**CZECH TECHNICAL
UNIVERSITY
IN PRAGUE**

**FACULTY
OF MECHANICAL
ENGINEERING**

DOCTORAL THESIS STATEMENT

Czech Technical University in Prague
Faculty of Mechanical Engineering

Numerical solution of the incompressible flow using a domain decomposition method

Doctoral thesis statement

Ing. Martin Hanek

Program
Mechanical Engineering

Study field
Mathematical and Physical Engineering

Supervisor
Prof. RNDr. Pavel Burda, CSc.

Co-supervisor
Ing. Jakub Šístek, Ph.D.

Prague 2022

The dissertation was produced during a Ph.D. study in a combined form at the Department of Technical Mathematics of the Faculty of Mechanical Engineering of the Czech Technical University in Prague.

Ph.D. candidate:	Ing. Martin Hanek	
Advisor:	Prof. RNDr. Pavel Burda CSc.	
Opponents:	Doc. Ing. Marek Brandner, Ph.D.	ZČU Plzeň
	Doc. Ing. Dalibor Lukáš, Ph.D.	VŠB Ostrava
	Doc. RNDr. Jan Chleboun, CSc.	ČVUT Praha

The doctoral thesis statement was distributed on: ....

The defence of the doctoral thesis will be held on .... at .... a.m./p.m. before the Board for the Defence of the Doctoral Thesis in the branch of study Mathematical and Physical Engineering in the meeting room No. .... of the Faculty of Mechanical Engineering of the CTU in Prague.

Those interested may get acquainted with the doctoral thesis concerned at the Department for Science and Research of the Faculty of Mechanical Engineering of the CTU in Prague, Technicka 4, Prague 6.

Abstrakt

V této disertační práci se zabývám numerickým řešením Navierových-Stokesových (N-S) rovnic pro stacionární nestlačitelné proudění pomocí metody Balancing Domain Decomposition by Constraints (BDDC). Pro diskretizaci N-S rovnic je použita metoda konečných prvků (MKP).

V práci je představena víceúrovňová metoda BDDC, která je vhodná pro řešení velkých soustav lineárních algebraických rovnic. Tato metoda a její teorie je dobře známa pro Poissonovu úlohu, úlohy lineární pružnosti a proudění popsané Stokesovými rovnicemi. Práce je věnována rozšíření této víceúrovňové metody na nesymetrickou úlohu vznikající diskretizací N-S rovnic. Pro tuto formu BDDC jsou také diskutovány různé vážící operátory, včetně vlastního vycházejícího z "upwind" schématu, a vhodného pro úlohy proudění. Dále je představen vlastní geometrický dělič sítě, který je vhodný pro tento typ úloh, speciálně pro úlohy s dimenzionálně komplexní geometrií.

Všechny nové přístupy prezentované v této práci jsou podrobeny numerickým experimentům na různých úlohách. Konkrétně jsou tu prezentovány výsledky slabé škálovatelnosti víceúrovňové metody BDDC pro proudění v 3D kavitě, výsledky pro test vlivu rozhraní mezi podoblastmi ve 2D a 3D na úloze zužujícího se kanálku a výsledky využívající získané poznatky pro simulaci proudění oleje uvnitř hydrostatického ložiska.

Abstract

The Ph.D. thesis is devoted to numerical simulations of the Navier-Stokes (N-S) equations for stationary incompressible flow using Balancing Domain Decomposition by Constraints (BDDC) method. The Finite Element Method (FEM) is used for the discretization of the N-S equations.

In the thesis, the multilevel BDDC method is introduced which is suitable for solving the large system of linear algebraic equations. This method and its theory are well known for the Poisson problem, for linear elasticity, and flow problems described by Stokes equations. The thesis is devoted to an extension of the multilevel BDDC method to the nonsymmetric problem arising from the discretization of the N-S equations. The weights operators for this form of the BDDC method are discussed including my own one based on an upwind scheme suitable for flow problems. Next, my geometric partitioner is presented which is suitable for these problems, especially for problems with dimensionally complex geometry.

All new approaches presented in this thesis are tested numerically for different problems. Namely, weak scalability of multilevel BDDC method is tested on the flow in the 3D cavity, the test of the influence of the interface between subdomains is tested on the problem of narrowing channel, and all these pieces of knowledge are employed for simulations of oil inside the hydrostatic bearing.

Contents

1 Introduction

1.1 Motivation

The rapid rise of the progress of computers regarding their memory and computational speed is providing a useful tool in many areas of science and technology. Especially in numerical mathematics, which is related to computational mechanics, physics, and mathematics. One of the most common methods in numerical mathematics is the finite element method (FEM), which has been developed and used since the first computers for simulating many physical processes described by partial differential equations (PDEs).

The finite element method is used and developed for many problems of mechanical engineering like structural analysis and flow of fluids. The application of FEM for problems of structural analysis (for example, linear elasticity) was there from the beginning of the development of the FEM method, and therefore, these applications are very advanced. The application of FEM for flow problems came slightly later. This was caused by the complexity of physics and the mathematical models behind these problems. Therefore, another widely spread method was developed, namely, the finite volume method (FVM).

With the growing performance of computers, the possibility of large-scale simulations with high resolution are becoming possible. As the size of the problem grows, the computational time and memory requirements grow as well. Due to the limitation of memory of a single computer as well as growing demands on the computational time, the idea of dividing the computation into several processors, started a new field of interest. This approach ignited the development of algorithms in numerical mathematics suitable for parallel computing. In general, these methods suitable for solving PDEs in parallel are called domain decomposition methods, and their development went hand in hand with the fast development of supercomputers.

Application of the FEM in computational fluid dynamics (CFD) comes with a lot of challenges. For incompressible flow, it is not an easy task to simulate flows in detail or cases with high Reynolds numbers. Computation with fine meshes and therefore with a large number of unknowns are a new way for CFD. For such problems, domain decomposition and parallelization of the algorithms are a must.

The main interest of this thesis is the development of the Balancing Domain Decomposition by Constraints (BDDC) method for incompressible Navier-Stokes equations, therefore, to nonsymmetric systems arising from the discretization of the equations by the finite element method. The idea of domain decomposition (DD) methods is the division of the solution domain into smaller parts called subdomains or substructures and distributing the whole computation to several problems, defined and solved on these subdomains. In the case of finite elements, this decomposition of the whole solution domain means the division into several submeshes. In this way, the methods allow massively parallel implementations.

1.2 State of the art

The topic of domain decomposition for elliptic partial differential equations is investigated by Smith, Bjorstad, and Gropp in [19]. A publication by Toselli and Widlund [21] is devoted to algorithms and theory for DD methods.

In domain decomposition, there are two basic types of division of the original mesh, namely, with overlapping and nonoverlapping subdomains. For overlapping subdomains, the common part of the original mesh that belongs to at least two subdomains is called the overlap which is formed by elements that belong to several subdomains. For nonoverllaping subdomains, the common part is called the interface, and it is formed only by the nodes on the boundaries of subdomains, where two or more subdomains touch each other. Both these methods allow distributing the computations to more processors and computing some parts of the simulation independently in parallel.

The idea of nonoverlapping DD methods is to form an interface problem using the elimination of interior unknowns of each subdomain. The Schur complement matrix and the right-hand side with respect to the interface are created. This matrix and right-hand side form a global interface problem which are much smaller than the original one. It can be so small that it can be solved by a direct method. Once the interface problem is solved, the interior unknowns can be found by solving a Dirichlet problem on each subdomain with a prescribed boundary value on the interface.

For solving Schur complement problems, a Krylov subspace method like the biconjugate gradient stabilized (BiCGstab) method introduced by van der Vorst in [25] are suitable, because only the multiplication of a vector by Schur complement matrix is needed. This product can be computed without an explicit construction of the Schur complement, and only the Dirichlet problem on each subdomain in each iteration is solved.

The BDDC method was introduced by Dohrmann in [6] for the Poisson problem and linear elasticity. The underlying theory for the bound of $O\left(\log^2\left(1 + H/h\right)\right)$ was presented by Mandel and Dohrmann in [14], and Mandel et al. in [15] has shown that the BDDC method is spectrally equivalent to the FETI-DP method [9]. The multilevel extension of the BDDC method was presented first for three levels by Tu in [22], and later for an arbitrary number of levels by Mandel et al. in [16]. The multilevel BDDC method was combined with the adaptive selection of coarse unknowns and implemented into an open-source parallel solver *BDDCML* by Sousedík et al. in [20].

In [13], the BDDC method was first applied to a saddle-point system with symmetric indefinite matrices arising from the discretization of the Stokes problem. This approach uses finite elements with a discontinuous approximation of pressure, which leaves only unknowns for the velocity components at the interface. An approach for the Stokes problem using elements with continuous pressure was investigated by Šístek et al. in [18], and a different approach was later introduced by Li and Tu in [12].

By discretizing and linearizing the Navier-Stokes equations, we get saddle-point systems with nonsymmetric matrices. An application of the BDDC method to nonsymmetric matrices arising from advection-diffusion problems was presented by Tu and Li in [23] and [24], where the method was formulated without an explicit coarse problem. An explicit coarse problem of BDDC was presented by Yano for nonsymmetric problems arising from the Euler equations in [27].

An important building block of the BDDC method is the choice of weights used for averaging a discontinuous solution at the interface between 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 presented by Dohrmann et al. in [7, 5].

1.3 Aims of the work

The aims of my research are two-fold:

- The first aim is to develop a numerical method for computational fluid dynamics employing the extension of the multilevel BDDC method towards the nonsymmetric systems arising from the discretization of the Navier-Stokes equations.
- The second aim is to perform missing detailed 3D simulations of the industrial problem of a flow of oil inside the whole moving hydrostatic bearings.

The problem of the bearing has been provided by Eduard Stach from the Research Center of Manufacturing Technology at the Faculty of Mechanical Engineering of the Czech Technical University in Prague.

The main goal of this thesis is to develop and present the algorithm for the multilevel BDDC method for nonsymmetric systems arising from the discretization of the Navier-Stokes equations by FEM. This novel algorithm is tested on a 3D lid-driven cavity problem. Next, I test the 2-level BDDC method for this problem for different types of weight operators including my own one based on the upwind scheme. Next, I present the strategy of mesh partitioning, including my partitioner preserving 2D interfaces between each subdomain which is shown to be suitable for dimensionally complex geometries, and a mesh builder for building solution domain by individual subdomains for large meshes. Finally, I present the chronology of the simulation for the industrial problem of oil flow inside the hydrostatic bearing. Here, I gradually use all pieces of knowledge from previous numerical experiments to perform simulation for a realistic geometry of the hydrostatic bearing and also validate the calculation with an experiment.

2 The Navier-Stokes equations and the finite element method

I consider a stationary incompressible flow in a bounded three-dimensional domain Ω with Lipschitz boundary governed by the Navier-Stokes equations with zero body forces (see e.g. [8]),

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

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

where

- $\boldsymbol{u} = (u_1, u_2, u_3)^T[\mathrm{ms}^{-1}]$ denotes the unknown velocity vector as a function of space variable $\mathbf{x}$
- p [Pa] is an unknown pressure normalized by (constant) density as a function of $\mathbf{x}$
- ν $[\mathrm{m}^2\mathrm{s}^{-1}]$ is a given kinematic viscosity
- $\boldsymbol{f}$ is a given volume force function
- Ω is a solution domain

In addition, we consider the following boundary conditions,

$$\boldsymbol{u} = \boldsymbol{g} \qquad \text{on } \Gamma_D, \tag{3}$$

$$-\nu(\nabla\boldsymbol{u})\mathbf{n} + p\mathbf{n} = 0 \qquad \text{on } \Gamma_N, \tag{4}$$

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.

This completed system has for low Reynolds numbers a unique solution (see [8]), except for case with prescribed velocity on the whole boundary, i.e. $\Gamma_D = \Gamma$. Then the solution of pressure is unique up to a constant.

2.1 Weak formulation

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

$$\int_\Omega (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\cdot\boldsymbol{v}\ \mathrm{d}\Omega + \nu\int_\Omega \nabla\boldsymbol{u} : \nabla\boldsymbol{v}\ \mathrm{d}\Omega - \int_\Omega p\nabla\cdot\boldsymbol{v}\ \mathrm{d}\Omega = 0 \quad \forall\boldsymbol{v}\in V, \tag{5}$$

$$\int_\Omega q\nabla\cdot\boldsymbol{u}\ \mathrm{d}\Omega = 0 \quad \forall q\in L^2(\Omega), \tag{6}$$

where the function spaces V_g and V are defined as

$$V_g := \left\{\boldsymbol{u}\in[H^1(\Omega)]^3\,|\boldsymbol{u} = \boldsymbol{g} \text{ on } \Gamma_D\right\}, \tag{7}$$

$$V := \left\{\boldsymbol{v}\in[H^1(\Omega)]^3\,|\boldsymbol{v} = \mathbf{0} \text{ on } \Gamma_D\right\}. \tag{8}$$

2.2 Discretization of the weak formulation using FEM

On each element we approximate solutions of the velocity and the pressure and the corresponding test functions by polynomials of a certain degree. For solving the Navier-Stokes equations, it is suitable to choose polynomials of different degree for velocity and pressure. For my computations, I choose the hexahedral Taylor-Hood $Q_2 - Q_1$ finite elements (see e.g. [8]). 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 inter-element boundaries.

The result of the mixed approximation method in the matrix form is

$$\begin{bmatrix} \nu\mathbf{A} + \mathbf{N}(\mathbf{u}^k) & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}^{k+1} \\ \mathbf{p}^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}, \tag{9}$$

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 (9) is assembled as (see [8]). The term $\mathbf{N}(\mathbf{u})$ is linearized using Picard's iteration. This simple method use for linearization of matrix $\mathbf{N}$ solution of the velocity from previous step $\mathbf{u}^k$. This—already linear—nonsymmetric system is solved by means of iterative substructuring using the BDDC method as a preconditioner.

3 Domain decomposition methods

Domain decomposition methods are mathematical algorithms for solving linear and nonlinear systems of algebraic equations which arise from discretization of partial differential equations (PDEs). These methods are suitable for parallel computations and their main idea is to decompose the solved problem into several smaller problems which are easier to solve and subsequently join each of these problems into the global one. This allows us to find the parts of the solution independently on each subdomain, therefore in parallel. It is obvious that the decomposition of the solution domain into completely independent problems is not possible and some communication and exchange of information between subdomains is necessary. The main motivation for usage of these methods is their potential for effective parallelization due to the local utilization of data and the ability to solve PDEs which shows different behaviour on different parts of the solution domain.

In general, we can divide domain decomposition methods into two basic types:

- Overlapping methods – also known as Schwarz methods, where individual subdomains overlap with at least one other subdomain and therefore share some part of the original solution domain (see Figure 1 (left))
- Nonoverlapping methods – also known as substructuring or Schur complement methods where there is no overlap between subdomains and subdomains share only the interface between them (see Figure 1 (right))

For both types of domain decomposition methods, there exist several kinds of algorithms with different behaviour and suitability for usage (see [19] and [21] for more details). In my calculations, I use the Balancing Domain Decomposition by Constraints (BDDC), which is a method with nonoverlapping domain decomposition of the solution domain used as a preconditioner for solving linear system of algebraic equations.

3.1 Iterative substructuring

The main idea of iterative substructuring is to reduce the original system arising from the finite element method to the Schur complement system as in [21]. In general, several levels of nested subdivision can be used in these algorithms. We can see iterative substructuring as a method, in

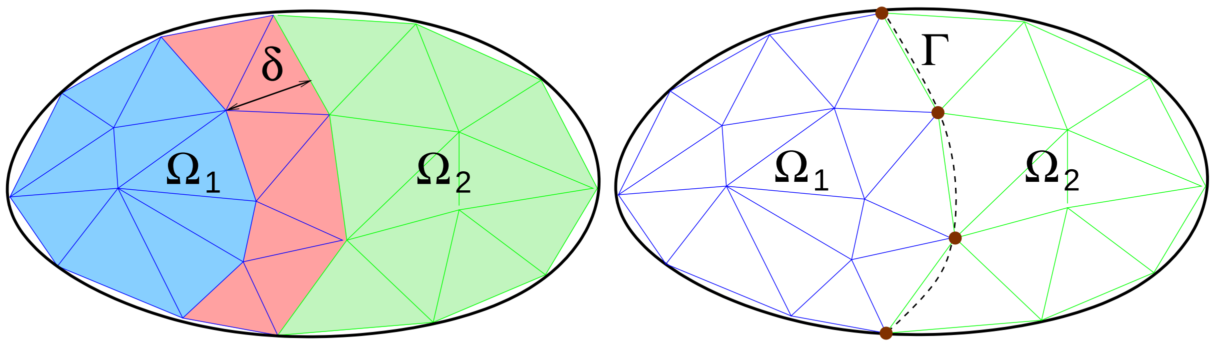

Figure 1: Types of domain decomposition methods: overlapping (left) and nonoverlapping (right)

which we stop this subdividing process at some point and solve the remaining linear system by a preconditioned Krylov subspace method. In parallel computing, one or several of these substructures are assigned to one processor and the interior problems can be solved independently, hence in parallel.

In order to reduce the system to the interface (or Schur complement system) we decompose our solution domain Ω into N nonoverlapping subdomains. 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 Γ.

We reduce the original system (9) to interface in the form

$$S \begin{bmatrix} \mathbf{u_2} \\ \mathbf{p_2} \end{bmatrix} = g, \tag{10}$$

where

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

is the so-called reduced right-hand side and

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

is the Schur complement with respect to the interface. Here subscript $_1$ denotes the part with the interior unknowns and subscript $_2$ denotes the part with the interface unknowns. In practice, the Schur complement matrix S is obtained by subassembling local Schur complements and often, it is not built explicitly. This operation could be very expensive and therefore only an action of the Schur complement on a vector is computed.

4 BDDC algorithms for nonsymmetric systems and its building components

In this chapter, I present the novel approach of using BDDC method for nonsymmetric systems arising from discretization of the Navier-Stokes equations and its implementation details, like interface scaling and solution domain partitioners. First, I describe multilevel extension of the BDDC method for Navier-Stokes equations (see [2]) presented in [4]. I also desribe interface weights for the BDDC preconditioner asociated with matrix of weights W_i in my calculations including our own new type of weights presented in [4]. I also introduced two types of partitioners for solution domain, especially our own one, which was first presented in [1]. Finally, I introduce my own mesh builder used for building solution domain by individual subdomains.

In my computations, problem (10) is solved by the BiCGstab method [25] with one step of two-, three-, and four-level 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 realised by the multilevel BDDC implementation in the *BDDCML* library[1] [20].

4.1 Multilevel BDDC for nonsymmetric systems

The main focus of this section 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. The content of this section is based on [4], we described the multilevel BDDC preconditioner for the nonsymmetric problems arising from the discretization of the Navier-Stokes equations above. In the k-th iteration, the preconditioner is applied to the residual obtained from the BiCGstab method generating an approximate solution to problem (10).

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. From the subdomain matrix on each level, we build and solve the saddle-point system

$$\begin{bmatrix} S_i^\ell & C_i^{\ell T} \\ C_i^\ell & 0 \end{bmatrix} \begin{bmatrix} \Psi_i^\ell \\ \Lambda_i^\ell \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix}, \tag{11}$$

where ℓ is the level of the BDDC method, S_i^ℓ is the Schur complement with respect to the interface of the i-th subdomain (built as in 10), and C_i^ℓ is the matrix defining the coarse dofs, which has as many rows as is 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.

As introduced in [27], a set of adjoint coarse basis functions $\Psi_i^{*\ell}$ is also needed for nonsymmetric problems. These are obtained as the solution to problem (11) with a transposed matrix,

$$\begin{bmatrix} S_i^{\ell T} & C_i^{\ell T} \\ C_i^\ell & 0 \end{bmatrix} \begin{bmatrix} \Psi_i^{*\ell} \\ \Lambda_i^{\ell T} \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix}. \tag{12}$$

After solving (11) and (12), we can get, on each level, the local coarse matrix on each subdomain as

$$A_{Ci}^\ell = (\Psi_i^{*\ell})^T S_i^\ell \Psi_i^\ell = -\Lambda_i^\ell. \tag{13}$$

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

$$A_j^{\ell+1} = \sum_{i=1}^{N_{S_j}} R_{Cij}^{\ell T} A_{Ci}^\ell R_{Cij}^\ell, \tag{14}$$

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 j-th subdomain on level $\ell+1$, R_{Ci}^ℓ is the restriction of the global vector of coarse unknowns to those present at the i-th subdomain on level ℓ and N_{S_j} is the number of subdomains from level ℓ which form the j-th 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 1.

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

Algorithm 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 (11) and (12) to get $\Psi_i^\ell$ and $\Psi_i^{*\ell}$
4:     get local coarse matrices $A_{Ci}^\ell$ from (13)
5:     build subdomain matrices on the next level $A_j^{\ell+1}$ by (14)
6: end for
7: factorize global coarse matrix $A^L$
```

After this setup, the preconditioner can be applied in each iteration. At the beginning, we take the residual vector r^k and extract its local parts on each subdomain as

$$r_i = W_i R_i r^k, \tag{15}$$

where R_i is the operator restricting a global interface vector to the i-th 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 4.2.

Then we get the coarse residual as

$$r_C = \sum_{i=1}^{N} R_{Ci}^T \Psi_i^{*T} r_i. \tag{16}$$

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

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

$$\begin{bmatrix} S_i & C_i^T \\ C_i & 0 \end{bmatrix} \begin{bmatrix} u_i \\ \lambda \end{bmatrix} = \begin{bmatrix} r_i \\ 0 \end{bmatrix} \tag{17}$$

is solved. Here λ are Lagrange multipliers, and S_i and C_i are the same matrices as in (11). After solving this problem on each subdomain, we get the subdomain correction u_i.

Then we have the coarse problem

$$A_C u_C = r_C. \tag{18}$$

In the 2-level method, we solve (18) by a direct method, whereas in the multilevel method, we apply a step of the BDDC method to solve the coarse problem (18) 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

$$r_C = \sum_{i=1}^{N} R_{Ci}^T \Psi_i^{*T} r_i, \tag{19}$$

and solve subdomain problems on the first level (17).

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

$$r_i^\ell = W_i^\ell R_i^\ell r^\ell, \tag{20}$$

and solve the following interior problem on each subdomain

$$A_{i_{11}}^\ell u_{i_1}^\ell = r_{i_1}^\ell, \tag{21}$$

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 pre-correction as

$$\tilde{r}^\ell_{i_2} = r^\ell_{i_2} - A^\ell_{i_{21}} u^\ell_{i_1}. \tag{22}$$

Next, we solve subdomain problems

$$\begin{bmatrix} S^\ell_i & C^{\ell T}_i \\ C^\ell_i & 0 \end{bmatrix} \begin{bmatrix} u^\ell_{i_2} \\ \lambda^\ell \end{bmatrix} = \begin{bmatrix} \tilde{r}^\ell_{i_2} \\ 0 \end{bmatrix} \tag{23}$$

and build the coarse residual

$$r^\ell_C = \sum_{i=1}^{N^\ell} R^{\ell T}_{Ci} \Psi^{*\ell^T}_i \tilde{r}^\ell_{i_2}. \tag{24}$$

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

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 (23)

$$u^\ell_{C_i} = \Psi^\ell_i R^\ell_{C_i} u^\ell_C \quad \text{and} \quad u^\ell_{i_2}. \tag{25}$$

Then we build the BDDC approximation of the global interface solution

$$u^\ell_2 = \sum_{i=1}^{N^\ell_S} R^{\ell^T}_i W^\ell_i (u^\ell_{C_i} + u^\ell_{i_2}). \tag{26}$$

After that, we distribute the interface solution on each subdomain

$$u^\ell_{i_2} = R^\ell_i u^\ell_2, \tag{27}$$

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

$$A^\ell_{i_1 1} \tilde{u}^\ell_{i_1} = -A^\ell_{i_1 2} u^\ell_{i_2}. \tag{28}$$

Finally, we bring in the interior correction from (21) and build the approximate coarse solution as

$$u^{\ell-1}_C = u^\ell = \begin{bmatrix} \sum_{i=1}^{N^\ell_S} R^{\ell T}_{1i} (\tilde{u}^\ell_{i_1} + u^\ell_{i_1}) \\ u^\ell_2 \end{bmatrix}, \tag{29}$$

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

After this, we have the approximate coarse solution on the second level on each subdomain as

$$u_{Ci} = \Psi_i R_{Ci} u_C. \tag{30}$$

and get the solution of problem

$$u^k = \sum_{i=1}^{N} R^T_i W_i (u_i + u_{Ci}). \tag{31}$$

The process is summarized in Algorithm 2.

Algorithm 2 Application of the BDDC preconditioner with L levels

```
1:  distribute residual to each subdomain on the first level $r_i$ (15)
2:  build the coarse residual on the first level $r_C$ (16)
3:  solve subdomain problems (17) 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 $r_i^\ell$ (20)
7:      solve (21)
8:      get pre-corrected residual $\tilde{r}_{i_2}^\ell$ (22)
9:      solve subdomain problems (23)
10:     build the coarse residual $r_C^\ell$ (24)
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 $u_{C_i}^\ell$ (25)
16:     build global interface solution $u_2^\ell$ (26)
17:     distribute interface solution to each subdomain $u_{i_2}^\ell$ (27)
18:     get interior correction on each subdomain by solving (28)
19:     build approximate coarse solution $u_C^{\ell-1}$ (29)
20: end for
21: distribute the coarse solution on the first level to each subdomain $u_{Ci}$ (30)
22: get preconditioned residual $u^k$ (31)
```

4.2 Interface scaling

Let us now look at the used weights in my computations. Several types of interface weights have been developed for the BDDC preconditioner, with each of them having its advantages and disadvantages for certain kinds of problems.

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 [21],

$$\sum_{i=1}^{N} R_i^T W_i R_i = I, \tag{32}$$

where I is the identity matrix.

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

$$W_i = \begin{pmatrix} w_i^1 & & \\ & w_i^2 & \\ & & \ddots \end{pmatrix}, \tag{33}$$

where w_i^k denotes the weight for the k-th (with respect to the subdomain interface) dof on the i-th 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 neighbouring 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 neighbours,

$$w_i^k = \frac{\widetilde{w}_i^k}{\sum_{j=1}^{N_S} \widetilde{w}_j^k}, \tag{34}$$

where N_S is the number of subdomains sharing the dof.

One of the 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 Fig. 2),

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

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 (34).

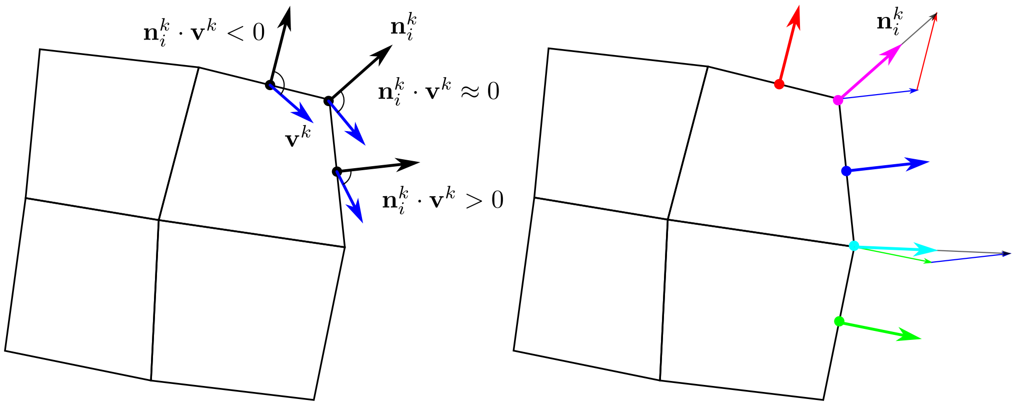
$\mathbf{n}_i^k \cdot \mathbf{v}^k < 0$
$\mathbf{n}_i^k$
$\mathbf{n}_i^k \cdot \mathbf{v}^k \approx 0$
$\mathbf{v}^k$
$\mathbf{n}_i^k \cdot \mathbf{v}^k > 0$
$\mathbf{n}_i^k$

Figure 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 Fig. 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.

4.3 Mesh partitioning

Let us now look at the employed partitioners. I use two approaches to partitioning the computational domain and mesh into subdomains.

A standard approach is based on converting the computational mesh into a graph. Graph partitioning provides an automated way for dividing 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 the graph, and the resulting interface can be very irregular. This is a known issue studied mathematically for elliptic problems e.g. by [11].

Our partitioner works similarly as the RCB partitioner to preserve 2D interfaces between subdomains. For a given problem, we first define cuboidal blocks of the solution domain and divide these blocks by recursive bisection along given axes into balanced subdomains. This comes hand in hand with creating a suitable structural mesh of the solution domain using *GMSH* software [10], and defined cuboidal volumes which also preserve 2D interfaces between each other. It seems reasonable to prescribe a suitable number of subdomains for each cuboidal block and obtain the total number of subdomains simply as a sum over these blocks for preventing load imbalance.

In the rest of the paper, we refer to these two strategies as the *graph* and the *geometric* partitioner. An example of the interfaces between two subdomains provided by these partitioners are in Fig. 3.

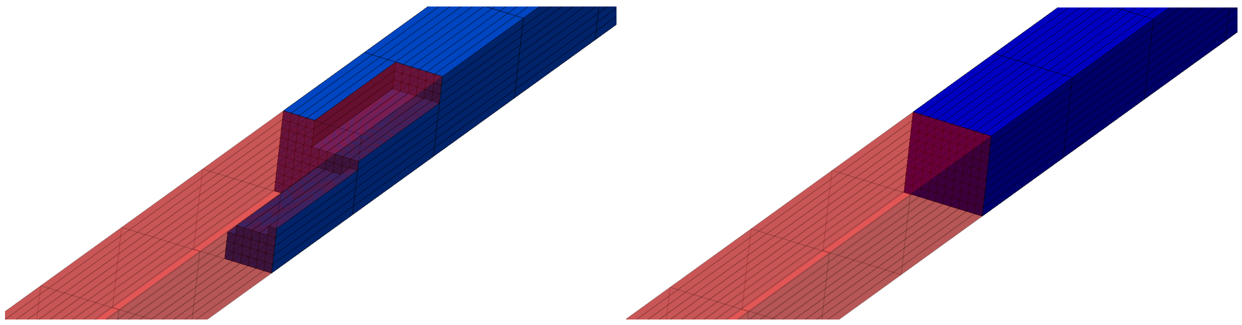
Figure 3: Interfaces between subdomains provided by a *graph* partitioner (left) and by a *geometric* partitioner (right).

4.4 Mesh builder

In this section, I introduce my software used for creating subdomains for a large mesh employed in the computation of the cavity problem described in Section 5.2. In these simulations, I needed to solve the problem of dividing a large cubic mesh into smaller cubes as subdomains. Due to the size of these meshes, I was not able to use neither the *graph* nor the *geometric* partitioner. Therefore, I create a program that produces files with individual subdomains and follows the global numbering of the whole solution domain created by the *GMSH* software.

This *MeshBuilder* software produces a set of cubic subdomains corresponding with unit cube under two parameters. The first is the number of subdomains per cube edge, and the second is the number of hexahedral Taylor-Hood Q2-Q1 elements per subdomain edge. The main part of this software is to make a correct mapping between the local and global degrees of freedom. Both, the global and the local numbering of DOFs are done subsequentially in the x, then in y, and finally in the z coordinate.

5 Numerical results

In this section, I apply the variants of the BDDC method to steady incompressible flow governed by the Navier-Stokes equations. I sumarize all my results concerning investigation of possibilities of the multilevel BDDC method as well as its applicability to the industrial problem of flow inside the hydrostatic bearings. All the computations are performed by a parallel finite element package written in C++ and described in [17], and using the *BDDCML* library [20] for solving the arising system of linear equations. For linearization, we use Picard's iteration with the precision measured as $\epsilon_N = \left\|\mathbf{u}^k - \mathbf{u}^{k-1}\right\|_2$. The BiCGstab method preconditioned by the BDDC preconditioner is used for the linearized system with precision measured as $\epsilon_L = \left\|r^k\right\|_2 / \left\|g\right\|_2$. According to our previous experience from [17], the convergence of the BiCGstab method is comparable to that of GMRES. Simulations were performed on the *Salomon* supercomputer at the IT4Innovations National Supercomputing Center using the cores of Intel Xeon E5-2680v3 12C 2.5GHz processors.

5.1 Narrowing channel

In this computations I aim at the influence of interface irregularities on the BDDC solver for Navier-Stokes equations. In particular, I investigate the effect of the aspect ratio of the finite elements at

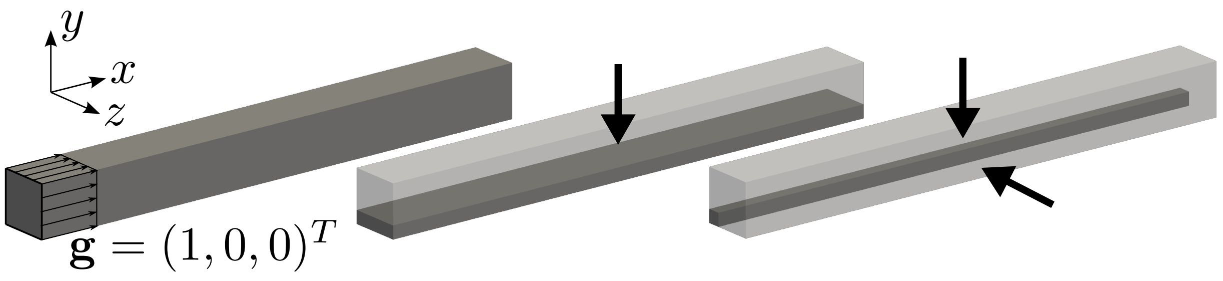

Figure 4: The narrowing channel 3-D benchmark; original channel (left), narrowing along the y-axis (centre), and narrowing along both y and z-axes (right).

partitioner		graph						
Æ		1	2	4	10	20	40	100
Picard's its.		4	4	4	5	8	19	*28*
BiCGstab its.	min	17.5	19.5	27.5	36	51	80	197
	max	18.5	20.5	28	41.5	53	92.5	*1000*
	mean	18.3	19.8	27.9	39.5	51.8	87.7	590

Table 1: Numbers of iterations for *graph* partitioner for the 3-D channel narrowed along both y and z-coordinates.

the interface on convergence. In order to study this phenomenon, a benchmark problem suitable for such a study is proposed and the partitioning strategies described in Section 4.4 are compared. Picard's iteration is terminated when $\epsilon_L \leq 10^{-5}$ or after performing 100 iterations. The BiCGstab method is stopped if $\epsilon_N \leq 10^{-6}$, with the limit of 1000 iterations.

As a measure of convergence, I monitor the number of BiCGstab iterations needed in one Picard's 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 where the dimension of the channels along one or two coordinates is gradually decreased (Fig. 4). The *aspect ratio of elements* $Æ = h_{\max}/h_{\min}$ is defined as the ratio of the longest edge of the element $h_{\max}$ to its shortest counterpart $h_{\min}$.

For the case, where we shrink both y and z dimensions of the cross-section (see Fig. 4). The graph partitioner produces rough interface, while the geometric partitioner leads strait cuts between subdomains. Resulting numbers of iterations are presented in Tables 1 and 2. Numbers in italic are runs that did not converge due to reaching the maximal number of iterations or time restrictions.

From Tables 1 and 2 we can conclude that Æ of faces at the interface has a remarkable influence on the number of BiCGstab iterations in each Picard's 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 good convergence independent of Æ.

5.2 Lid-driven cavity

In this section we investigate behaviour of the BDDCML solver in application to benchmartk problem of 3-D lid-driven cavity suggested in [26]. The computational domain is a unit cube with Dirichlet boundary conditions. A twisted unit tangential velocity vector is considered on the top wall, $\boldsymbol{u}_{\text{top}} = (1/\sqrt{3}, \sqrt{2}/\sqrt{3}, 0)$. Zero velocity is considered on the remaining 5 walls. The computational mesh is uniform with hexahedral elements, and it is divided into cubic subdomains

partitioner		geometric						
AR		1	2	4	10	20	40	100
Picard's its.		4	4	4	5	5	5	4
BiCGstab its.	min	5.5	5.5	6	5	4.5	4.5	4.5
	max	5.5	6	6	5.5	5	5	4.5
	mean	5.5	5.9	6	5.1	4.9	4.6	4.5

Table 2: Numbers of iterations for *geometric* partitioner for the 3-D channel narrowed along both y and z-coordinates.

with a rising number of them per domain edge (see Fig. 5) using my *Meshbuilder* software described in Section 4.4. 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 Picard's iterations after reaching the precision $\epsilon_N \leq 10^{-5}$ or after 100 iterations, while the inner linear iterations are terminated when $\epsilon_L \leq 10^{-6}$ or after reaching the maximum number of 1000 iterations.

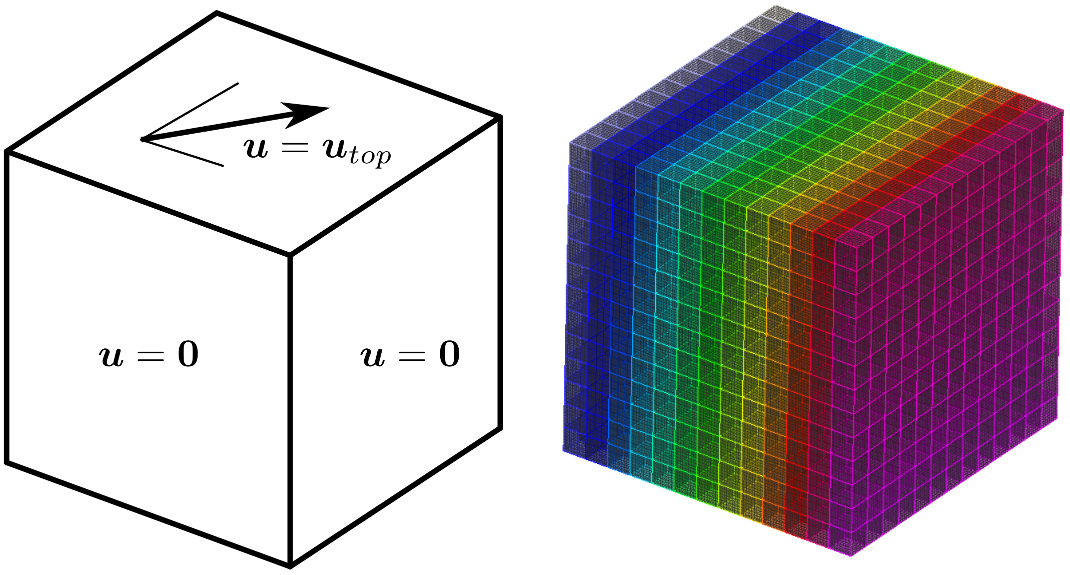
$\boldsymbol{u} = \boldsymbol{u}_{top}$
$\boldsymbol{u} = \boldsymbol{0}$
$\boldsymbol{u} = \boldsymbol{0}$

Figure 5: Boundary conditions (left) and example of mesh division with 13 subdomains per edge (right) for the lid-driven cavity.

I present three performed tests of behaviour of BDDCML method. Namely, weak scalability compared for 2-, 3-, and 4-level method, comparision of four used weights type described in Section 4.2, including our own new upwind weight type.

5.2.1 Weak scalability

In this section, I compare the weak scalability of the multilevel BDDC method in application to 3D lid-driven cavity. Results for the 2-, 3-, and 4-level BDDC method are presented.

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. The comparison of the behaviour of the 2- and 3-level methods for these parameters with a rising number of processors is presented in Fig. 6.

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

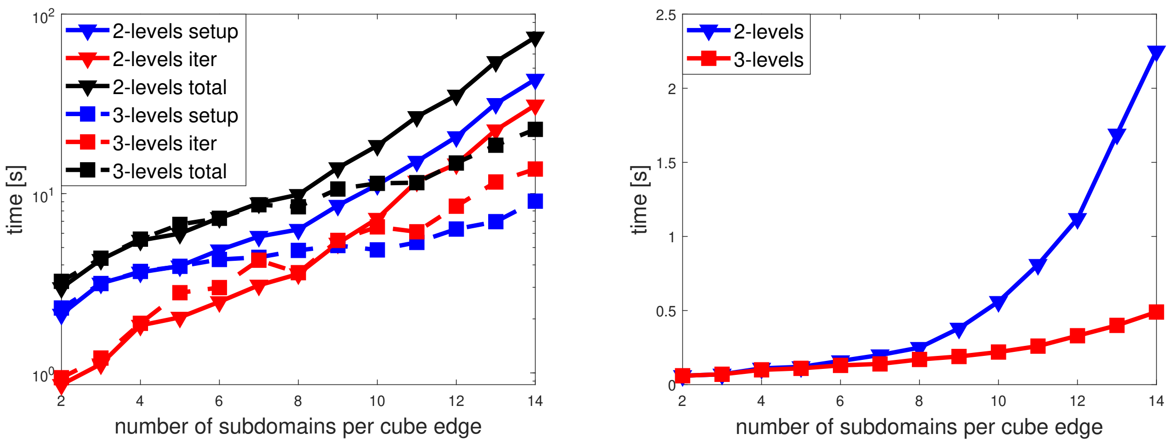

Figure 6: $Re = 100$. Mean time for setup, mean time for the BiCGstab iterations and mean total time (left), mean time for one iteration (right).

communication related to propagating the higher-level solutions and residuals what worsens the weak scalability also for the 3-level method.

The comparison of the behaviour of the 3- and the 4-level methods for these parameters with a rising number of processors is presented in Fig. 7.

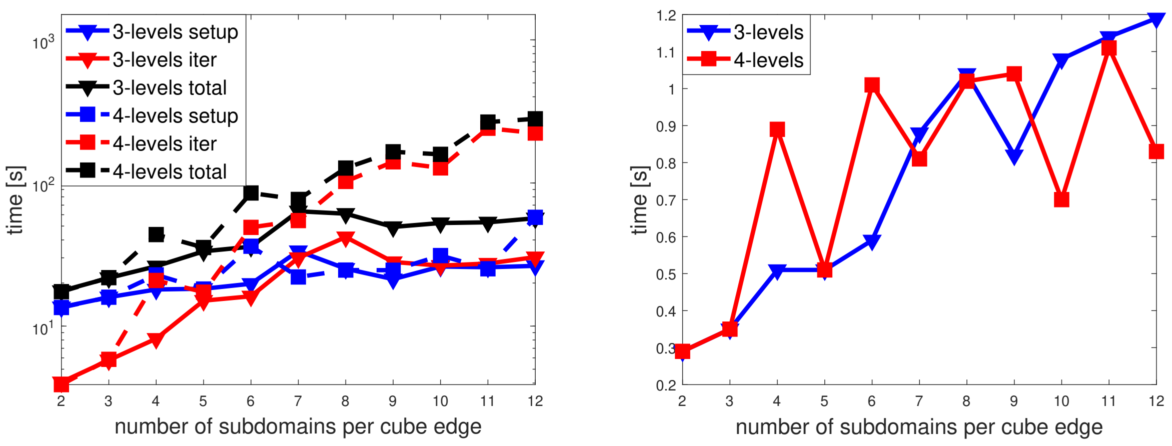

Figure 7: $Re = 1$. Mean time for setup, mean time for the BiCGstab iterations and mean total time (left), mean time for one iteration (right).

In Figure 7, there are once again some wild drops and ups 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.

5.2.2 Interface scalings

In this section, we compare the behaviour of the 2-level BDDC method for different types of interface weights, 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 8 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, the mean time for the Krylov subspace method, and the mean time for one iteration. These values are presented in Tables 3 and 4.

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	23	14.5	14.3	4.05	1.69 (0.12)	5.74

Table 3: $Re = 100$. 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	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

Table 4: $Re = 200$. 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.

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

From Tables 3 and 4, 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, 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.

5.3 Hydrostatic bearings

In this section we finally combine the previous experience and present the results for driving application of our research, namely simulations of oil flow in hydrostatic bearings. These are parts of production machines that keep 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.

Hydrostatic bearings have been studied for a long time at the Research Center of Manufacturing Technology at the Faculty of Mechanical Engineering of the Czech Technical University in Prague. These parts are typically designed using analytic solution of flow inside the throttling gap. However, a detailed picture of the 3D flow in the whole bearing was missing.

Numerical simulation of this problem comes with several challenges, like a large pressure gradient realized in the throttling gap, small thickness of the throttling gap, and moving of the bearing. During my research, I have gradually addressed most of these issues, at the end being able to simulate real-scale geometry problems of sliding bearings.

Utilizing the findings from the previous experiments, I was able to perform simulations on a real geometry of the bearing from a production machine. Here I recall our results from [3]. Our calculations aim at an industrial problem of oil flow inside hydrostatic bearings. In this section, I present results for a bearing sliding in a straight direction, and this set-up corresponds to a running production machine.

The computational mesh was created and divided into specified structured volumes using the *GMSH* software. These volumes preserve flat interfaces between each other to make it possible to decompose the solution domain into 21 global subdomains using our 'geometric' partitioner prefers straight cuts between subdomains described in [1]. The mesh consists of 124 828 elements which correspond with 3 269 679 unknowns and 186 834 interface unknowns. The decomposed mesh is presented in Figure 8. In Figure 9 there are streamtraces in different parts of the bearing coloured by the velocity magnitude.

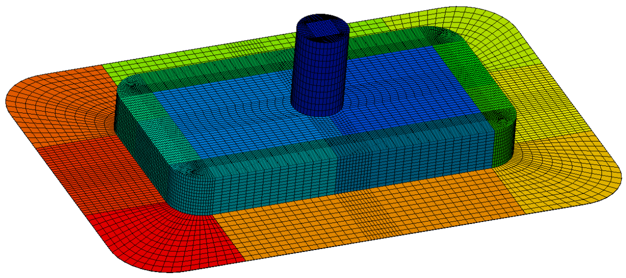

Figure 8: Computational mesh decomposed into 21 subdomains.

For presented case, we needed 3 Picard's iterations to reach precision $\epsilon_N \leq 10^{-4}$, while on average 748 BiCGstab iterations were needed for the linearised system to achieve the precision of the relative residual $\epsilon_L \leq 10^{-5}$.

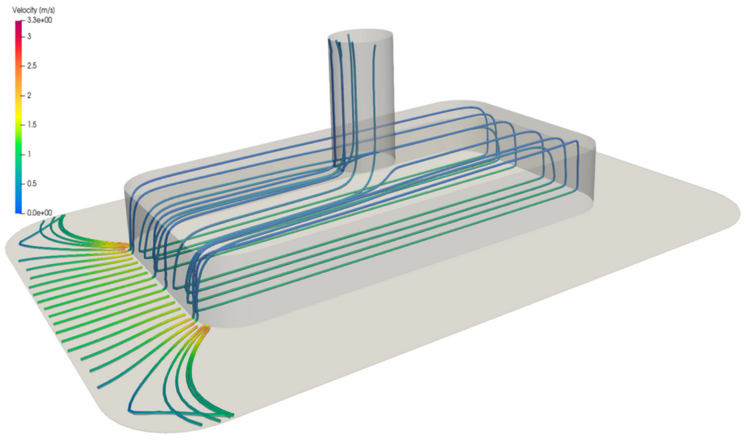

Figure 9: Streamtraces coloured by the magnitude of velocity in the sliding hydrostatic bearing along the short edge in negative x direction.

We can see that a large vortex rolling inside of the hydrostatic cell is formed, and the oil is pulled by the motion of the bottom wall towards one side of the throttling gap. This results in the fact that the maximal magnitude of the velocity of the oil is at the entrance to the throttling gap in the side which corresponds with the movement of the hydrostatic bearing. The latter effect is important for setting the operational regime of the machine, especially the maximum velocity of the sliding such that the oil flow towards the front of the bearing is still maintained.

6 Conclusion

In today's computation, a high resolution large-scale simulations are necessary for many engineering applications. This is allowed by a suitable mathematical methods, namely, the domain decomposition methods. Using this approach, it is possible to perform many kinds of parallel simulations.

Therefore, fast and effective parallel solvers for computational fluid dynamics play an important role in nowadays research.

In this thesis, the mathematical model of flow of a fluid described by steady incompressible Navier-Stokes equations is considered. These equations are discretized by the finite element method and the arising nonlinear systems are linearized by Picard's iteration. The system of linear equations is solved using a nonoverlapping domain decomposition method by means of iterative substructuring. The arising nonsymmetric interface problem is solved by the BiCGstab method preconditioned by one step of BDDC and its multilevel variant. In addition, various weight types and used partitioners are discussed in this chapter. Computations are performed for several problems. I have compared the weak scalability of 2-, 3-, and 4-level BDDC methods for 3D lid-driven cavity, compared two types of partitioners of the solution domain, compared four different weights on the interface including my own upwind based, and presented the chronology of simulating the flow of oil inside the hydrostatic bearing ending with real-scale geometry simulations. Now I recall the main original results and achievements.

By combining the approaches from [18] and [27], I applied the 2-level BDDC method to the Navier-Stokes equations for the lid-driven cavity benchmark problem in [2]. Next, I have presented a formulation of the multilevel BDDC preconditioner for nonsymmetric problems and its application to linear systems obtained by Picard's linearization of the Navier-Stokes equations in a journal paper [4]. My computations employed the *BDDCML* solver and a parallel finite element package written in C++ described in [17], which I have extended towards the flow problems related to my thesis.

The tests of the influence of the interface between subdomains were performed for two partitions in [1]. The first was created by a partitioner based on the graph partitioning implemented in the *METIS* library. The second one was my own partitioner based on geometry and division of the domain with straight cuts promoting just a 2D interface between individual subdomains. From the comparison of the number of linear iterations with a growing aspect ratio of the finite elements, one can see that using the new geometric partitioner with straight cuts significantly decreases the number of linear iterations with increasing the aspect ratio of the elements.

For a benchmark of 3-D lid-driven cavity problem, I have explored the behaviour of the 2-, 3-, and 4-level BDDC method. I have focused mainly on the number of linear iterations and the 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. For this problem, I also introduced my software for building the computational mesh by individual subdomains which was necessary, especially for large meshes.

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. My results have suggested that with a growing Reynolds number, the importance of the scaling type increases, and the upwind weights provide promising results as presented in [4].

In [4], I have performed experiments investigating the behaviour 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 problems, the numbers of BiCGstab iterations seem to confirm the logarithmic behaviour has proven for symmetric positive definite problems.

Finally, I have applied the BDDC methodology to an industrial problem in engineering, namely, the flow of oil in hydrostatic bearings. By solving a number of issues, I have managed to perform challenging computations on a real geometry of the bearing by means of the 2-level BDDC method presented in [3].

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 bearing problem. This could also help to explain the rapidly worsening behaviour 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

Author's publications

[1] M. Hanek, J. Šístek, and P. Burda. The effect of irregular interfaces on the BDDC method for the Navier-Stokes equations. In Ch.-O. Lee, X.-Ch. Cai, D. E. Keyes, H. H. Kim, A. Klawonn, E.-J. Park, and O. B. Widlund, editors, *Domain Decomposition Methods in Science and Engineering XXIII*, pages 171–178. Springer International Publishing AG, 2017.

[2] 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, *Programs and algorithms of numerical mathematics 17*, pages 77–85. Institute of Mathematics AS CR, 2015.

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

[4] M. Hanek, J. Šístek, and P. Burda. Multilevel BDDC for incompressible Navier-Stokes equations. *SIAM J. Sci. Comput.*, 42(6):C359–C383, 2020.

Bibliography

[5] C. Dohrmann and B. Widlund, O. A BDDC algorithm with deluxe scaling for three-dimensional H(curl) problems. *Comm. Pure Appl. Math.*, 69(4):745–770, 2016.

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

[7] R. Dohrmann, C. and B. Widlund, O. Some recent tools and a BDDC algorithm for 3D problems in H(curl). In Randolph Bank, Michael Holst, Olof Widlund, and Jinchao Xu, editors, *Domain Decomposition Methods in Science and Engineering XX*, volume 91 of *Lecture Notes in Computational Science and Engineering*, pages 15–25. Springer, 2013.

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

[9] Ch. Farhat, M. Lesoinne, and K. Pierson. A scalable dual-primal domain decomposition method. *Numer. Linear Algebra Appl.*, 7:687–714, 2000.

[10] 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.*, 79:1309–1331, 2009.

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

[12] J. Li and X. Tu. A nonoverlapping domain decomposition method for incompressible Stokes equations with continuous pressures. *SIAM J. Numer. Anal.*, 51(2):1235–1253, 2013.

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

[14] J. Mandel and R. Dohrmann, C. Convergence of a balancing domain decomposition by constraints and energy minimization. *Numer. Linear Algebra Appl.*, 10(7):639–659, 2003.

[15] J. Mandel, R. Dohrmann, C., and R. Tezaur. An algebraic theory for primal and dual substructuring methods by constraints. *Appl. Numer. Math.*, 54(2):167–193, 2005.

[16] J. Mandel, B. Sousedík, and R. Dohrmann, C. Multispace and multilevel BDDC. *Computing*, 83(2-3):55–85, 2008.

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

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

[19] F. Smith, B. *Domain decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Numerical Mathematics and Scientific Computation. Cambridge University Press, Cambridge, 1996.

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

[21] A. Toselli and B. Widlund, O. *Domain Decomposition Methods—Algorithms and Theory*, volume 34 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2005.

[22] X. Tu. Three-level BDDC in three dimensions. *SIAM J. Sci. Comput.*, 29(4):1759–1780, 2007.

[23] X. Tu and J. Li. A balancing domain decomposition method by constraints for advection-diffusion problems. *Commun. Appl. Math. Comput. Sci*, 3(1):25–60, 2008.

[24] X. Tu and J. Li. BDDC for nonsymmetric positive definite and symmetric indefinite problems. In M. Bercovie, M. Gander, R. Kornhuber, and O. Widlund, editors, *Domain Decomposition Methods in Science and Engineering XVIII*, pages 75–86. Springer International Publishing AG, 2009.

[25] 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. Statist. Comput.*, 13(2):631–644, 1992.

[26] J. Wathen, A., D. Loghin, A. Kay, D., C. Elman, H., and J. Silvester, D. A new preconditioner for the Oseen equations. In F. Brezzi, A. Buffa, S. Corsaro, and A. Murli, editors, *Numerical mathematics and advanced applications*, pages 979–988, Milano, 2003. Springer-Verlag Italia. Proceedings of ENUMATH 2001, Ischia, Italy.

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