The effect of irregular interfaces on the BDDC method for the Navier-Stokes equations

Martin Hanek[1], Jakub Šístek[2,3] and Pavel Burda[1]

1 Introduction

The Balancing Domain Decomposition based on Constraints (BDDC) was introduced by Dohrmann [2003] as an efficient method to solve large systems of linear equations arising from the finite element method on parallel computers. Dohrmann [2003] applied BDDC to elliptic problems, namely Poisson equation and linear elasticity. Li and Widlund [2006] extended the method to the Stokes equations. However, the approach requires a discontinuous approximation of the pressure. An attempt to apply the BDDC method in connection to a continuous approximation of the pressure was presented by Šístek et al. [2011] employing Taylor-Hood finite elements. Another construction of the BDDC preconditioner for the Stokes problem with a continuous approximation of the pressure was proposed by Li and Tu [2013].

Hanek et al. [2015] combined the approach to building the interface problem by Šístek et al. [2011] with the extension of BDDC to nonsymmetric problems from Yano [2009]. The algorithm has been applied to linear systems obtained by Picard linearisation of the Navier-Stokes equations. One step of BDDC is applied as a preconditioner for the BiCGstab method. These generalizations have been implemented to our open-source parallel multilevel BDDC solver *BDDCML* described by Sousedík et al. [2013].

The main focus of this study is an investigation of the robustness of the algorithm of Hanek et al. [2015] with respect to interface irregularities and element aspect ratios. The motivation comes from simulations of hydrostatic bearings, where very bad element aspect ratios appear. A benchmark problem of a narrowing channel is proposed in two dimensions (2D) and three dimensions (3D), and numerical results for this problem are presented.

Faculty of Mechanical Engineering, Czech Technical University in Prague, Karlovo náměstí 13, CZ - 121 35 Prague 2, Czech Republic, `martin.hanek@fs.cvut.cz`, `pavel.burda@fs.cvut.cz` · Institute of Mathematics of the Czech Academy of Sciences, Žitná 25, CZ - 115 67 Prague 1, Czech Republic, `sistek@math.cas.cz` · School of Mathematics, The University of Manchester, Manchester, M13 9PL, United Kingdom

2 BDDC for Navier-Stokes equations

In this section, we briefly recall our approach to using BDDC for steady Navier-Stokes problems. Details of the method can be found in Hanek et al. [2015].

A steady flow of an incompressible fluid in a two-dimensional (2-D) or three-dimensional (3-D) domain Ω is governed by the Navier-Stokes equations without body forces

$$(\mathbf{u}\cdot\nabla)\mathbf{u} - \nu\Delta\mathbf{u} + \nabla p = \mathbf{0} \quad \text{in } \Omega, \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \quad \text{in } \Omega, \tag{2}$$

where $\mathbf{u}$ is an unknown velocity vector, p is an unknown pressure normalised by (constant) density, and ν is a given kinematic viscosity. In addition, the usual 'no-slip' boundary conditions $\mathbf{u} = \mathbf{g}$ on Γ_D and 'do-nothing' boundary conditions $-\nu(\nabla\mathbf{u})\mathbf{n} + p\mathbf{n} = 0$ on Γ_N are considered.

Applying the finite element method leads to a nonlinear system of algebraic equations (see e.g. Elman et al. [2005]). For its linearisation, we use the Picard iteration and get the system

$$\begin{bmatrix} \nu A + N(\mathbf{u}^k) & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}^{k+1} \\ \mathbf{p}^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}, \tag{3}$$

where $\mathbf{u}^{k+1}$ is the vector of unknown coefficients of velocity in the $(k+1)$-th iteration, $\mathbf{p}^{k+1}$ is the vector of unknown coefficients of the pressure, A is the matrix of diffusion, $N(\mathbf{u}^k)$ is the matrix of the advection where we substitute velocity from the previous step, B is the matrix from the continuity equation, and $\mathbf{f}$ and $\mathbf{g}$ are discrete right-hand side vectors arising from the Dirichlet boundary conditions. This already linear nonsymmetric system is solved by means of iterative substructuring.

To this end, we decompose Ω into N_S nonoverlapping subdomains. Degrees of freedom shared by several subdomains form the *interface*, whereas the rest are in the interior of subdomains. Importantly, for the Taylor–Hood elements employed in this work, parts of both velocity and pressure unknowns form the interface, denoted $\mathbf{u}_\Gamma$ and $\mathbf{p}_\Gamma$, respectively (superscript $^{k+1}$ will be omitted).

By eliminating interior unknown coefficients for velocity and pressure on each subdomain, the local Schur complement S_i can be formed. Finally, a global Schur complement can be assembled as $S = \sum_{i=1}^{N_S} R_i^{\Gamma T} S_i R_i^{\Gamma}$, where R_i^{Γ} is the 0–1 matrix selecting the interface unknowns of the i-th subdomain from the global vector of interface unknowns. We then solve the problem

$$S \begin{bmatrix} \mathbf{u}_\Gamma \\ \mathbf{p}_\Gamma \end{bmatrix} = g, \tag{4}$$

where g is the reduced right-hand side vector. In our implementation, Schur complements are not actually constructed. Instead, only their actions on vectors are evaluated within each iteration of a Krylov method.

Problem (4) is solved by the BiCGstab method and one step of BDDC is used as a preconditioner. As usual, a coarse correction is combined with independent subdomain corrections in each action of the preconditioner. The main difference of the employed approach from the standard BDDC preconditioner as introduced by Dohrmann [2003] is the need of the *adjoint* coarse basis functions for mapping fine residuals to the coarse problem, following Yano [2009]. This involves solving two saddle-point systems in the set-up phase of the preconditioner,

$$\begin{bmatrix} S_i & C_i^T \\ C_i & 0 \end{bmatrix} \begin{bmatrix} \Psi_i \\ \Lambda_i \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix}, \quad \begin{bmatrix} S_i^T & C_i^T \\ C_i & 0 \end{bmatrix} \begin{bmatrix} \Psi_i^* \\ \Lambda_i^T \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix},$$

where C_i is the matrix defining the local coarse degrees of freedom, which has as many rows as coarse degrees of freedom located in the subdomain. Finally, Ψ_i and Ψ_i^* are the matrices of standard and adjoint coarse basis functions.

As coarse degrees of freedom, we consider components of the velocity and the pressure at several *corners* selected according to Šístek et al. [2012], and arithmetic averages over edges and faces of subdomains. Constraints on their continuity in the coarse space are enforced component-wise on the velocities as well as on the pressure. The averaging at the interface unknowns applies diagonal matrix of weights to satisfy the partition of unity. The weights correspond to the inverse of the number of subdomains containing an interface unknown in this work.

3 Mesh partitioning

We compare two approaches to partitioning the computational domain and the mesh into subdomains. A standard approach is based on a conversion of the computational mesh into a graph. In the so-called dual graph, the finite elements represent vertices of the graph and if two elements share an edge (in 2D) or a face (in 3D), the corresponding graph vertices are connected by a graph edge. The task of partitioning a mesh is translated into a problem of dividing a graph into subgraphs, with the goal that the subgraphs contain approximately the same number of vertices and the number of edges connecting the subgraphs is minimized. We make use of the *METIS* library (version 4.0) for this purpose.

Graph partitioning provides an automated way for dividing the computational mesh into subdomains of well-balanced sizes even for complex geometries and meshes. However, information about the geometry of the interface is lost during the conversion into a graph, and the resulting interface can be very irregular. This is a known issue studied mathematically for elliptic problems e.g. by Klawonn et al. [2008].

Another, somewhat opposite, strategy is based on the geometry of the domain. The domain can be enclosed into its cuboidal bounding box $[x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]$. Two subdomains are created by bisecting the box into halves, with the cutting plane perpendicular to the longest edge. In the recursive bisection (RCB) algorithm, the longest subdomain edge is found as the maximum over subdomains, and one of the adjacent subdomains is bisected. This process is repeated until the given number of subdomains is reached.

This algorithm does not work well for complex unstructured meshes, since the strategy ignores numbers of elements in each block, and it can even create 'empty subdomains' with no elements. Nevertheless, for simple cuboidal domains, it is straightforward to produce a partition avoiding irregular interfaces. For a suitable number of subdomains and regular meshes, subdomain sizes are well-balanced in addition. In the rest of the paper, we refer to this strategy as the *geometric* partitioner.

Many geometries, including those of the hydrostatic bearings we aim at, are not completely general and can be decomposed into several cuboidal blocks in the first stage. In the second stage, each of these blocks can be partitioned as above.

4 Numerical results

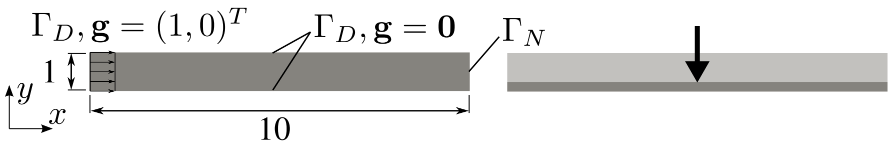
Our computations aim at the influence of interface irregularities on the BDDC solver for Navier-Stokes equations. In particular, we investigate the effect of the aspect ratio of the finite elements at the interface on convergence. This is motivated by our target application—simulations of oil flow in hydrostatic bearings with very narrow throttling gaps. In order to study this phenomenon, a benchmark problem suitable for such a study is proposed and the partitioning strategies described in Section 3 are compared.

The computations are performed by a parallel finite element package written in C++ and described by Šístek and Cirak [2015], with the *BDDCML* library being used for solving the arising systems of linear equations. The Picard iteration is terminated based on the change of subsequent solutions when $\left\|u^k - u^{k-1}\right\|_2 \leq 10^{-5}$ or after performing 100 iterations. The BiCGstab method is stopped based on the relative residual if $\left\|r^k\right\|_2 / \|g\|_2 \leq 10^{-6}$, with the limit of 1000 iterations.

As a measure of convergence, we monitor the number of BiCGstab iterations needed in one Picard iteration. Two matrix-vector multiplications are needed in each iteration of BiCGstab, and after each of them, the terminal condition is evaluated. Correspondingly, inspired by the Matlab `bicgstab` function, termination after the first matrix-vector multiplication is reported by a half iteration in the BiCGstab iteration counts. Numbers of iterations are presented as minimum, maximum, and mean over all nonlinear iterations for a given case.

The benchmark problem consists of a sequence of simple channels in 2D (Fig. 1) and 3D (Fig. 2). The dimension of the channels along one or two (in 3D) coordinates is gradually decreased, with the initial dimensions $10\times1\times1$ along the x, y, and z axes.

The computational mesh is based on rectangular (in 2D) or cuboidal (in 3D) finite elements uniformly distributed along each direction. The number of elements is $100\times10\times10$ along the x, y and z coordinates. In total, the 3-D problem contains 10 000 elements, 88 641 nodes, and 278 144 unknowns.



$\Gamma_D, \mathbf{g} = (1,0)^T$
$\Gamma_D, \mathbf{g} = \mathbf{0}$
Γ_N
1
y
x
10

Fig. 1 The narrowing channel 2-D benchmark; original channel (left) and narrowing along the y-axis (right).

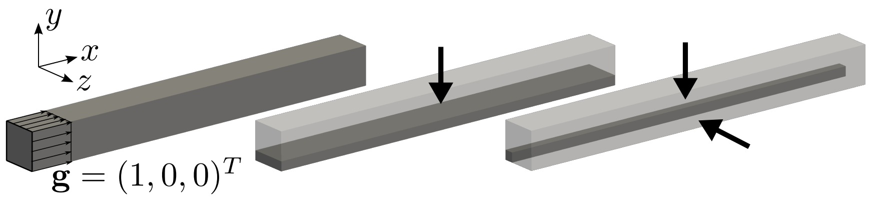


Fig. 2 The narrowing channel 3-D benchmark; original channel (left), narrowing along the y-axis (centre), and narrowing along both y and z-axes (right).

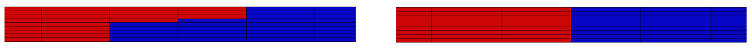


Fig. 3 Detail of the interface between two subdomains in 2D for graph (left) and geometric (right) partitioner.

The *aspect ratio of elements* $AR = h_{max}/h_{min}$ is defined as the ratio of the longest edge of the element h_{max} to its shortest counterpart h_{min}. The $AR = 1$ corresponds to square (or cubic) elements. We test the sequence of narrowing channels for $AR \in \{1,2,4,10,20,40,100\}$.

The velocity at the inlet starts from $\mathbf{g} = (1,0,0)^T$ for $x = 0$, the velocity at the walls is fixed to $\mathbf{g} = \mathbf{0}$, and the face of the channel for $x = 10$ corresponds to Γ_N. We have considered two scenarios for the inflow velocity during the narrowing. The first is simply keeping the magnitude of the velocity fixed throughout the sequence. In the second scenario, the magnitude of the velocity is increased proportionally to the decrease of the height, so that the Reynolds number, defined as $Re = \frac{|\mathbf{u}|D}{\nu}$, is kept constant for the decreasing channel height D. However, results for both scenarios of the inlet boundary condition have been almost identical, and we present only the results for fixed Reynolds number for brevity. We use $\nu = 1$ for our computations. The channel is divided into 4 subdomains by the graph and the geometric partitioners described in Section 3.

First we look at the two-dimensional problem. For the graph partitioner, the interface contains both long and short edges of elements. On the other hand, the interface is composed solely from short edges for the geometric partitioner (see Fig. 3). Corresponding results are in Table 1.

For the 3-D case, we consider two kinds of problems. First we decrease only the y-dimension of the channel, while in the second case, we shrink both y and z dimensions of the cross-section (see Fig. 2). The graph partitioner produces rough

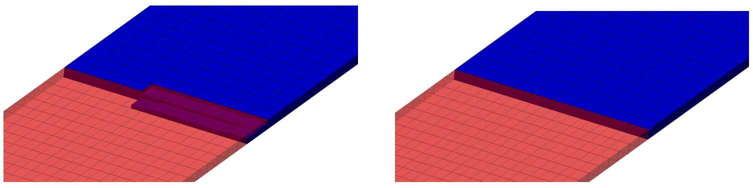


Fig. 4 Detail of the interface between two subdomains for narrowing along the *y*-coordinate in 3D for graph (left) and geometric (right) partitioner.

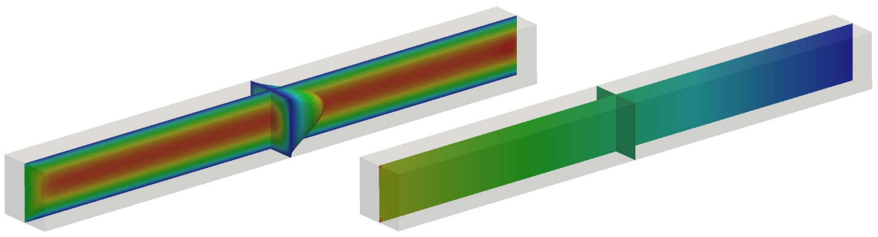


Fig. 5 Solution in the initial 3-D channel geometry; magnitude of velocity (left) and pressure in the plane of symmetry (right).

partitioner		graph							geometric						
Æ		1	2	4	10	20	40	100	1	2	4	10	20	40	100
Picard its.		4	4	5	5	7	6	40	3	4	5	5	6	6	5
BiCGstab its.	min	9	10.5	13.5	13.5	15	16.5	17.5	4.5	4.5	4.5	4	3	3	3
	max	9.5	10.5	13.5	15	16	17.5	19.5	4.5	4.5	4.5	4	3	3	3
	mean	9.4	10.5	13.5	14.2	15.2	16.7	18.1	4.5	4.5	4.5	4	3	3	3

Table 1 Numbers of iterations for graph and geometric partitioners for 2-D narrowing channel.

interface in both cases, while the geometric partitioner leads to rectangular faces at the interface in the first case (see Fig. 4) and square faces in the second case. Resulting numbers of iterations are presented in Tables 2 and 3. Numbers in italic are runs that did not converge due to reaching the maximal number of iterations or time restrictions. A solution of the problem for the initial channel geometry is presented in Fig. 5.

From Tables 1, 2, and 3 we can conclude that Æ of faces at the interface has a remarkable influence on the number of BiCGstab iterations in each Picard iteration.

Using the graph partitioner results in a rough interface combining long and short edges. This has a large impact on the efficiency of the BDDC preconditioner and the number of linear iterations increases significantly.

Employing the geometric partitioner leads to straight cuts between subdomains aligned with layers of elements. In 2D, this is sufficient to achieve convergence of the linear solver independent of Æ. In 3D, the situation is more subtle. For the case of narrowing the channel only along the *y*-axis, the aspect ratio of the rectangular

partitioner		graph							geometric						
$\mathcal{AR}$		1	2	4	10	20	40	100	1	2	4	10	20	40	100
Picard its.		4	5	5	42	5	*100*	*100*	4	5	5	5	5	5	99
BiCGstab its.	min	17.5	20	25.5	44.5	84.5	145	400	5.5	6.5	7.5	11.5	16	19.5	19.5
	max	18.5	20.5	25.5	51	113.5	858	*1000*	5.5	6.5	7.5	12	17.5	19.5	21
	mean	18.3	20.4	25.5	46.2	93.9	209	761	5.5	6.5	7.5	11.9	17.2	19.5	19.5

Table 2 Numbers of iterations for graph and geometric partitioners for 3-D channel narrowed along the y-coordinate.

partitioner		graph							geometric						
$\mathcal{AR}$		1	2	4	10	20	40	100	1	2	4	10	20	40	100
Picard its.		4	4	4	5	8	19	*28*	4	4	4	5	5	5	4
BiCGstab its.	min	17.5	19.5	27.5	36	51	80	197	5.5	5.5	6	5	4.5	4.5	4.5
	max	18.5	20.5	28	41.5	53	92.5	*1000*	5.5	6	6	5.5	5	5	4.5
	mean	18.3	19.8	27.9	39.5	51.8	87.7	590	5.5	5.9	6	5.1	4.9	4.6	4.5

Table 3 Numbers of iterations for graph and geometric partitioners for 3-D channel narrowed along both y and z-coordinates.

element faces at the interface also worsens during contracting the channel. This is translated into a slight growth of the number of BiCGstab iterations in Table 2 even in this case, although the convergence is much more favourable than for the graph partitioner. If we narrow the channel along both y and z coordinates, the shape of the element faces at the interface does not deteriorate from squares, and we observe fast convergence independent of $\mathcal{AR}$ in Table 3.

5 Conclusion

We have investigated the influence of an irregular interface on the performance of the BDDC method for Navier-Stokes equations. A benchmark problem of a narrowing channel in 2D and 3D has been proposed to evaluate the impact of aspect ratios of finite elements on the convergence of iterative solvers for the arising system of equations. A simple partitioning strategy based on an application of a regular geometric division of simple sub-blocks of the computational mesh has been presented. This approach was applied to the benchmark channel problems. The number of BiCGstab iterations required when using the geometric partitioner has been compared to the number of iterations required when using a graph partitioner. This rather simple idea has dramatically improved convergence of our BDDCML solver. Our next aim is to apply the idea to real geometries of hydrostatic bearings with block structured meshes. The preliminary results in this direction are very promising.

Acknowledgements

We are grateful to Fehmi Cirak for valuable discussions on domain partitioning by the recursive bisection algorithm, and to Santiago Badia for discussing the impact of element aspect ratios on convergence of BDDC. This work was supported by the Czech Technical University in Prague through the student project SGS16/206/OHK2/3T/12, by the Czech Science Foundation through grant 14-02067S, and by the Czech Academy of Sciences through RVO:67985840.

References

C. R. Dohrmann. A preconditioner for substructuring based on constrained energy minimization. *SIAM J. Sci. Comput.*, 25(1):246–258, 2003.

H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Numerical Mathematics and Scientific Computation. Oxford University Press, New York, 2005.

M. Hanek, J. Šístek, and P. Burda. An application of the BDDC method to the Navier-Stokes equations in 3-D cavity. In J. Chleboun, P. Přikryl, K. Segeth, J. Šístek, and T. Vejchodský, editors, *Proceedings of Programs and Algorithms of Numerical Mathematics 17, Dolní Maxov, Czech Republic, June 8–13, 2014*, pages 77–85. Institute of Mathematics AS CR, 2015.

A. Klawonn, O. Rheinbach, and O. B. Widlund. An analysis of a FETI-DP algorithm on irregular subdomains in the plane. *SIAM J. Numer. Anal.*, 46(5):2484–2504, 2008.

J. Li and X. Tu. A nonoverlapping domain decomposition method for incompressible Stokes equations with continuous pressures. *SIAM Journal on Numerical Analysis*, 51(2):1235–1253, 2013.

J. Li and O. B. Widlund. BDDC algorithms for incompressible Stokes equations. *SIAM J. Numer. Anal.*, 44(6):2432–2455, 2006.

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.

J. Šístek, B. Sousedík, P. Burda, J. Mandel, and J. Novotný. Application of the parallel BDDC preconditioner to the Stokes flow. *Comput. & Fluids*, 46:429–435, 2011.

J. Šístek, M. Čertíková, P. Burda, and J. Novotný. Face-based selection of corners in 3D substructuring. *Math. Comput. Simulat.*, 82(10):1799–1811, 2012.

B. Sousedík, J. Šístek, and J. Mandel. Adaptive-Multilevel BDDC and its parallel implementation. *Computing*, 95(12):1087–1119, 2013.

M. Yano. Massively parallel solver for the high-order Galerkin least-squares method. Master's thesis, Massachusests Institute of Technology, 2009.