurda and T. Bodnár, editors, *Proceedings of Topical Problems of Fluid Mechanics 2015*, pages 193–198. Institute of Thermomechanics AS CR, 2015.
10. J. Šístek and F. Cirak. Parallel iterative solution of the incompressible Navier-Stokes equations with application to rotating wings. *Comput. Fluids*, 122:165–183, 2015.
11. Bedřich Sousedík, Jakub Šístek, and Jan Mandel. Adaptive-Multilevel BDDC and its parallel implementation. *Computing*, 95(12):1087–1119, 2013.
12. Xuemin Tu. Three-level BDDC in three dimensions. *SIAM J. Sci. Comput.*, 29(4):1759–1780, 2007.