SIAM J. Sci. Comput.
Vol. 42, No. 6, pp. C359–C383

MULTILEVEL BDDC FOR INCOMPRESSIBLE NAVIER–STOKES EQUATIONS[*]

MARTIN HANEK[†], JAKUB ŠÍSTEK[‡], AND PAVEL BURDA[§]

Abstract. We present an approach to the numerical solution of steady Navier–Stokes equations. Approximation by the finite element method (FEM) leads to a nonlinear saddle-point system. The system is linearized by the Picard iteration, which leads to a sequence of linear saddle-point systems with nonsymmetric matrices. In this paper, we study the application of Balancing Domain Decomposition based on Constraints (BDDC) to these systems. In particular, we formulate the multilevel BDDC method and explore its applicability for the benchmark problem of lid-driven cavity. Another contribution of the paper is describing the development and application of our BDDC solver to real-world problems of oil flow in hydrostatic bearings.

Key words. multilevel BDDC, domain decomposition, finite element method, Navier–Stokes equations

AMS subject classifications. 65N55, 65Y05, 76D05

DOI. 10.1137/19M1276479

1. Introduction.

The Balancing Domain Decomposition based on Constraints (BDDC) was introduced in [8] for the Poisson problem and linear elasticity. The underlying theory was presented in [20, 21]. It was shown that the BDDC method is spectrally equivalent to the FETI-DP method [11]. This preconditioner was then used without an explicit coarse problem for systems with symmetric positive definite matrices in [19]. The multilevel extension of the BDDC method was presented first for three levels in [28], and later for an arbitrary number of levels in [22]. The multilevel BDDC method was combined with the adaptive selection of coarse unknowns and implemented into an open-source parallel solver *BDDCML* in [26]. A recent overview of adaptive BDDC was provided in [23]. The potential of the multilevel method to scale up to 499 thousand cores was demonstrated in [3]. More on domain decomposition methods in general can be found in the monograph [27].

In [18], the BDDC method was first applied to a saddle-point system with symmetric indefinite matrix arising from the discretization of the Stokes problem. This approach uses finite elements with discontinuous approximation of pressure, which leaves only unknowns for velocity components at the interface. An approach for the Stokes problem using elements with continuous pressure was investigated in [25], and a different approach was later introduced in [17].

By discretizing and linearizing an approximation of the Navier–Stokes equations, we get saddle-point systems with nonsymmetric matrices. An application of the

[*]Submitted to the journal's Software and High-Performance Computing section July 22, 2019; accepted for publication (in revised form) August 31, 2020; published electronically November 19, 2020.
https://doi.org/10.1137/19M1276479
Funding: This research was supported by the Czech Science Foundation through grant 18-09628S, and by the Czech Academy of Sciences through RVO:67985840. 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."

[†]Czech Technical University in Prague, Prague, 120 00, Czech Republic, and Institute of Mathematics of the Czech Academy of Sciences, Prague, 115 67, Czech Republic (martin.hanek@fs.cvut.cz).

[‡]Institute of Mathematics of the Czech Academy of Sciences, Prague, 115 67, Czech Republic (sistek@math.cas.cz).

[§]Czech Technical University in Prague, Prague, 120 00, Czech Republic (pavel.burda@fs.cvut.cz).

BDDC method to nonsymmetric matrices arising from the advection-diffusion problems was presented in [29, 30], where the method was formulated without an explicit coarse problem. An explicit coarse problem of BDDC was presented for nonsymmetric problems arising from the Euler equations in [33]. Earlier domain decomposition methods for problems with nonsymmetric matrices include the Robin-Robin preconditioner [1, 2] for advection-diffusion problems.

By combining the approaches from [25, 33], we applied the 2-level BDDC method to the Navier–Stokes equations for the lid-driven cavity benchmark problem in [14]. The main focus of this study is a formulation of the multilevel BDDC preconditioner for nonsymmetric problems and its application to linear systems obtained by Picard linearization of the Navier–Stokes equations. Our computations employ the *BDDCML* solver. For the benchmark problem of flow in a three-dimensional (3-D) lid-driven cavity, we compare the convergence behavior of the 2-, 3-, and 4-level method and perform weak scaling tests. Next, the 2-level variant of the BDDC preconditioner is applied to two industrial problems of oil flow in hydrostatic bearings.

An important building block of the BDDC method is the choice of weights used for averaging a discontinuous solution at the interface among subdomains. There are some standard types of weights like arithmetic average (also known as cardinality scaling), or weighted average based on diagonal entries of subdomain matrices. A recent advanced type of scaling is the deluxe scaling [9, 4, 7]. While this scaling is not available in our implementation, we propose a new type of interface weights inspired by the solved flow problems. We refer to this strategy as the upwind-based scaling.

This paper is organized as follows. Section 2 presents the application of the finite element method to the Navier–Stokes equations. Section 3 is devoted to iterative substructuring. In section 4, we describe the multilevel BDDC method for nonsymmetric systems of equations, while section 5 is devoted to the description of interface scaling types used in our calculations. In section 6, we present numerical results. Finally, conclusions are drawn in section 7.

2. Navier–Stokes equations and the finite element method.

In this section, we introduce the mathematical model and its numerical approximation. We consider stationary incompressible flow in 3-D domain Ω governed by the Navier–Stokes equations with zero body forces (see, e.g., [10]),

$$(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - \nu\Delta\boldsymbol{u} + \nabla p = \boldsymbol{0} \quad \text{in } \Omega, \tag{2.1}$$

$$\nabla\cdot\boldsymbol{u} = 0 \quad \text{in } \Omega, \tag{2.2}$$

where $\boldsymbol{u} = (u_x, u_y, u_z)^T$ is an unknown velocity vector, p is an unknown pressure normalized by (constant) density, ν is a given kinematic viscosity, and Ω is the solution domain. In addition, we consider the following boundary conditions:

$$\boldsymbol{u} = \boldsymbol{g} \quad \text{on } \Gamma_D, \tag{2.3}$$

$$-\nu(\nabla\boldsymbol{u})\mathbf{n} + p\mathbf{n} = 0 \quad \text{on } \Gamma_N, \tag{2.4}$$

where Γ_D and Γ_N are parts of the boundary $\partial\Omega$, $\overline{\Gamma_D}\cup\overline{\Gamma_N} = \partial\Omega$, $\Gamma_D\cap\Gamma_N = \emptyset$, $\mathbf{n}$ is the outer unit normal vector of the boundary, and $\boldsymbol{g}$ is a given function.

2.1. Weak formulation.

To apply the numerical approximation by the finite element method, we first derive the mixed weak formulation. We multiply (2.1) and (2.2) by test functions, which are different for velocity and pressure, and integrate over the solution domain Ω. After using the divergence theorem, we get the final formulation:

Seek $\boldsymbol{u} \in V_g$ and $p \in L^2(\Omega)$ satisfying

$$\int_\Omega (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} \cdot \boldsymbol{v} \mathrm{d}\Omega + \nu \int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} \mathrm{d}\Omega - \int_\Omega p \nabla \cdot \boldsymbol{v} \mathrm{d}\Omega = 0 \quad \forall \boldsymbol{v} \in V, \tag{2.5}$$

$$\int_\Omega q \nabla \cdot \boldsymbol{u} \mathrm{d}\Omega = 0 \quad \forall q \in L^2(\Omega). \tag{2.6}$$

Here the function spaces are

$$V_g := \left\{ \boldsymbol{u} \in H^1(\Omega)^3, \boldsymbol{u} = \boldsymbol{g} \text{ on } \Gamma_D \right\},$$
$$V := \left\{ \boldsymbol{v} \in H^1(\Omega)^3, \boldsymbol{v} = \boldsymbol{0} \text{ on } \Gamma_D \right\},$$

where the function equalities are understood in the sense of traces, and $H^1(\Omega)$ is the usual Sobolev space.

2.2. Finite element approximation. The first step for using the finite element method is creating a mesh. We are using hexahedral finite elements in this study. On each element, we approximate the solution of pressure and velocity and their test functions by polynomials of a certain degree.

A number of finite elements are suitable for the Navier–Stokes equations. We employ the 3-D Taylor–Hood $Q_2 - Q_1$ finite elements (see, e.g., [10]). These elements locally approximate pressure functions by polynomials of the first degree, and velocity components by polynomials of the second degree, while both fields are continuous on the interelement boundaries.

The convergence theory of the $Q_2 - Q_1$ finite elements is well-established for the Stokes problem; see, e.g., [5, 10]. The 3-D version of the elements satisfies the Babuška–Brezzi (also known as *inf-sup*) stability condition, one of the requirements for developing the optimal error estimate. Given a sufficient regularity of the solution $(\boldsymbol{u}, p)$ and a bounded aspect ratio of elements, the $Q_2 - Q_1$ elements satisfy the a priori error estimate [10, Theorem 5.6],

$$\|\nabla(\boldsymbol{u} - \boldsymbol{u}_h)\| + \|p - p_h\|_{0,\Omega} \leq Ch^2(\|D^3\boldsymbol{u}\| + \|D^2 p\|), \tag{2.7}$$

where $\|D^3\boldsymbol{u}\|$ and $\|D^2 p\|$ measure the regularity of the solution, h is the length of the longest edge in the mesh, $\|\cdot\|$ is the L_2-norm, and $\|\cdot\|_{0,\Omega}$ is the quotient space norm which removes the mean value of pressure.

During the assembly of the system of algebraic equations, we substitute the finite element counterparts of $\boldsymbol{u}$, p, $\boldsymbol{v}$, and q,

$$\boldsymbol{u}_h = \sum_{i=1}^{3n_{\boldsymbol{u}}} u_i \boldsymbol{\phi}_i, \quad p_h = \sum_{i=1}^{n_p} p_i \psi_i, \quad \boldsymbol{v}_h = \sum_{i=1}^{3n_{\boldsymbol{u}}} v_i \boldsymbol{\phi}_i, \quad q_h = \sum_{i=1}^{n_p} q_i \psi_i,$$

in the weak formulation (2.5) and (2.6).

Here $\boldsymbol{\phi}_i$ are vector basis functions for velocity, ψ_i are scalar basis functions for pressure, $n_{\boldsymbol{u}}$ is the number of nodes with velocity unknowns, and n_p is the number of nodes with pressure unknowns. For the considered Taylor–Hood finite elements, $n_{\boldsymbol{u}}$ is approximately eight times larger than n_p.

After the above substitution, we obtain the following system of algebraic equations:

$$\begin{bmatrix} \nu\mathbf{A} + \mathbf{N}(\mathbf{u}) & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}, \tag{2.8}$$

where $\mathbf{u}$ is the vector of unknown coefficients of velocity, $\mathbf{p}$ is the vector of unknown coefficients of pressure, $\mathbf{A}$ is the matrix of diffusion, $\mathbf{N}(\mathbf{u})$ is the matrix of advection which depends on the solution, 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. Each part of system (2.8) is assembled as (see [10])

$$\mathbf{A} = [a_{ij}], \quad a_{ij} = \int_\Omega \nabla\boldsymbol{\phi}_i : \nabla\boldsymbol{\phi}_j \, \mathrm{d}\Omega, \tag{2.9}$$

$$\mathbf{N}(\mathbf{u}) = [n_{ij}], \quad n_{ij} = \int_\Omega (\mathbf{u}\cdot\nabla)\boldsymbol{\phi}_j \cdot \boldsymbol{\phi}_i \, \mathrm{d}\Omega, \tag{2.10}$$

$$B = [b_{lj}], \quad b_{lj} = -\int_\Omega \psi_l \nabla\cdot\boldsymbol{\phi}_j \, \mathrm{d}\Omega, \tag{2.11}$$

(2.12)

$$\mathbf{f} = [f_i], \quad f_i = -\sum_{j=3n_{\boldsymbol{u}}+1}^{3(n_{\boldsymbol{u}}+\partial n_{\boldsymbol{u}})} u_j \int_\Omega (\mathbf{u}\cdot\nabla)\boldsymbol{\phi}_j\cdot\boldsymbol{\phi}_i \, \mathrm{d}\Omega - \nu \sum_{j=3n_{\boldsymbol{u}}+1}^{3(n_{\boldsymbol{u}}+\partial n_{\boldsymbol{u}})} u_j \int_\Omega \nabla\boldsymbol{\phi}_j : \nabla\boldsymbol{\phi}_i \, \mathrm{d}\Omega,$$

(2.13)

$$\mathbf{g} = [g_l], \quad g_l = \sum_{j=3n_{\boldsymbol{u}}+1}^{3(n_{\boldsymbol{u}}+\partial n_{\boldsymbol{u}})} u_j \int_\Omega \psi_l \nabla\cdot\boldsymbol{\phi}_j \, \mathrm{d}\Omega.$$

System (2.8) is nonlinear due to the matrix $\mathbf{N}(\mathbf{u})$, and we use the Picard iteration for its linearization. This leads to solving a sequence of linear systems of equations in the form

$$\begin{bmatrix} \nu\mathbf{A} + \mathbf{N}(\mathbf{u}^k) & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}^{k+1} \\ \mathbf{p}^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}, \tag{2.14}$$

where $\mathbf{N}(\mathbf{u}^k)$ means that we substitute a solution of velocity from the previous step to the matrix $\mathbf{N}$. This—already linear—nonsymmetric system is solved by means of iterative substructuring in the subsequent part of this paper.

3. Iterative substructuring.

We decompose our solution domain Ω into N nonoverlapping subdomains (see Figure 1). This decomposition means that the degrees of freedom (dofs) shared by several subdomains are only at the interface of each subdomain, while the remaining unknowns are in the interior of the subdomains. For the Taylor–Hood finite elements used in our work, both velocity and pressure unknowns are shared among subdomains and hence become part of the interface.

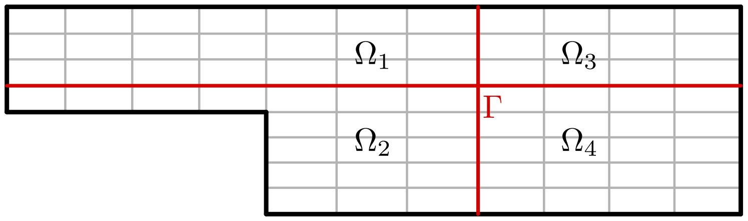


FIG. 1. *Nonoverlapping domain decomposition with interface among subdomains.*

In order to apply the BDDC preconditioner to problem (2.14), we formally reorder this system so that the interface unknowns for velocity and pressure are at the end of the corresponding part of the vector of unknowns. This leads to the following block system:

$$\begin{bmatrix} \nu\mathbf{A}_{11}+\mathbf{N}_{11} & \nu\mathbf{A}_{12}+\mathbf{N}_{12} & B_{11}^T & B_{21}^T \\ \nu\mathbf{A}_{21}+\mathbf{N}_{21} & \nu\mathbf{A}_{22}+\mathbf{N}_{22} & B_{12}^T & B_{22}^T \\ B_{11} & B_{12} & 0 & 0 \\ B_{21} & B_{22} & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u_1} \\ \mathbf{u_2} \\ \mathbf{p_1} \\ \mathbf{p_2} \end{bmatrix} = \begin{bmatrix} \mathbf{f_1} \\ \mathbf{f_2} \\ \mathbf{g_1} \\ \mathbf{g_2} \end{bmatrix}, \tag{3.1}$$

where subscript $_1$ denotes the part with the interior unknowns, and subscript $_2$ denotes the part with the interface unknowns.

This block system is now permuted and divided to get the interface problem, as was similarly done for the Stokes problem in [25],

$$S \begin{bmatrix} \mathbf{u_2} \\ \mathbf{p_2} \end{bmatrix} = g, \tag{3.2}$$

where

$$\begin{aligned} S = & \begin{bmatrix} \nu\mathbf{A}_{22}+\mathbf{N}_{22} & B_{22}^T \\ B_{22} & 0 \end{bmatrix} \\ & - \begin{bmatrix} \nu\mathbf{A}_{21}+\mathbf{N}_{21} & B_{12}^T \\ B_{21} & 0 \end{bmatrix} \begin{bmatrix} \nu\mathbf{A}_{11}+\mathbf{N}_{11} & B_{11}^T \\ B_{11} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \nu\mathbf{A}_{12}+\mathbf{N}_{12} & B_{21}^T \\ B_{12} & 0 \end{bmatrix} \end{aligned}$$

is the Schur complement with respect to the interface unknowns and

$$g = \begin{bmatrix} \mathbf{f_2} \\ \mathbf{g_2} \end{bmatrix} - \begin{bmatrix} \nu\mathbf{A}_{21}+\mathbf{N}_{21} & B_{12}^T \\ B_{21} & 0 \end{bmatrix} \begin{bmatrix} \nu\mathbf{A}_{11}+\mathbf{N}_{11} & B_{11}^T \\ B_{11} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{f_1} \\ \mathbf{g_1} \end{bmatrix}$$

is the reduced right-hand side.

Problem (3.2) is solved by the BiCGstab method [31] with one step of BDDC used as the preconditioner. Domain decomposition allows us to perform the action of the BDDC preconditioner and of the matrix S in parallel in each iteration. The solution is realized by the multilevel BDDC implementation in the *BDDCML* library[1] [26].

4. Multilevel BDDC for nonsymmetric systems.

In this section, we describe the multilevel BDDC preconditioner for the nonsymmetric problems arising from the discretization of the Navier–Stokes equations above. In the kth iteration, the preconditioner is applied to the residual,

$$r^k = g - S \begin{bmatrix} \mathbf{u_2}^k \\ \mathbf{p_2}^k \end{bmatrix}, \tag{4.1}$$

obtained from the BiCGstab method generating an approximate solution to problem (3.2).

A crucial component of a BDDC algorithm is the choice of the primal variables, also called coarse dofs, on each level. The BDDC preconditioner then solves an approximate interface problem in the space of functions that are continuous in these primal variables.

First, we classify the interface into vertices, edges, and faces of subdomains. For the Taylor–Hood elements, dofs are located at nodes. Hence, faces are defined as

[1] http://users.math.cas.cz/~sistek/software/bddcml.html

subsets of nodes shared by the same two subdomains, edges are subsets of nodes shared by several subdomains, and vertices are degenerate edges with only one node.

The main primal unknowns in this paper are arithmetic averages over edges and faces defined independently for each component of velocity and for the pressure unknowns. In addition, pointwise continuity of these components is required at subdomain vertices, also called corners. This means that for subdomains that have in common at least a face (f), an edge (e), or a corner (c), the following functionals are equal:

$$L^{e,f}(u) = \sum_j u_{ij}, \quad L^{e,f}(p) = \sum_j p_j, \quad L^c(u) = u_i, \quad L^c(p) = p,$$

where u_i, $i = 1, 2, 3$, are the components of velocity, and index j corresponds to the number of unknowns on the shared edge or face.

In each iteration of the BiCGstab algorithm, two actions of the BDDC preconditioner are required. This involves solving a coarse problem and independent subdomain problems on each level. First, we look at the considered coarse problem. In [29, 30], the BDDC preconditioner was used without an explicit coarse problem for solving the advection-diffusion problem. Here we use an approach with an explicit coarse problem.

Before applying the preconditioner, we have to set it up. First, we find the coarse basis functions independently for each subdomain on each level. For finding the subdomains on the next level, we look at the subdomains on the previous level as elements for the next level, and the coarse dofs will be considered as unknowns. Subdomains on the next level are formed by connecting several subdomains from the previous level. For example, in Figure 1, we can join subdomains Ω_1 and Ω_2 to form the first subdomain, and subdomains Ω_3 and Ω_4 to form the second subdomain on the next level. From the subdomain matrix on each level, we build and solve the saddle-point system

$$\begin{bmatrix} S_i^\ell & C_i^{\ell^T} \\ C_i^\ell & 0 \end{bmatrix} \begin{bmatrix} \Psi_i^\ell \\ \Lambda_i^\ell \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix}, \tag{4.2}$$

where ℓ is the level of the BDDC method, S_i^ℓ is the Schur complement with respect to the interface of the ith subdomain (built as in 3.2), and C_i^ℓ is the matrix defining the coarse dofs, which has as many rows as the number of coarse dofs defined at the subdomain. The solution Ψ_i^ℓ is the matrix of coarse basis functions, with every column corresponding to one coarse unknown on the subdomain. These functions are equal to one in one coarse dof, and they are equal to zero in the remaining local coarse unknowns.

Note the presence of the explicit Schur complement S_i^ℓ in (4.2). In BDDC, an analogous problem can be formed for the original subdomain matrix A_i^ℓ, as was used, e.g., in the original paper [8], considering the discrete harmonic extension of the coarse basis functions to subdomain interiors. To be more specific, dividing the local unknowns into interior (subscript $_1$) and interface (subscript $_2$) categories, we introduce a block structure

$$A_i^\ell = \begin{bmatrix} A_{i_{11}}^\ell & A_{i_{12}}^\ell \\ A_{i_{21}}^\ell & A_{i_{22}}^\ell \end{bmatrix},$$

from which the local Schur complement can be expressed as

$$S_i^\ell = A_{i_{22}}^\ell - A_{i_{21}}^\ell (A_{i_{11}}^\ell)^{-1} A_{i_{12}}^\ell.$$

Although we describe the algorithm using the explicit Schur complements S_i^ℓ, the computations are performed without their explicit construction, and only factorizations and back-substitutions with $A_{i_{11}}^\ell$ are required.

As introduced in [33], a set of adjoint coarse basis functions $\Psi_i^{*\ell}$ is also needed for nonsymmetric problems. These are obtained as the solution to the problem (4.2) with a transposed matrix,

$$\begin{bmatrix} S_i^{\ell^T} & C_i^{\ell^T} \\ C_i^\ell & 0 \end{bmatrix} \begin{bmatrix} \Psi_i^{*\ell} \\ \Lambda_i^{\ell^T} \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix}. \tag{4.3}$$

After solving (4.2) and (4.3), we can get, on each level, the local coarse matrix on each subdomain as

$$A_{Ci}^\ell = (\Psi_i^{*\ell})^T S_i^\ell \Psi_i^\ell = -\Lambda_i^\ell. \tag{4.4}$$

The local matrices A_{Ci}^ℓ resemble the element matrices from the FEM. In the multilevel method, they are assembled into subdomain matrices on the next level,

$$A_j^{\ell+1} = \sum_{i=1}^{N_{S_j}} R_{Cij}^\ell A_{Ci}^\ell R_{Cij}^{\ell^T}, \tag{4.5}$$

where $R_{Cij}^\ell = R_j^{\ell+1} R_{Ci}^{\ell^T}$. Here $R_j^{\ell+1}$ is the restriction of the global vector of unknowns to those present on the jth subdomain on level $\ell+1$, R_{Ci}^ℓ is the restriction of the global vector of coarse unknowns to those present at the ith subdomain on level ℓ, and N_{S_j} is the number of subdomains from level ℓ which form the jth subdomain on the $(\ell+1)$st level. On the last level, we build the global coarse problem. Therefore, $A_j^L = A^L$, where L is the number of levels of the BDDC method. The setup of the multilevel BDDC is described in Algorithm 4.1.

Algorithm 4.1 Setup of the BDDC preconditioner with L levels

```
1: for level $\ell = 1, \dots, L-1$ do
2:     from subdomain matrix $A_i^\ell$ get $S_i^\ell$
3:     solve problems (4.2) and (4.3) to get $\Psi_i^\ell$ and $\Psi_i^{*\ell}$
4:     get local coarse matrices $A_{Ci}^\ell$ from (4.4)
5:     build subdomain matrices on the next level $A_j^{\ell+1}$ (4.5)
6: end for
7: factorize global coarse matrix $A^L$
```

After this setup, the preconditioner can be applied in each iteration. Let us first look at the 2-level method and describe the modification for the multilevel method later. To begin, we take the residual vector r^k and extract its local parts on each subdomain as

$$r_i = W_i R_i r^k, \tag{4.6}$$

where R_i is the operator restricting a global interface vector to the ith subdomain, and matrix W_i applies weights to satisfy the partition of unity. The types of weights we use are described in detail in section 5.

Then we get the coarse residual as

$$r_C = \sum_{i=1}^{N} R_{Ci}^T \Psi_i^{*T} r_i. \tag{4.7}$$

Note that here we use the adjoint coarse basis functions Ψ_i^{*T}. Then we solve the coarse problem

$$A_C u_C = r_C \tag{4.8}$$

and distribute the coarse solution to each subdomain, followed by prolonging it on the fine mesh (using standard coarse basis functions Ψ_i) as

$$u_{Ci} = \Psi_i R_{Ci} u_C. \tag{4.9}$$

Now we look at the subdomain problems. On each subdomain, a saddle-point system

$$\begin{bmatrix} S_i & C_i^T \\ C_i & 0 \end{bmatrix} \begin{bmatrix} u_i \\ \lambda \end{bmatrix} = \begin{bmatrix} r_i \\ 0 \end{bmatrix} \tag{4.10}$$

is solved. Here λ are Lagrange multipliers, and S_i and C_i are the same matrices as in (4.2). After solving this problem on each subdomain, we get the subdomain correction u_i.

After using the 2-level BDDC preconditioner, we get the preconditioned residual u^k, $M_{BDDC} : r^k \to u^k$, by combining the subdomain corrections with the subdomain coarse solution as

$$u^k = \sum_{i=1}^{N} R_i^T W_i (u_i + u_{Ci}). \tag{4.11}$$

If we want to add more levels to the method, we have to look at problem (4.8) first. In the 2-level method, we solve (4.8) by a direct method, whereas in the multilevel method, we apply a step of the BDDC method to solve the coarse problem (4.8) only approximately. This means that we build residuals and matrices (in setup) corresponding to the cluster of subdomains, which will form one subdomain on the next level. First, we build global coarse residual on the first level as in the standard 2-level method,

$$r_C = \sum_{i=1}^{N} R_{Ci}^T \Psi_i^{*T} r_i, \tag{4.12}$$

and solve subdomain problems on the first level (4.10).

Now starting on the second level and ending on the $(L-1)$st level we denote

$$u^\ell = u_C^{\ell-1} \quad \text{and} \quad r^\ell = r_C^{\ell-1},$$

extract local parts of the residual on each subdomain,

$$r_i^\ell = W_i^\ell R_i^\ell r^\ell, \tag{4.13}$$

and solve the following interior problem on each subdomain:

$$A_{i_{11}}^\ell u_{i_1}^\ell = r_{i_1}^\ell, \tag{4.14}$$

where subscript $_1$ again corresponds to the the interior unknowns of the subdomain (subscript $_2$ will correspond to the interface unknowns of the subdomain). After solving this problem we perform the interior precorrection as

$$\tilde{r}_{i_2}^\ell = r_{i_2}^\ell - A_{i_{21}}^\ell u_{i_1}^\ell. \tag{4.15}$$

Next, we solve subdomain problems

$$\begin{bmatrix} S_i^\ell & C_i^{\ell^T} \\ C_i^\ell & 0 \end{bmatrix} \begin{bmatrix} u_{i_2}^\ell \\ \lambda^\ell \end{bmatrix} = \begin{bmatrix} \tilde{r}_{i_2}^\ell \\ 0 \end{bmatrix} \tag{4.16}$$

and build the coarse residual,

$$r_C^\ell = \sum_{i=1}^{N^\ell} R_{Ci}^{\ell^T} \Psi_i^{*\ell T} \tilde{r}_{i_2}^\ell. \tag{4.17}$$

Finally, we solve the coarse problem on the last level L to get $u^L = u_C^{L-1}$.

After getting the solution on the highest level, we gradually build the approximate coarse solution on the first level. Starting on level $L-1$ and going down to level 2 we have the coarse solution distributed to each subdomain and the subdomain solution prepared in (4.16),

$$u_{C_i}^\ell = \Psi_i^\ell R_{C_i}^\ell u_C^\ell \quad \text{and} \quad u_{i_2}^\ell. \tag{4.18}$$

Then we build the BDDC approximation of the global interface solution,

$$u_2^\ell = \sum_{i=1}^{N_S^\ell} R_i^{\ell^T} W_i^\ell (u_{C_i}^\ell + u_{i_2}^\ell). \tag{4.19}$$

After that, we distribute the interface solution on each subdomain,

$$u_{i_2}^\ell = R_i^\ell u_2^\ell, \tag{4.20}$$

and compute the interior correction $\tilde{u}_{1_i}^\ell$ on each subdomain,

$$A_{i_1 1}^\ell \tilde{u}_{i_1}^\ell = -A_{i_1 2}^\ell u_{i_2}^\ell. \tag{4.21}$$

Finally, we bring in the interior correction from (4.14) and build the approximate coarse solution as

$$u_C^{\ell-1} = u^\ell = \begin{bmatrix} \sum_{i=1}^{N_S^\ell} R_{1i}^{\ell^T} (\tilde{u}_{i_1}^\ell + u_{i_1}^\ell) \\ u_2^\ell \end{bmatrix}, \tag{4.22}$$

where $R_{1i}^{\ell^T}$ is the operator mapping interior unknowns from subdomain to global vector on that level.

After this, we have the approximate coarse solution on the second level and get the solution of problem (4.11) like in the 2-level method. The process is summarized in Algorithm 4.2. For the 2-level method, the algorithm just follows steps 1–3, 12, 13, 21, and 22. The rest of the steps are meaningful just in the multilevel case. Another difference is that in steps 5–10 and 15–19 of the multilevel method, we work with unknowns over the whole domain, while in the 2-level method, we are restricted to the interface.

Let us now recall the convergence properties of the Multilevel BDDC method for symmetric positive definite problems [28, 22]. The condition number of the Schur complement preconditioned by the Multilevel BDDC, κ, is bounded as

$$\kappa \leq \prod_{i=1}^{L-1} C_i \left(1 + \log \frac{H_i}{H_{i-1}}\right)^2, \tag{4.23}$$

Algorithm 4.2 Application of the BDDC preconditioner with L levels

```
1: distribute residual to each subdomain on the first level $r_i$ (4.6)
2: build the coarse residual on the first level $r_C$ (4.7)
3: solve subdomain problems (4.10) on the first level
4: for level $\ell = 2, \dots, L-1$ do
5:     $r^\ell = r_C^{\ell-1}$ and $u^\ell = u_C^{\ell-1}$
6:     extract local parts of residual $r_i^\ell$ (4.13)
7:     solve (4.14)
8:     get precorrected residual $\tilde{r}_{i_2}^\ell$ (4.15)
9:     solve subdomain problems (4.16)
10:    build the coarse residual $r_C^\ell$ (4.17)
11: end for
12: $r^L = r_C^{L-1}$, $u^L = u_C^{L-1}$ and solve $A^L u^L = r^L$
13: $u_C^{L-1} = u^L$
14: for level $\ell = L-1, \dots, 2$ do
15:    distribute coarse solution $u_C^\ell$ to each subdomain $u_{C_i}^\ell$ (4.18)
16:    build global interface solution $u_2^\ell$ (4.19)
17:    distribute interface solution to each subdomain $u_{i_2}^\ell$ (4.20)
18:    get interior correction on each subdomain by solving (4.21)
19:    build approximate coarse solution $u_C^{\ell-1}$ (4.22)
20: end for
21: distribute the coarse solution on the first level to each subdomain $u_{Ci}$ (4.9)
22: get preconditioned residual $u^k$ (4.11)
```

where H_i is the characteristic subdomain size on the ith level, and $H_0 = h$ the characteristic element size.

As suggested in [10, section 2.1.2], the usual bound on the error reduction of the conjugate gradient (CG) method suggests that the number of CG iterations is proportional to $\sqrt{\kappa}$ for large κ and a prescribed tolerance. Although far from describing the complex behavior of the CG method even for the symmetric positive definite case, it can still suggest that when combined with (4.23), we can expect a number of iterations proportional to

$$k \sim \prod_{i=1}^{L-1} \left(1 + \log \frac{H_i}{H_{i-1}}\right), \tag{4.24}$$

i.e., removing the power from the terms in the product. Finally, note that in the case of the 2-level method, bound (4.24) simplifies to

$$k \sim \left(1 + \log \frac{H}{h}\right). \tag{4.25}$$

Neither the BDDC theory for the Stokes problem [18, 17] nor the one for the advection-diffusion problem [29] applies to our case. Nevertheless, in section 6.1.3, we perform numerical experiments with varying H/h to investigate whether the behavior of the method agrees with (4.24).

5. Interface scaling.

Several types of interface weights have been developed for the BDDC preconditioner, each of them having its advantages and disadvantages for certain kinds of problems. In this section, we describe four types of weights used in our calculations presented in section 6.

Before we start with the description of the individual scalings, we recall that the main requirement on the matrix of weights W_i is that it forms a partition of unity [27],

$$\sum_{i=1}^{N} R_i^T W_i R_i = I, \tag{5.1}$$

where I is the identity matrix.

An important class of the interface averaging operators is represented by diagonal matrices

$$W_i = \begin{pmatrix} w_i^1 & & \\ & w_i^2 & \\ & & \ddots \end{pmatrix}, \tag{5.2}$$

where w_i^k denotes the weight for the kth (with respect to the subdomain interface) dof on the ith subdomain. In this case, the requirements are fulfilled by the following simple construction: First, every subdomain generates a nonnegative weight $\widetilde{w}_i^k$. These values are then shared with all neighboring subdomains, and the normalized weight w_i^k satisfying the partition of unity is obtained by dividing the local weight with the sum of contributions from all neighbors,

$$w_i^k = \frac{\widetilde{w}_i^k}{\sum_{j=1}^{N_S} \widetilde{w}_j^k}, \tag{5.3}$$

where N_S is the number of subdomains sharing the dof.

The first type of weights is perhaps the most standard in literature. It is based on the cardinality (*card*) of the set of subdomains sharing the dof. Hence, $\widetilde{w}_i^k = 1$, and

$$w_i^k = \frac{1}{N_S}. \tag{5.4}$$

For example, the weight is simply $w_i^k = 1/2$ if the dof is shared by two subdomains.

The second type of weights is also well established. It is derived from diagonal entries (*diag*) of the subdomain matrices A_i^ℓ. The weight in this case is defined as $\widetilde{w}_i^k = a_{ss,i}$, where s is the subdomain index corresponding to the kth interface dof. The construction is then completed by (5.3). However, in our case, a modification of these weights is required for the block of pressure unknowns, where the diagonal of A_i contains zeros. In these dofs, we switch to the cardinality scaling above, $\widetilde{w}_i^k = 1$.

An important generalization of diagonal weights is the *deluxe scaling* [9, 4, 7]. Although relatively costly, deluxe scaling leads to very robust BDDC preconditioners even for complicated problems. However, an important ingredient of this scaling is the change of basis converting primal dofs to particular unknowns. Since our implementation of BDDC is based on the saddle-point systems with constraints (4.10) without changing the basis, this type of weights is not compatible with our current implementation in *BDDCML*.

Instead, we test a type of weights inspired by the deluxe scaling and introduced in [6] as *unit load* (*ul*) scaling. Within this averaging, we compose a vector $\widetilde{\mathbf{w}}_i = (\widetilde{w}_i^1, \widetilde{w}_i^2, \ldots)^T$ from local solutions to the ith subdomain problem (4.10) under a unit load applied to a face, i.e., with right-hand side vector equal to those at the unknowns of the face. In the interface dofs corresponding to corners and edges, we again resort to the cardinality scaling, $\widetilde{w}_i^k = 1$. The construction is again completed by (5.3). The main difference from deluxe scaling is that instead of solving the local problems with the current solution in every iteration, these problems are solved just once for specific right-hand sides, and the solutions are then converted into the diagonal matrices W_i.

One final tested type of weights is a recent idea inspired by numerical schemes for flow problems. Loosely speaking, for dominant advection, it should be beneficial to consider the subdomain from which the fluid flows with a higher weight than for the one where the dof is a part of an inflow boundary. In numerical schemes, this is sometimes referred to as upwinding.

More specifically, these *upwind* weights are based on the inner product of the vector of velocity and the unit vector of outer normal to the subdomain boundary (see Figure 2),

$$p_i^k = \frac{\mathbf{n}_i^k \cdot \mathbf{v}^k}{\|\mathbf{v}^k\|_2}.$$

The values of the p_i^k are from the interval $[-1, 1]$. To derive a nonnegative weight, we map these values into the interval $[0, 1]$ by taking $\widetilde{w}_i^k = \frac{p_i^k+1}{2}$, which is used for all velocity unknowns. For pressure dofs, we again consider $\widetilde{w}_i^k = 1$. The final partition of unity is achieved again by (5.3).

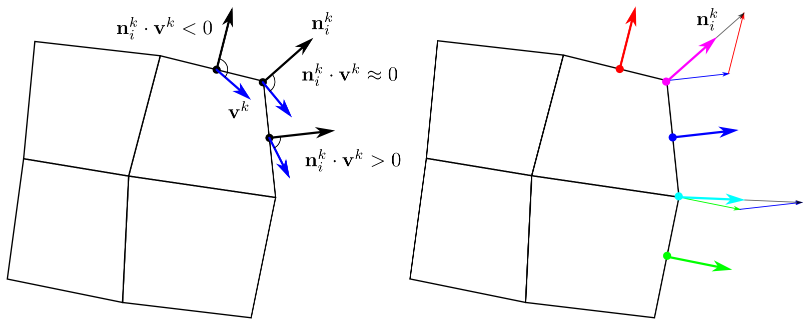

$\mathbf{n}_i^k \cdot \mathbf{v}^k < 0$
$\mathbf{n}_i^k$
$\mathbf{n}_i^k \cdot \mathbf{v}^k \approx 0$
$\mathbf{v}^k$
$\mathbf{n}_i^k \cdot \mathbf{v}^k > 0$
$\mathbf{n}_i^k$

FIG. 2. *Relation of the unit outer normal vectors to the boundary and the velocity vectors in the construction of the* upwind *interface scaling (left), and the construction of the approximate interface normals (right).*

Obviously, this kind of weights would introduce a nonlinearity by the dependence of the weight on the solution itself. We use the velocity field from the previous nonlinear iteration as an approximation of the actual velocity vector. Note that the weights are then fixed throughout the BiCGstab iterations.

The final remark is related to constructing the approximation of the unit outer normal vector at positions of the nodes, i.e., element nodes, and edge and face midsides. As illustrated in Figure 2, we take the normals of the element faces as the building block, and define the normals at vertices and edges simply as an arithmetic average of the normals of the adjacent faces.

6. Numerical results. We apply the BDDC method to steady incompressible flow governed by the Navier–Stokes equations. In section 6.1 we investigate the behavior of the 2-, 3-, and 4-level BDDC methods for the benchmark problem of a lid-driven cavity. Also the effect of different types of interface weights is examined for the 2-level method. In section 6.2, the 2-level BDDC method is applied to the industrial problem of hydrostatic bearings. All the computations are performed by a parallel finite element package written in C++ and described in [24], and using the *BDDCML* library [26] for solving the arising system of linear equations. For linearization, we use Picard iteration with the precision measured as $\left\|\mathbf{u}^k - \mathbf{u}^{k-1}\right\|_2$. The BiCGstab method preconditioned by the BDDC preconditioner is used for the linearized system with precision measured as $\left\|r^k\right\|_2 / \left\|g\right\|_2$. According to our previous experience from [24], the convergence of the BiCGstab method is comparable to that of GMRES.

6.1. Lid-driven cavity. As the benchmark problem, we consider a 3-D lid-driven cavity suggested in [32]. The computational domain is a unit cube with Dirichlet boundary conditions. A twisted unit tangential velocity vector is considered on the top wall, $\boldsymbol{u}_{top} = (1/\sqrt{3}, \sqrt{2}/\sqrt{3}, 0)$. Zero velocity is considered on the remaining five walls. The computational mesh is uniform with hexahedral elements, and it is divided into cubic subdomains with a rising number of them per domain edge (see Figure 3). Simulations were performed on the *Salomon* supercomputer at the IT4Innovations National Supercomputing Center using the same number of cores of Intel Xeon E5-2680v3 12C 2.5GHz processors as the number of subdomains. We terminate the Picard iterations after reaching the precision $\left\|\mathbf{u}^k - \mathbf{u}^{k-1}\right\|_2 \leq 10^{-5}$ or after 100 iterations, while the inner linear iterations are terminated when $\left\|r^k\right\|_2 / \left\|g\right\|_2 \leq 10^{-6}$ or after reaching the maximum number of 1000 iterations.

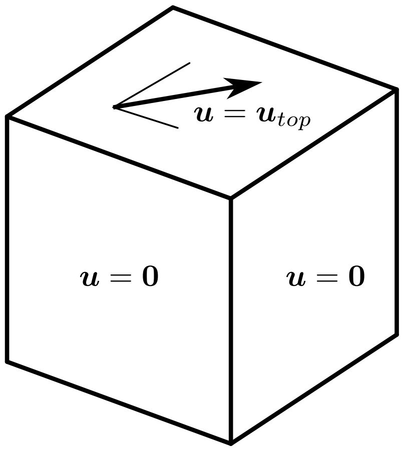
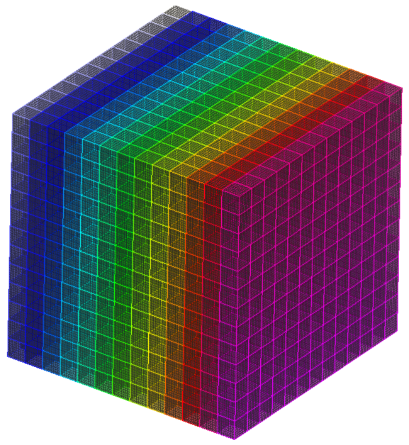

$\boldsymbol{u} = \boldsymbol{u}_{top}$
$\boldsymbol{u} = \mathbf{0}$
$\boldsymbol{u} = \mathbf{0}$


Fig. 3. *Boundary conditions (left) and an example of mesh division with* 13 *subdomains per edge (right) for the lid-driven cavity.*

6.1.1. Weak scalability. In this section, we compare the weak scalability of the multilevel BDDC method in application to 3-D lid-driven cavity. Results for the 2-, 3-, and 4-level BDDC methods for Reynolds numbers 1 and 100 are presented.

6.1.1.1. 2- and 3-level method. First, we compare the 2- and 3- level BDDC methods. The number of subdomains grows from 8 to 4,913, and there are eight elements per subdomain edge. The number of unknowns grows from 112,724 to 63,610,604, and the size of the interface problem grows from 10,324 to 10,915,264 unknowns. We consider two values of the Reynolds number, in particular, 1 and 100. We define the Reynolds number as LU/ν, where L is the length of an edge of the solution domain, U is the velocity of the top wall, and ν is the kinematic viscosity. With increasing Reynolds number, the significance of the symmetric part of (2.14) is decreasing, while the nonsymmetric part has larger influence (see [31]). For these simulations, we use cubes as subdomains on the first level (see Figure 3) for both cases. To build subdomains on the second level for the 3-level method, we use the METIS graph partitioner [16]. An example of a cluster of subdomains of the first level composing a subdomain on the second level is shown in Figure 4. We monitor the mean, maximal, and minimal number of linear iterations, the number of nonlinear iterations, the mean setup time of the BDDC preconditioner, the mean time for the Krylov subspace method, and the mean time for one iteration for the 2- and the 3-level methods for both cases of the Reynolds number. These values are presented in Tables 1–4.

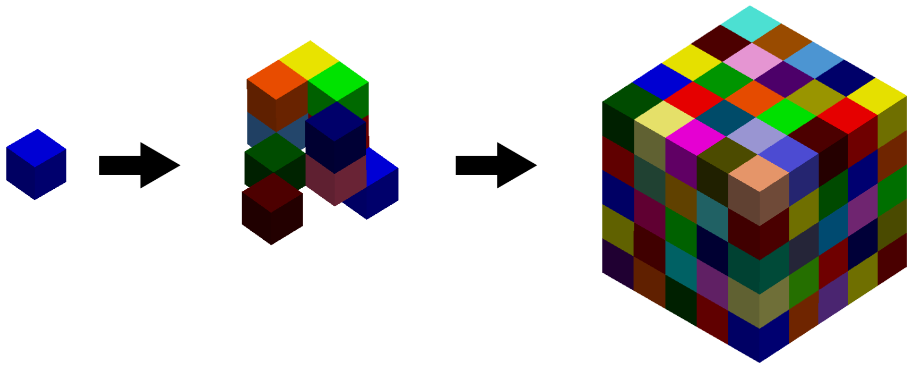


FIG. 4. *Clustering of subdomains on the second level using METIS.*

TABLE 1
$Re = 1$, 2-levels. Number of nonlinear iterations, number of linear iterations (minimal, maximal, and mean), mean setup time, time for the BiCGstab iterations, time for one linear iteration, and the total time.

nproc	nonl	Linear solve			Time [s]		
		min	max	Mean	Setup	BiCGstab iter (one iter)	Total
8	4	8.5	9	8.9	2.33	0.58 (0.07)	2.91
27	4	10.5	11.5	11.3	3.16	0.84 (0.07)	4.00
64	4	11.5	11.5	11.5	3.71	1.24 (0.11)	4.95
125	4	12.5	12.5	12.5	3.98	1.50 (0.12)	5.48
216	4	11.5	12	11.9	4.80	1.91 (0.16)	6.71
343	4	12.5	12.5	12.5	5.39	2.46 (0.20)	7.85
512	4	12.5	12.5	12.5	6.31	3.04 (0.24)	9.35
729	4	13	13	13	8.79	5.45 (0.42)	14.2
1000	4	12.5	12.5	12.5	11.3	7.00 (0.56)	18.3
1331	4	13	13	13	15.1	10.3 (0.79)	25.4
1728	4	12.5	12.5	12.5	20.7	14.0 (1.12)	34.7
2197	4	13	13	13	31.0	22.0 (1.69)	53.0
2744	4	12.5	12.5	12.5	42.8	28.3 (2.26)	71.1
3375	4	13	13	13	56.8	40.9 (3.15)	97.7
4096	4	12.5	12.5	12.5	79.6	21.6 (1.73)	101.2
4913	4	13	13	13	111.8	29.4 (2.26)	141.2

We can see that for the case with Reynolds number $Re = 1$, the numbers of nonlinear iterations stay the same for almost all cases for both 2- and 3-level methods.

Table 2

$Re = 1$, 3-levels. Number of nonlinear iterations, number of linear iterations (minimal, maximal, and mean), mean setup time, time for the BiCGstab iterations, time for one linear iteration, and the total time.

nproc	nonl	Linear solve			Time [s]		
		min	max	Mean	Setup	BiCGstab iter (one iter)	Total
8	4	10	10	10	2.33	0.65 (0.07)	2.98
27	4	12.5	12.5	12.5	3.18	0.93 (0.07)	4.11
64	4	13.5	13.5	13.5	3.71	1.40 (0.10)	5.11
125	4	17.5	17.5	17.5	3.95	1.94 (0.11)	5.89
216	4	15	16	15.3	4.43	2.05 (0.13)	6.48
343	4	19.5	19.5	19.5	4.41	2.74 (0.14)	7.15
512	4	17.5	17.5	17.5	4.72	2.78 (0.16)	7.50
729	4	21.5	22.5	21.8	5.20	4.03 (0.18)	9.23
1000	4	16.5	18	17.6	5.10	3.78 (0.21)	8.88
1331	5	18	18.5	18.4	5.25	7.87 (0.43)	13.1
1728	4	17.5	20.5	19.8	6.45	6.61 (0.33)	13.1
2197	4	17.5	20.5	18.1	6.70	7.14 (0.39)	13.8
2744	4	18.5	18.5	18.5	8.86	9.25 (0.50)	18.1
3375	5	17.5	19.5	18.7	8.40	10.7 (0.57)	19.4
4096	4	19.5	20.5	20.3	10.8	8.55 (0.42)	19.4
4913	4	19.5	19.5	19.5	13.2	9.80 (0.50)	23.0

Table 3

$Re = 100$, 2-levels. Number of nonlinear iterations, number of linear iterations (minimal, maximal, and mean), mean setup time, time for the BiCGstab iterations, time for one linear iteration, and the total time.

nproc	nonl	Linear solve			Time [s]		
		min	max	Mean	Setup	BiCGstab iter (one iter)	Total
8	18	9.5	14	13.8	2.22	0.86 (0.06)	2.98
27	19	10.5	17.5	15.1	3.16	1.12 (0.07)	4.28
64	20	11.5	18.5	17.2	3.64	1.84 (0.11)	5.48
125	21	13	19	17.0	3.94	2.04 (0.12)	5.98
216	21	12	16.5	15.4	4.84	2.49 (0.16)	7.33
343	21	11	16	15.3	5.76	3.08 (0.20)	8.84
512	22	12.5	15	14.4	6.30	3.56 (0.25)	9.86
729	22	12.5	14.5	14.0	8.58	5.28 (0.38)	13.9
1000	22	12	14	13.0	11.2	7.26 (0.56)	18.5
1331	22	13	14.5	14.4	15.1	11.7 (0.81)	26.8
1728	23	12.5	13.5	13.0	20.7	14.6 (1.12)	35.3
2197	23	13	13.5	13.4	31.7	22.6 (1.69)	54.3
2744	23	12.5	14	13.8	43.3	31.1 (2.25)	74.4

The number of linear iterations appears to have a similar, slightly increasing trend. From the setup times, we can see that for the 2-level method, the setup time rapidly increases with the number of processors, whereas for the 3-level method, the increase of the setup time is significantly slower. From the mean times in Tables 1 and 2, we can see that the 3-level method is faster than the 2-level one, especially for a bigger number of processors. Also, there is not such a rapid increase of computational time for the 3-level method as for the 2-level case.

One can observe that while the 3-level method gets considerably faster than the 2-level method, the computational times are not perfectly weakly scalable even for the 3-level case. While this can be clearly attributed to the coarse problem solves for the 2-level method, it is likely the global communication related to propagating the higher-level solutions and residuals what worsens the weak scalability also for the 3-level method.

Table 4

$Re = 100$, 3-levels. Number of nonlinear iterations, number of linear iterations (minimal, maximal, and mean), mean setup time, time for the BiCGstab iterations, time for one linear iteration, and the total time.

nproc	nonl	Linear solve			Time [s]		
		min	max	Mean	Setup	BiCGstab iter (one iter)	Total
8	18	10.5	16	14.7	2.30	0.94 (0.06)	3.24
27	19	12.5	18.5	16.4	3.15	1.21 (0.07)	4.36
64	20	13.5	20.5	18.1	3.68	1.89 (0.10)	5.57
125	100	17.5	27.5	25.4	3.93	2.80 (0.11)	6.73
216	26	17.5	22.5	22.2	4.29	2.99 (0.13)	7.28
343	100	19.5	34.5	30.1	4.41	4.25 (0.14)	8.66
512	24	17.5	21.5	21.3	4.82	3.62 (0.17)	8.44
729	100	22.5	31.5	28.2	5.13	5.48 (0.19)	10.6
1000	100	18.5	16	29.7	4.85	6.52 (0.22)	11.4
1331	30	18.5	30.5	23.1	5.34	6.12 (0.26)	11.5
1728	28	17.5	29.5	25.7	6.34	8.50 (0.33)	14.8
2197	100	17.5	32.5	29.2	6.97	11.6 (0.40)	18.6
2744	100	18.5	32.5	27.7	9.07	13.7 (0.49)	22.8

Let us look closer at the case of $Re = 100$. We can see that for the 2-level method, the number of nonlinear iterations is slightly increasing with the rising number of processors. For the 3-level method, the number of nonlinear iterations is similar in some cases, and in some cases, it reaches the limit of 100 nonlinear iterations. Looking closer at the output of these simulations, we can see that after the maximum of 20 nonlinear iterations, the residual $\left\|\mathbf{u}^k - \mathbf{u}^{k-1}\right\|_2$ oscillates between $5 \cdot 10^{-5}$ and 10^{-4}, but never drops below the prescribed 10^{-5}. For the setup time, we can see similar behavior as in the case with $Re = 1$, and the same stands for the time for solving the linear problem and the time for 1 iteration. If we compare the results for $Re = 1$ and $Re = 100$, a significant difference is just in the number of nonlinear iterations. This can be attributed to the bigger influence of the nonsymmetric part of (2.14).

Since we are mainly interested in the efficiency of the BDDC method and the linear solver, we focus on the mean number of linear iterations over all nonlinear iterations, the mean setup time for preparing the BDDC preconditioner, the mean time for solving the linear problem, and the mean time for one linear iteration. Comparisons of the behavior of the 2- and 3-level methods for these parameters with a rising number of processors for both cases of the Reynolds number are presented in Figures 5 and 6.

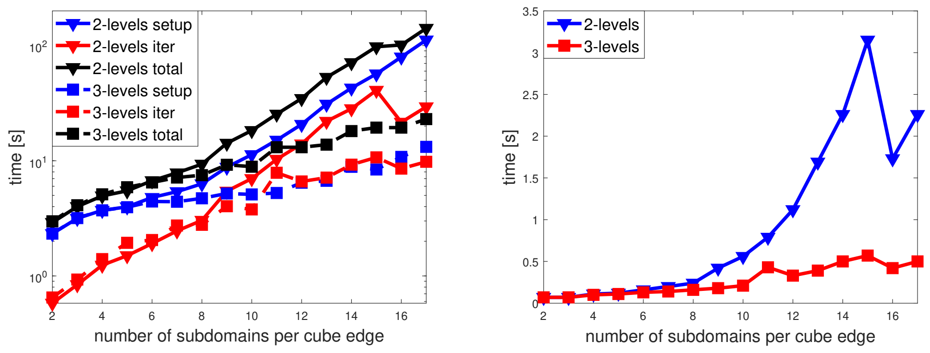


Fig. 5. *$Re = 1$. Mean time for setup, mean time for the BiCGstab iterations, and mean total time (left). Mean time for one iteration (right).*

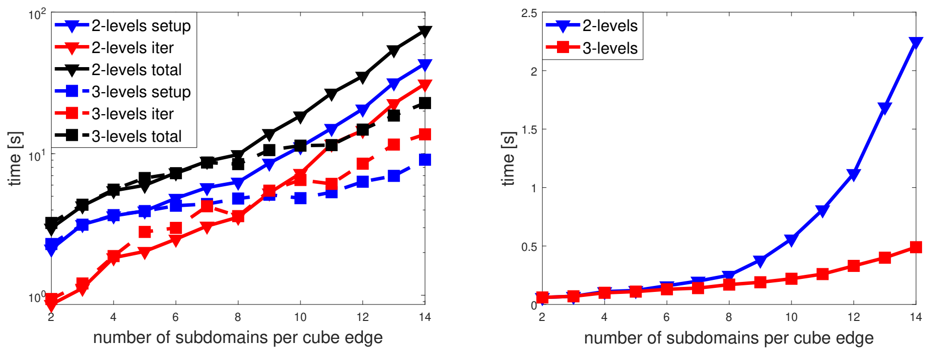


FIG. 6. *Re* = 100. *Mean time for setup, mean time for the BiCGstab iterations, and mean total time (left). Mean time for one iteration (right).*

In Figure 5, there is a large drop of time for one iteration for the 2-level method. While reproducible, we are not able to satisfactorily explain this phenomenon.

6.1.1.2. 3- and 4-level method. Next, we compare the 3- and 4-level methods. The number of subdomains goes from 8 up to 1,728 with 12 elements per subdomain edge. Because the number of elements per subdomain edge is different, we had to also recompute the results for the 3-level method, so the results in this section are different from those in section 6.1.1.1. As a consequence, we now have larger local problems. The number of unknowns ranges from 368,572 to 75,461,332, and the size of the interface problem grows from 22,972 to 8,600,372 unknowns. Here we form subdomains on coarse levels for the 3-level method in the same way as in the previous subsection, i.e., using the METIS graph partitioner. Figure 7 shows the potentially complex subdomains on higher levels resulting from employing METIS. However, for the 4-level method, we form regular subdomains on all levels. Thus, on the first level, we have cubes as subdomains, on the second level we join a line of cubes into a cuboid, and on the third level we join plate of cuboids to form the subdomains (see Figure 8). The reason for this explicit creation of the subdomains is that the simulations using METIS did not converge for the 4-level method. This seems to be related to our previous findings about a significantly better convergence of the BDDC method with regular subdomains (see [13] for more details). We set the value of the Reynolds number to 1, and we compare the same parameters as in the previous case. We again report the number of linear iterations (maximal, minimal, and mean) over all nonlinear iterations, the mean setup time of the BDDC preconditioner, the mean time for solving the linear problem, and the mean time for one linear iteration. All these values are in Tables 5 and 6.

We can see that the numbers of nonlinear iterations remain similar as in the previous subsection for almost all cases for the 3- and 4-level methods. However, the numbers of linear iterations are rapidly increasing for the 4-level method. If we look at the setup times, we can see that the times for the 3- and 4-level methods seem to be slowly increasing with the number of subdomains, with a similar rate. A big difference is in the total time for solving the linear problem. Due to the rapid increase in the mean number of linear iterations, the total time for solving the linear problem by the 4-level method gets much larger than for the 3- and even the 2-level methods although each iteration is cheaper for larger numbers of subdomains. Thus further worsening of the approximation of the preconditioner by introducing the fourth level

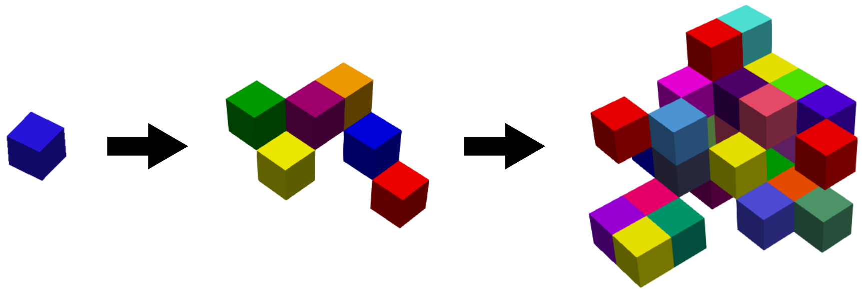


Fig. 7. *Decomposition by METIS for the* 4*-level BDDC method. Subdomains on level* 1 *(left), level* 2 *(center), and level* 3 *(right).*

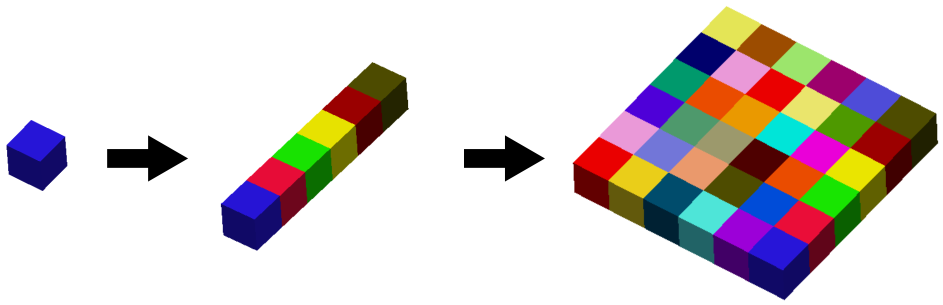


Fig. 8. *Regular decomposition for the* 4*-level BDDC method. Subdomains on level* 1 *(left), level* 2 *(center), and level* 3 *(right).*

is not beneficial for the problem of our experiment.

Again, we mainly monitor the mean number of linear iterations over all nonlinear iterations, the mean setup time for preparing the BDDC preconditioner, the mean time for solving the linear problem, and the mean time for one linear iteration. The comparison of the behavior of the 3- and 4-level methods for these parameters with a rising number of processors is presented in Figure 9.

In Figure 9, there are once again some wild rises and falls which we cannot satisfactorily explain, but at least for the 3-level method, it could be caused by certain more suitable decompositions by the METIS graph partitioner on higher levels. We again see a suboptimal weak scalability, probably caused by the coarse problem global communication.

6.1.2. Interface scalings. In this section, we compare the behavior of the 2-level BDDC method for different types of interface weights described in section 5, namely the cardinality scaling (*card*), scaling by diagonal stiffness (*diag*), scaling weights from unit load (*ul*), and the proposed *upwind* weights. For these simulations, the number of subdomains is 125 with eight elements per subdomain edge. We consider Reynolds numbers 100 and 200. Also the division into subdomains remains. We once again monitor the mean, maximal, and minimal number of linear iterations, the number of nonlinear iterations, the mean setup time of the BDDC preconditioner,

Table 5

$Re = 1$, 3-levels. Number of nonlinear iterations, number of linear iterations (minimal, maximal, and mean), mean setup time, time for the BiCGstab iterations, time for one linear iteration, and the total time.

nproc	nonl	Linear solve			Time [s]		
		min	max	Mean	Setup	BiCGstab iter (one iter)	Total
8	4	14	14	14	13.57	4.06 (0.29)	17.6
27	4	16.5	16.5	16.5	15.92	5.78 (0.35)	21.7
64	4	16	16	16	18.03	8.17 (0.51)	26.2
125	6	28.5	30.5	29.7	18.24	15.06 (0.51)	33.3
216	16	26.5	27.5	27.3	19.77	16.16 (0.59)	35.9
343	8	33.5	36	34	33.46	29.95 (0.88)	63.4
512	5	38.5	41.5	40.3	25.18	41.85 (1.04)	67.0
729	20	32	36	34.1	21.28	28.05 (0.82)	49.3
1000	4	24	24.5	24.4	26.08	26.46 (1.08)	52.5
1331	4	23.5	26	24.1	25.77	27.37 (1.14)	53.1
1728	4	25.5	25.5	25.5	26.36	30.34 (1.19)	56.7

Table 6

$Re = 1$, 4-levels. Number of nonlinear iterations, number of linear iterations (minimal, maximal, and mean), mean setup time, time for the BiCGstab iterations, time for one linear iteration, and the total time.

nproc	nonl	Linear solve			Time [s]		
		min	max	Mean	Setup	BiCGstab iter (one iter)	Total
8	5	13.5	13.5	13.5	13.49	3.90 (0.29)	17.4
27	4	16.5	17.5	16.8	15.92	5.86 (0.35)	21.8
64	4	22.5	23.5	23.3	22.80	20.84 (0.89)	43.6
125	4	33	34.5	34.1	18.16	17.28 (0.51)	35.4
216	5	47.5	49.5	48.3	36.14	48.95 (1.01)	85.1
343	22	66.5	73	67.6	22.04	54.61 (0.81)	76.7
512	12	97	100.5	100.2	24.63	102.49 (1.02)	127.1
729	11	122	145	135.2	24.58	140.49 (1.04)	165.1
1000	4	175.5	189	182.4	31.10	127.39 (0.70)	158.5
1331	12	207.5	238	218	25.14	241.14 (1.11)	266.3
1728	4	250.5	282	267.6	57.55	223.12 (0.83)	280.7

the mean time for the Krylov subspace method, and the mean time for one iteration. These values are presented in Tables 7 and 8.

Table 7

$Re = 100$. Number of nonlinear iterations, number of linear iterations (minimal, maximal, and mean), mean setup time, time for the BiCGstab iterations, time for one linear iteration, and the total time.

Weights type	nonl	Linear solve			Time [s]		
		min	max	Mean	Setup	BiCGstab iter (one iter)	Total
card	23	13	19	17	4.18	1.99 (0.12)	6.17
diag	23	13	18	15.1	3.85	1.78 (0.12)	5.63
ul	23	12.5	14.5	13.5	4.22	1.60 (0.12)	5.82
upwind	23	13	14.5	14.3	4.05	1.69 (0.12)	5.74

The maximal number of nonlinear iteration was again set to 100, which was reached for several cases (Table 8). For those cases, the error $\left\|\mathbf{u}^k - \mathbf{u}^{k-1}\right\|_2$ oscillated between $5 \cdot 10^{-5}$ and 10^{-4}, but never dropped below 10^{-5}.

From Tables 7 and 8, we can see that there is no significant difference in using different weights for Reynolds number 100. However, for Reynolds number 200, we can observe a large difference in the mean number of linear iterations. Consequently,

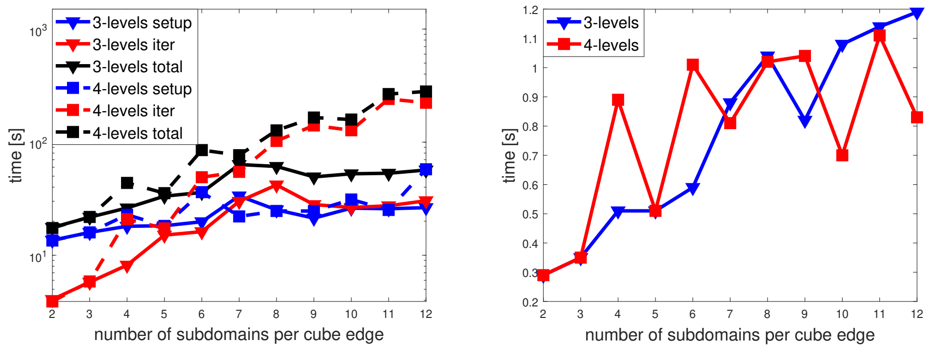


FIG. 9. *Re* = 1. *Mean time for setup, mean time for the BiCGstab iterations, and mean total time (left). Mean time for one iteration (right).*

TABLE 8
Re = 200. *Number of nonlinear iterations, number of linear iterations (minimal, maximal, and mean), mean setup time, time for the BiCGstab iterations, time for one linear iteration, and the total time.*

Weights type	nonl	Linear solve			Time [s]		
		min	max	Mean	Setup	BiCGstab iter (one iter)	Total
card	100	12	84.5	68.8	3.84	8.03 (0.12)	11.03
diag	33	12	77.5	69.5	3.87	8.09 (0.12)	11.96
ul	100	12	81	54.4	4.19	6.35 (0.12)	10.54
upwind	48	22	80.5	28.2	4.28	3.31 (0.12)	7.59

the time for linear iterations is also different, although the time per one iteration stays the same. Based on these tables, we can conclude that the upwind weights may lead to a significant reduction of the number of linear iterations, as well as of the computational time.

6.1.3. Experimental H/h dependence. Next, we perform an experimental test of convergence of the BDDC method with respect to changing the H/h ratio, i.e., the ratio of the characteristic size of a subdomain to the characteristic size of an element. For cubic subdomains, H/h represents the number of elements along a subdomain edge. In these simulations, we have 64 subdomains and vary the number of elements per subdomain edge from 2 to 24. The Reynolds number was set to 1, and we compare the behavior of the 2- and 3-level methods. The results are presented in Figure 10.

From Figure 10, we can see that with the growing H/h ratio, the number of BiCGstab iterations approximately follows the logarithmic dependence (4.24) for both the 2- and, with slight irregularities, the 3-level methods. Although a theoretical insight is not available for this class of problems, the dependence seems to resemble the behavior for problems with a symmetric positive definite matrix.

6.2. Hydrostatic bearings. Our research in this field has been motivated by simulations of oil flow in hydrostatic bearings. These are parts of production machines that lubricate moving parts of the machines on a thin layer of oil to provide low friction. Oil is pressurized to a few MPa, and it flows through the input of the hydrostatic bearing into a so-called hydrostatic cell. Then it flows out through a very thin (few tens of micrometers) throttling gap.

Numerical simulation of this problem comes with several challenges, like a large

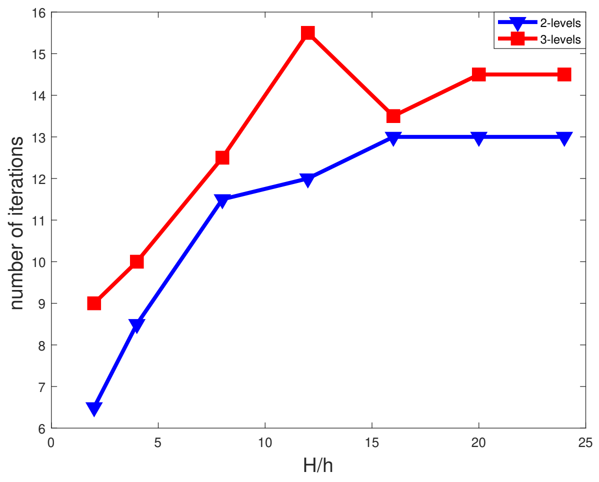


FIG. 10. *Dependence of the number of iterations on the H/h ratio.*

pressure gradient realized in the throttling gap, small thickness of the throttling gap, and movement of the bearing. A major issue is the emergence of finite elements with bad aspect ratio in the throttling gap; see Figure 11. During our research, we have gradually addressed most of these issues. In [13], we studied the influence of the interface geometry on the convergence of the BDDC solver. The convergence of BDDC largely improves by using a mesh partitioner preferring straight interfaces. A simulation on a real geometry of a bearing from a production machine was recently presented in [15].

In the next part of this paper, we describe two other cases of simulations of hydrostatic bearings. Input parameters of both simulations, such as the input mean velocity $|\boldsymbol{u}_{input\ mean}|$, the velocity of the bottom wall $|\boldsymbol{u}_{bottom\ wall}|$, the dynamic viscosity μ, and the height of the throttling gap h, are summarized in Table 9. In both simulations, we use the 2-level BDDC method with cardinality weights.

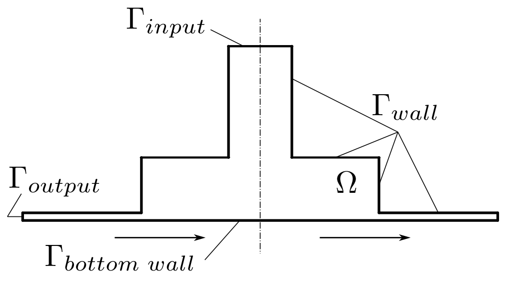


FIG. 11. *A scheme of a cross-section of the solution domain for hydrostatic bearings with boundary conditions.*

6.2.1. Case 1. At the beginning of our research, we dealt with simulations of hydrostatic bearings with an artificially magnified throttling gap and geometry axially symmetric along the vertical axis (see Figure 12). However, by considering a linear sliding of the bearing, the boundary conditions lack axial symmetry, and the problem

TABLE 9
Parameters of simulations for both cases of the hydrostatic bearings.

Case	$\lvert \boldsymbol{u}_{input\ mean} \rvert$ [m/s]	h [μm]	μ [Ns/m^2]	$\lvert \boldsymbol{u}_{bottom\ wall} \rvert$ [m/s]
1	0.3	1000	0.809	1
2	0.3	200	0.5	0

can no longer be solved as axially symmetric. From the beginning, we aimed at reaching the real-scale height of the throttling gap. Unfortunately, we were able to reach only a throttling gap of the height 1 mm, which is 10 times more than the real gap, but it still gave us an initial insight into the flow behavior during the movement of the bearing. However, convergence deteriorated quickly with decreasing the throttling gap height. For these calculations, we have created meshes in the Gmsh software [12] and divided them into 32 subdomains using the METIS partitioner. The problem contains almost 609 thousand unknowns, with 93 thousand unknowns at the interface. Our simulations were performed on the SGI Altix supercomputer at the supercomputing center of the Czech Technical University in Prague using 32 cores of Intel Xeon 2.66 GHz processors. Precision for the Picard iteration was set to $\left\| \mathbf{u}^k - \mathbf{u}^{k-1} \right\|_2 \leq 10^{-5}$, and the linear iterations were terminated when $\left\| r^k \right\|_2 / \left\| g \right\|_2 \leq 10^{-6}$ or after reaching the maximum number of 100 iterations. This set of prescribed precisions was used for all presented calculations of hydrostatic bearings. The solution required 12 nonlinear iterations, each of them performing on average 511 linear iterations. The solution is presented in Figure 12.

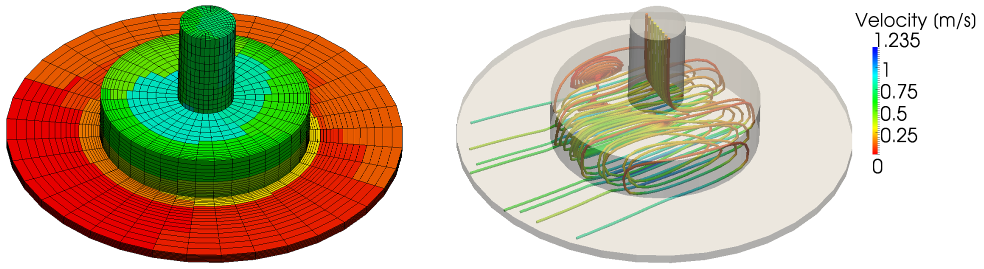


FIG. 12. *Case* 1. *The solution domain decomposed into* 32 *subdomains (left) and the streamtraces colored by the magnitude of velocity (right).*

6.2.2. Case 2. In the next phase, we switched to a rectangular geometry of the hydrostatic bearing and tried to solve the throttling-gap-height issue by using a very fine mesh and decomposition into hundreds of subdomains (precisely 1200), still using the METIS partitioner. We were able to reach the height of 100 μm of the throttling gap, which is the real height for certain cases, but only without the motion of the bearing. Hence, this solution has given us a better idea about the pressure values and development inside the hydrostatic bearing. The mesh was also generated in the Gmsh software. It has about 13 million unknowns, with 2.7 million unknowns at the interface. The simulations were performed on the *Salomon* supercomputer using 1,200 cores. Precisions of the iterations were set as above. The presented calculation converged after three nonlinear iterations with an average of 754 linear iterations for each of them. The solution is shown in Figure 13. Recall that there is no sliding considered in this case.

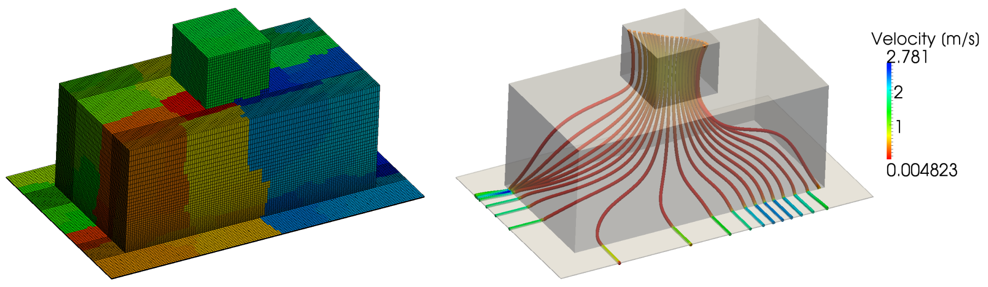


FIG. 13. *Case* 2. *The solution domain decomposed into* 1200 *subdomains (left) and the streamtraces colored by the magnitude of velocity (right).*

7. Conclusions.

We have formulated the Multilevel BDDC preconditioner for linear systems with nonsymmetric matrices. The preconditioner is applied to systems arising from the discretization of the incompressible Navier–Stokes equations. The nonlinear problem is linearized by the Picard iteration, which leads to a sequence of linear saddle-point systems. The method has been implemented into our open source parallel library *BDDCML*.

As a benchmark, we have used the 3-D lid-driven cavity problem, and we have explored the behavior of the 2-, 3-, and 4-level BDDC methods for this problem. We have focused mainly on the number of linear iterations and mean times for the setup of the preconditioner and for the iterations of the Krylov subspace method. The performance was tested on up to 5 thousand CPU cores. The 3-level method has shown remarkable speedup compared to the 2-level method, especially for large numbers of subdomains where the coarse problem significantly grows. However, switching to the 4-level method has not brought us an overall improvement. Although each iteration was cheaper than in the 3-level case, the method has required a very large number of iterations and thus performed even worse than the 2-level method.

A new type of weights inspired by numerical schemes for flow problems has been proposed. This upwind-based scaling has been compared with three other suitable interface scaling types. Our results have suggested that with a growing Reynolds number, the importance of the scaling type increases, and the upwind weights provided promising results.

We have performed experiments investigating the behavior of the 2-level and 3-level BDDC method with respect to the H/h ratio. Although theoretical estimates are not available for this class of problem, the numbers of BiCGstab iterations seem to confirm the logarithmic behavior proven for symmetric positive definite problems.

Finally, we have applied the BDDC methodology to an industrial problem from engineering, namely the flow in hydrostatic bearings. We have presented two cases of these simulations. By solving a number of issues over several years, we have managed to perform challenging computations with a real geometry of the bearing by means of the 2-level BDDC method.

Several topics are left for future investigation. A better understanding of the effect of forming subdomains on higher levels is still required, especially for applying the Multilevel BDDC method to the hydrostatic bearings problem. This could also help to explain the rapidly worsening behavior of the 4-level method for the cavity problem.

More experiments are also required for confirming the benefits of the upwind-based interface scaling.

REFERENCES

[1] Y. Achdou, P. Le Tallec, F. Nataf, and M. Vidrascu, *A domain decomposition preconditioner for an advection–diffusion problem*, Comput. Methods Appl. Mech. Engrg., 184 (2000), pp. 145–170, https://doi.org/https://doi.org/10.1016/S0045-7825(99)00227-3.

[2] Y. Achdou and F. Nataf, *A Robin-Robin preconditioner for an advection-diffusion problem*, C. R. Acad. Sci. Paris Sér. I Math., 325 (1997), pp. 1211–1216, https://doi.org/https://doi.org/10.1016/S0764-4442(97)83556-2.

[3] S. Badia, A. F. Martín, and J. Principe, *Multilevel balancing domain decomposition at extreme scales*, SIAM J. Sci. Comput., 38 (2016), pp. C22–C52, https://doi.org/10.1137/15M1013511.

[4] L. Beirão da Veiga, L. S. Pavarino, S. Scacchi, O. B. Widlund, and S. Zampini, *Isogeometric BDDC preconditioners with deluxe scaling*, SIAM J. Sci. Comput., 36 (2014), pp. A1118–A1139, https://doi.org/10.1137/130917399.

[5] D. Boffi, F. Brezzi, and M. Fortin, *Mixed Finite Element Methods and Applications*, Springer Ser. Comput. Math. 44, Springer, Heidelberg, 2013.

[6] M. Čertíková, J. Šístek, and P. Burda, *Different approaches to interface weights in the BDDC method in 3D*, in Programs and Algorithms of Numerical Mathematics 17, J. Chleboun, P. Přikryl, K. Segeth, J. Šístek, and T. Vejchodský, eds., Acad. Sci. Czech Repub. Inst. Math., Prague, 2015, pp. 47–57.

[7] C. Dohrmann and O. B. Widlund, *A BDDC algorithm with deluxe scaling for three-dimensional H(curl) problems*, Comm. Pure Appl. Math., 69 (2016), pp. 745–770.

[8] C. R. Dohrmann, *A preconditioner for substructuring based on constrained energy minimization*, SIAM J. Sci. Comput., 25 (2003), pp. 246–258, https://doi.org/10.1137/S1064827502412887.

[9] C. R. Dohrmann and O. B. Widlund, *Some recent tools and a BDDC algorithm for 3D problems in H(curl)*, in Domain Decomposition Methods in Science and Engineering XX, R. Bank, M. Holst, O. Widlund, and J. Xu, eds., Lecture Notes in Comput. Sci. Eng. 91, Springer, Heidelberg, 2013, pp. 15–25, https://doi.org/10.1007/978-3-642-35275-1_2.

[10] 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.

[11] C. Farhat, M. Lesoinne, and K. Pierson, *A scalable dual-primal domain decomposition method*, Numer. Linear Algebra Appl., 7 (2000), pp. 687–714.

[12] C. Geuzaine and J.-F. Remacle, *Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities*, Internat. J. Numer. Methods Engrg., 79 (2009), pp. 1309–1331.

[13] M. Hanek, J. Šístek, and P. Burda, *The effect of irregular interfaces on the BDDC method for the Navier-Stokes equations*, in Domain Decomposition Methods in Science and Engineering XXIII, C.-O. Lee, X.-C. Cai, D. E. Keyes, H. H. Kim, A. Klawonn, E.-J. Park, and O. B. Widlund, eds., Lect. Notes Comput. Sci. Eng., 116, Springer, Cham, 2017, pp. 171–178.

[14] M. Hanek, J. Šístek, and P. Burda, *An application of the BDDC method to the Navier-Stokes equations in 3-D cavity*, in Programs and Algorithms of Numerical Mathematics 17, J. Chleboun, P. Přikryl, K. Segeth, J. Šístek, and T. Vejchodský, eds., Acad. Sci. Czech Repub. Inst. Math., Prague, 2015, pp. 77–85.

[15] M. Hanek, J. Šístek, and P. Burda, *Parallel domain decomposition solver for flows in hydrostatic bearings*, in Topical Problems of Fluid Mechanics 2018, D. Šimurda and T. Bodnár, eds., Institute of Thermomechanics AS CR, 2018, pp. 137–144.

[16] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392, https://doi.org/10.1137/S1064827595287997.

[17] J. Li and X. Tu, *A nonoverlapping domain decomposition method for incompressible Stokes equations with continuous pressures*, SIAM J. Numer. Anal., 51 (2013), pp. 1235–1253, https://doi.org/10.1137/120861503.

[18] J. Li and O. B. Widlund, *BDDC algorithms for incompressible Stokes equations*, SIAM J. Numer. Anal., 44 (2006), pp. 2432–2455, https://doi.org/10.1137/050628556.

[19] J. Li and O. B. Widlund, *FETI-DP, BDDC, and block Cholesky methods*, Internat. J. Numer.

Methods Engrg., 66 (2006), pp. 250–271.

[20] J. Mandel and C. R. Dohrmann, *Convergence of a balancing domain decomposition by constraints and energy minimization*, Numer. Linear Algebra Appl., 10 (2003), pp. 639–659.

[21] J. Mandel, C. R. Dohrmann, and R. Tezaur, *An algebraic theory for primal and dual substructuring methods by constraints*, Appl. Numer. Math., 54 (2005), pp. 167–193.

[22] J. Mandel, B. Sousedík, and C. R. Dohrmann, *Multispace and multilevel BDDC*, Computing, 83 (2008), pp. 55–85.

[23] C. Pechstein and C. R. Dohrmann, *A unified framework for adaptive BDDC*, Electron. Trans. Numer. Anal., 46 (2017), pp. 273–336.

[24] J. Šístek and F. Cirak, *Parallel iterative solution of the incompressible Navier-Stokes equations with application to rotating wings*, Comput. & Fluids, 122 (2015), pp. 165–183.

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

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

[27] A. Toselli and O. B. Widlund, *Domain Decomposition Methods—Algorithms and Theory*, Springer Ser. Comput. Math. 34, Springer-Verlag, Berlin, 2005.

[28] X. Tu, *Three-level BDDC in three dimensions*, SIAM J. Sci. Comput., 29 (2007), pp. 1759–1780, https://doi.org/10.1137/050629902.

[29] X. Tu and J. Li, *A balancing domain decomposition method by constraints for advection-diffusion problems*, Commun. Appl. Math. Comput. Sci., 3 (2008), pp. 25–60.

[30] X. Tu and J. Li, *BDDC for nonsymmetric positive definite and symmetric indefinite problems*, in Domain Decomposition Methods in Science and Engineering XVIII, M. Bercovie, M. Gander, R. Kornhuber, and O. Widlund, eds., Lect. Notes Comput. Sci. Eng. 70, Springer, Berlin, 2009, pp. 75–86.

[31] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. and Stat. Comput., 13 (1992), pp. 631–644, https://doi.org/10.1137/0913035.

[32] A. J. Wathen, D. Loghin, D. A. Kay, H. C. Elman, and D. J. Silvester, *A new preconditioner for the Oseen equations*, in Numerical Mathematics and Advanced Applications, F. Brezzi, A. Buffa, S. Corsaro, and A. Murli, eds., Milano, 2003, Springer Italia, Milan, pp. 979–988, Proceedings of ENUMATH 2001, Ischia, Italy.

[33] M. Yano, *Massively Parallel Solver for the High-order Galerkin Least-squares Method*, Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 2009.