MATHEMATICS OF COMPUTATION
Volume 78, Number 266, April 2009, Pages 789–814
S 0025-5718(08)02172-8
Article electronically published on November 24, 2008

# DERIVING A NEW DOMAIN DECOMPOSITION METHOD FOR THE STOKES EQUATIONS USING THE SMITH FACTORIZATION

VICTORITA DOLEAN, FRÉDÉRIC NATAF, AND GERD RAPIN

Abstract. In this paper the Smith factorization is used systematically to derive a new domain decomposition method for the Stokes problem. In two dimensions the key idea is the transformation of the Stokes problem into a scalar bi-harmonic problem. We show, how a proposed domain decomposition method for the bi-harmonic problem leads to a domain decomposition method for the Stokes equations which inherits the convergence behavior of the scalar problem. Thus, it is sufficient to study the convergence of the scalar algorithm. The same procedure can also be applied to the three-dimensional Stokes problem.

As transmission conditions for the resulting domain decomposition method of the Stokes problem we obtain natural boundary conditions. Therefore it can be implemented easily.

A Fourier analysis and some numerical experiments show very fast convergence of the proposed algorithm. Our algorithm shows a more robust behavior than Neumann-Neumann or FETI type methods.

## 1. Introduction

The last decade has shown, that Neumann-Neumann type algorithms, FETI, and BDDC methods are very efficient domain decomposition methods. Most of the early theoretical and numerical work has been carried out for scalar symmetric positive definite second order problems; see for example [6], [13]–[15], [23]. Then, the method was extended to other problems, like the advection-diffusion equations [1, 7], plate and shell problems [27] or the Stokes equations [26, 22].

In the literature one can also find other preconditioners for the Schur complement of the Stokes equations (cf. [2, 26]). Moreover, there exist some Schwarz-type algorithms for non-overlapping decompositions (cf. [16, 19, 20, 24]). A more complete list of domain decomposition methods for the Stokes equations can be found in [22, 28]. Also FETI [11] and BDDC methods [12] are applied to the Stokes problem with success.

Our work is motivated by the fact that in some sense the domain decomposition methods for Stokes are less optimal than the domain decomposition methods for scalar problems. Indeed, in the case of two subdomains consisting of the two half-planes it is well known that the Neumann-Neumann preconditioner is an exact preconditioner for the Schur complement equation for scalar equations like the Laplace problem (cf. [23]). A preconditioner is called *exact*, if the preconditioned

Received by the editor October 17, 2006 and, in revised form, October 29, 2007.
2000 *Mathematics Subject Classification.* 65-xx .

operator simplifies to the identity. Unfortunately, this does not hold in the vector case. It is shown in [18] that the standard Neumann-Neumann preconditioner for the Stokes equations does not possess this property.

Our aim in this paper is the construction of a method, which preserves this property. Thus, one can expect a very fast convergence for such an algorithm; and indeed, the numerical results clearly support our approach. This paper explains the ideas of [4] in more detail. For an application to the compressible Euler equations see [3].

Let us give a short outline of the paper. In Section 2 we introduce the Stokes equations. Concentrating on the two-dimensional case, these equations are transformed into a bi-harmonic operator with the help of the Smith factorization. Then, in Section 3 we first introduce an iterative domain decomposition method for the bi-harmonic equations and we show how it can be used for the Stokes equations. Moreover, in Section 4 we discuss briefly, how this approach can be extended to the linearized Navier-Stokes equations (Oseen equations). In the case of two subdomains we were able to derive an algorithm which converges independently of the Reynolds number in two iterations. Most likely, ongoing research will show that we will retrieve this behavior for more general decompositions. Then, in Section 5 the algorithm is extended to the three-dimensional Stokes problem. A finite volume discretization is discussed in Section 6. Section 7 is dedicated to numerical results for the two-dimensional Stokes problem. Finally, we give some concluding remarks.

## 2. Equivalence between the Stokes equations and the bi-harmonic problem

In this section we show the equivalence between the Stokes system and a fourth order scalar problem (the bi-harmonic problem) by means of the Smith factorization. This is motivated by the fact that scalar problems are easier to manipulate and the construction of new algorithms is more intuitive. Additionally, the existing theory of scalar problems can be used.

2.1. **Stokes equations.** We consider the stationary Stokes problem in a bounded domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$. The Stokes equations are given by a velocity $\vec{u}$ and a pressure $p$ satisfying

$$\begin{aligned} -\nu\Delta\vec{u} + \nabla p + c\vec{u} &= \vec{f} \quad \text{in } \Omega, \\ \nabla \cdot \vec{u} &= 0 \quad \text{in } \Omega, \end{aligned}$$

and some boundary conditions on $\partial\Omega$. The Stokes problem is a simple model for incompressible flows. The right hand side of $\vec{f} = (f_1, \ldots, f_d)^T \in [L^2(\Omega)]^d$ is a source term, $\nu$ is the viscosity and $c \geq 0$ is a constant reaction coefficient. Very often $c$ stems from an implicit time discretization and then $c$ is given by the inverse of the time step size.

In the following we denote the $d$-dimensional Stokes operator by $\mathcal{S}_d(\vec{v}, q) := (-\nu\Delta\vec{v} + c\vec{v} + \nabla q, \nabla \cdot \vec{v})$.

2.2. **Smith factorization.** Now we show that the Stokes problem can be transformed into a scalar fourth order problem using the Smith factorization. We recall the Smith factorization of a matrix with polynomial entries ([29], Theorem 1.4):

**Theorem 2.1.** *Let $n$ be an positive integer and $A$ an invertible $n \times n$ matrix with polynomial entries with respect to the variable $\lambda$: $A = (a_{ij}(\lambda))_{1 \leq i,j \leq n}$. Then,*

*there exist matrices $E$, $D$ and $F$ with polynomial entries satisfying the following properties:*

- *$det(E)$ and $det(F)$ are constants,*
- *$D$ is a diagonal matrix uniquely determined up to a multiplicative constant,*
- *$A = EDF$.*

*Here $E$ and $F$ are matrices, which operate on the rows, respectively, columns. The entries of the diagonal matrix $D = (d_{ij}(\lambda))$ are given by $d_{ii} = \phi_i/\phi_{i-1}$, where $\phi_i$ is the greatest common divisor of the determinants of all $i \times i$ submatrices of $A$ and $\phi_0 = 1$.*

The Smith factorization is a classical tool in computer algebra and in control of ordinary differential equations. Since its use in scientific computing is rather new, we give here a few comments:

- Smith was an English mathematician of the end of the nineteenth century. He worked in number theory and considered the problem of factorizing matrices with integer entries. We give here the polynomial version of his theorem in the special case where the matrix $A$ is square and invertible but the result is more general and applies as well when the matrix $A$ is rectangular.
- One interesting fact of the theorem is the following. By Cramer's formula, the inverse of $A$ is in general a matrix with rational entries. By the Smith factorization, we have $A^{-1} = F^{-1}D^{-1}E^{-1}$. Since $det(E)$ and $det(F)$ are constants, the inverse of $E$ and $F$ are still matrices with polynomial entries in $\lambda$. The rational part of the inverse of $A$ is thus in $D^{-1}$ which is an intrinsic diagonal matrix.
- The proof of the theorem is constructive and gives an algorithm for computing matrices $E$, $D$ and $F$. As stated in the theorem, matrix $D$ is intrinsic but matrices $E$ and $F$ are not unique.
- In the sequel, we write the Stokes equations as a matrix with partial differential operator entries applied to the unknown velocity and pressure fields. The direction normal to the interface of the subdomains is particularized and denoted by $\partial_x$. Each partial differential operator is then considered as a polynomial in the "variable $\partial_x$" (e.g. $\lambda$ is related to $\partial_x$ and $\lambda^2$ to $\partial_{xx}$). It is then possible to apply the Smith factorization; see below.

**Application to the two-dimensional Stokes problem.** The Smith factorization is applied to the following model problem in the whole plane $\mathbb{R}^2$,

$$\mathcal{S}_d(\vec{u}, p) = \vec{g} \quad \text{in } \mathbb{R}^2, \tag{2.1}$$

$$|\vec{u}(\vec{x})| \to 0 \quad \text{for } |\vec{x}| \to \infty \tag{2.2}$$

with the right hand side $\vec{g} = (f_1, f_2, 0)^T$. Moreover, it is assumed, that the coefficients $c, \nu$ are constants.

We start with the two-dimensional case. The spatial coefficients are denoted by $x$ and $y$. In order to apply the factorization to the Stokes system, we first take formally the Fourier transform of (2.1) with respect to $y$. The dual variable is denoted by $k$. The Fourier transform of a function $f$ is written as $\hat{f}$ or $\mathcal{F}_y f$. Thus,

equation (2.1) yields $\hat{\mathcal{S}}_2(\hat{\vec{u}},\hat{p}) = \hat{\vec{g}}$ with $\hat{\vec{u}} = (\hat{u},\hat{v})$ and

$$\hat{\mathcal{S}}_2(\hat{\vec{u}},\hat{p}) = \begin{pmatrix} -\nu(\partial_{xx}-k^2)+c & 0 & \partial_x \\ 0 & -\nu(\partial_{xx}-k^2)+c & ik \\ \partial_x & ik & 0 \end{pmatrix} \begin{pmatrix} \hat{u} \\ \hat{v} \\ \hat{p} \end{pmatrix}. \tag{2.3}$$

Considering $\hat{\mathcal{S}}_2$ as a matrix with polynomial entries with respect to $\partial_x$, we perform for $k \neq 0$ the Smith factorization. We obtain

$$\hat{\mathcal{S}}_2 = \hat{E}_2 \hat{D}_2 \hat{F}_2 \tag{2.4}$$

with

$$\hat{D}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & (\partial_{xx}-k^2)\hat{\mathcal{L}}_2 \end{pmatrix}, \qquad \hat{F}_2 = \begin{pmatrix} \nu k^2 + c & \nu ik\partial_x & \partial_x \\ 0 & \hat{\mathcal{L}}_2 & ik \\ 0 & 1 & 0 \end{pmatrix}$$

and

$$\hat{E}_2 = \hat{T}_2^{-1} \begin{pmatrix} ik\hat{\mathcal{L}}_2 & \nu\partial_{xxx} & -\nu\partial_x \\ 0 & \hat{T}_2 & 0 \\ ik\partial_x & -\partial_{xx} & 1 \end{pmatrix}$$

where $T_2$ is a differential operator in the $y$-direction whose symbol is $ik(\nu k^2 + c)$. Moreover, $\hat{\mathcal{L}}_2 := \nu(-\partial_{xx} + k^2) + c$ is the Fourier transform of $\mathcal{L}_2 := -\nu\Delta + c$.

*Remark* 2.2. Using this factorization, problem (2.1) can be written as

$$\hat{D}_2\hat{\vec{w}} = \hat{E}_2^{-1}\hat{\vec{g}}, \qquad \hat{\vec{w}} := (\hat{w}_1,\hat{w}_2,\hat{w}_3)^T := \hat{F}_2(\hat{\vec{u}},\hat{p})^T. \tag{2.5}$$

From (2.5) we get $\hat{w}_1 = (\hat{E}_2^{-1}\hat{\vec{g}})_1$ and $\hat{w}_2 = (\hat{E}_2^{-1}\hat{\vec{g}})_2$. Noticing that

$$\hat{w}_3 = \left(\hat{F}_2(\hat{\vec{u}},\hat{p})^T\right)_3 = \hat{v}$$

the previous equation yields, after applying an inverse Fourier transform,

$$\Delta(-\nu\Delta + c)v = \mathcal{F}_y^{-1}\left((\hat{E}_2^{-1}\hat{\vec{g}})_3\right).$$

Since the matrices $\hat{E}_2$ and $\hat{F}_2$ have a determinant which is a non-zero number (i.e. a polynomial of degree zero), the entries of their inverses are still polynomial in $\partial_x$. Thus, applying $\hat{E}_2^{-1}$ to the right hand side $\hat{\vec{g}}$, amounts to taking derivatives of $\hat{\vec{g}}$ and making linear combinations of them. If the plane $\mathbb{R}$ is split into subdomains $\mathbb{R}^- \times \mathbb{R}$ and $\mathbb{R}^+ \times \mathbb{R}$, the application of $\hat{E}_2^{-1}$ and $\hat{F}_2^{-1}$ to a vector can be done for each subdomain independently. No communication between the subdomains is necessary.

At first glance, it is surprising that the two-dimensional Stokes equations can be mainly characterized by the scalar fourth order differential operator $\Delta(-\nu\Delta + c)$. But one should note that the stream function formulation gives the same differential equation for the stream function in the two-dimensional case (cf. [8]). More interestingly, in the three-dimensional case the Smith factorization yields a representation of the system as two decoupled scalar equations; cf. Section 5.1.

## 3. A new algorithm for the Stokes equations

Our goal is to write for the Stokes equations on the whole plane divided into two half-planes an algorithm converging in two iterations. Section 2.2 shows that the design of an algorithm for the fourth order operator $\mathcal{B} := \Delta\mathcal{L}_2 = \Delta(-\nu\Delta + c)$ is a key ingredient for this task. Therefore, we derive an algorithm for the operator $\mathcal{B}$ and then, via the Smith factorization, we recast it in a new algorithm for the Stokes system.

### 3.1. An optimal algorithm for the scalar fourth order operator.

We consider the following problem: Find $\phi : \mathbb{R}^2 \to \mathbb{R}$ such that

$$\mathcal{B}(\phi) = g \text{ in } \mathbb{R}^2, \qquad |\phi(\vec{x})| \to 0 \text{ for } |\vec{x}| \to \infty \tag{3.1}$$

where $g$ is a given right hand side. The domain $\Omega$ is decomposed into two half-planes $\Omega_1 = \mathbb{R}^- \times \mathbb{R}$ and $\Omega_2 = \mathbb{R}^+ \times \mathbb{R}$. Let the interface $\{0\} \times \mathbb{R}$ be denoted by $\Gamma$ and $(\vec{n}_i)_{i=1,2}$ be the outward normal of $(\Omega_i)_{i=1,2}$. The algorithm, we propose, is given as follows:

**Algorithm 3.1.** *We choose the initial values $\phi_1^0$ and $\phi_2^0$ such that $\phi_1^0 = \phi_2^0$ and $\mathcal{L}_2\phi_1^0 = \mathcal{L}_2\phi_2^0$ on $\Gamma$. We obtain $(\phi_i^{n+1})_{i=1,2}$ from $(\phi_i^n)_{i=1,2}$ by the following iterative procedure:*

**Correction step**. *We compute the corrections $(\tilde{\phi}_i^{n+1})_{i=1,2}$ as solutions of the homogeneous local problems*

$$\begin{cases} \mathcal{B}\tilde{\phi}_1^{n+1} = 0 \text{ in } \Omega_1, \\ \lim\limits_{|\vec{x}|\to\infty} |\tilde{\phi}_1^{n+1}| = 0, \\ \dfrac{\partial\tilde{\phi}_1^{n+1}}{\partial\vec{n}_1} = \gamma_1^n \text{ on } \Gamma, \\ \dfrac{\partial\mathcal{L}_2\tilde{\phi}_1^{n+1}}{\partial\vec{n}_1} = \gamma_2^n \text{ on } \Gamma, \end{cases} \qquad \begin{cases} \mathcal{B}\tilde{\phi}_2^{n+1} = 0 \text{ in } \Omega_2, \\ \lim\limits_{|\vec{x}|\to\infty} |\tilde{\phi}_2^{n+1}| = 0, \\ \dfrac{\partial\tilde{\phi}_2^{n+1}}{\partial\vec{n}_2} = \gamma_1^n \text{ on } \Gamma, \\ \dfrac{\partial\mathcal{L}_2\tilde{\phi}_2^{n+1}}{\partial\vec{n}_2} = \gamma_2^n \text{ on } \Gamma, \end{cases} \tag{3.2}$$

*where* $\gamma_1^n = -\dfrac{1}{2}\left(\dfrac{\partial\phi_1^n}{\partial\vec{n}_1} + \dfrac{\partial\phi_2^n}{\partial\vec{n}_2}\right)$ *and* $\gamma_2^n = -\dfrac{1}{2}\left(\dfrac{\partial\mathcal{L}_2\phi_1^n}{\partial\vec{n}_1} + \dfrac{\partial\mathcal{L}_2\phi_2^n}{\partial\vec{n}_2}\right)$.

**Updating step.** *We update $(\phi_i^{n+1})_{i=1,2}$ by solving the local problems*

$$\begin{cases} \mathcal{B}\phi_1^{n+1} = g \text{ in } \Omega_1, \\ \lim\limits_{|\vec{x}|\to\infty} |\phi_1^{n+1}| = 0, \\ \phi_1^{n+1} = \phi_1^n + \delta_1^{n+1} \text{ on } \Gamma, \\ \mathcal{L}_2\phi_1^{n+1} = \mathcal{L}_2\phi_1^n + \delta_2^{n+1} \text{ on } \Gamma, \end{cases} \qquad \begin{cases} \mathcal{B}\phi_2^{n+1} = g \text{ in } \Omega_2, \\ \lim\limits_{|\vec{x}|\to\infty} |\phi_2^{n+1}| = 0, \\ \phi_2^{n+1} = \phi_2^n + \delta_1^{n+1} \text{ on } \Gamma, \\ \mathcal{L}_2\phi_2^{n+1} = \mathcal{L}_2\phi_2^n + \delta_2^{n+1} \text{ on } \Gamma, \end{cases} \tag{3.3}$$

*where* $\delta_1^{n+1} = \dfrac{1}{2}(\tilde{\phi}_1^{n+1} + \tilde{\phi}_2^{n+1})$ *and* $\delta_2^{n+1} = \dfrac{1}{2}(\mathcal{L}_2\tilde{\phi}_1^{n+1} + \mathcal{L}_2\tilde{\phi}_2^{n+1})$.

This algorithm has the proposed remarkable property. Formally we can show:

**Proposition 3.2.** *Algorithm* 3.1 *converges in two iterations.*

*Proof.* The equations and the algorithm are linear. It suffices to prove convergence to zero of the above algorithm when $g \equiv 0$. We make use of the Fourier transform in the $y$ direction. First of all, as $\phi_1^0 = \phi_2^0$ and $\mathcal{L}_2\phi_1^0 = \mathcal{L}_2\phi_2^0$ on $\Gamma$, from (3.3)

we obtain the same properties for $\phi_1^1$ and $\phi_2^1$. Then, note that at each step of the algorithm $\phi_i^n$ satisfies the homogeneous equation in each subdomain

$$\hat{\mathcal{B}}\hat{\phi}_i^n(x,k) = (\partial_{xx} - k^2)(-\nu(\partial_{xx} - k^2) + c)\hat{\phi}_i^n(x,k) = 0. \tag{3.4}$$

For each $k \in \mathbb{R}$, (3.4) is a fourth order ordinary differential equation in $x$. The solution in each domain tends to 0 as $|x|$ tends to $\infty$. Just in order to simplify computations we assume $c > 0$. See [18] for the case $c = 0$. Therefore, we get

$$\begin{array}{l} \hat{\phi}_1^n(x,k) = \alpha_1^n(k)e^{|k|x} + \beta_1^n(k)e^{\lambda(k)x}, \\ \hat{\phi}_2^n(x,k) = \alpha_2^n(k)e^{-|k|x} + \beta_2^n(k)e^{-\lambda(k)x} \end{array} \tag{3.5}$$

with $\lambda(k) = \sqrt{c/\nu + k^2}$. The first continuity relation $\mathcal{L}_2\phi_1^1 = \mathcal{L}_2\phi_2^1$ on the interface $\Gamma$ leads to $\alpha_1^1(k) = \alpha_2^1(k)$ as

$$\begin{array}{rcl} \hat{\mathcal{L}}_2\hat{\phi}_i^1(0,k) & = & (-\nu(\partial_{xx} - k^2) + c)\hat{\phi}_i^1(0,k) \\ & = & -\nu(-k^2 + \lambda^2(k))\beta_i^1(k) + c(\alpha_i^1(k) + \beta_i^1(k)) = c\alpha_i^1(k),\ i = 1,2, \end{array}$$

and from $\phi_1^1 = \phi_2^1$ on $\Gamma$ we finally get $\beta_1^1(k) = \beta_2^1(k)$. Therefore, we can omit the subscript indicating the number of the subdomain in $\alpha$ and $\beta$. Then, we can compute $\gamma_1^1$, $\gamma_2^1$ used by the *correction step* (3.2):

$$\begin{array}{l} \gamma_1^1 = -(\alpha^1(k)|k| + \beta^1(k)\lambda(k)), \\ \gamma_2^1 = -\alpha^1(k)|k|c. \end{array}$$

A direct computation shows that the solutions of the *correction step* $\tilde{\phi}_i^2$, $i = 1,2$, whose expressions are of the form (3.5) are given by

$$\begin{array}{l} \widehat{\tilde{\phi}_1^2}(x,k) = -\alpha^1(k)e^{|k|x} - \beta^1(k)e^{\lambda(k)x}, \\ \widehat{\tilde{\phi}_2^2}(x,k) = -\alpha^1(k)e^{-|k|x} - \beta^1(k)e^{-\lambda(k)x}. \end{array}$$

Inserting this into (3.3) shows that the right hand side of the boundary conditions are zero. Since we assumed $g \equiv 0$, this shows that $\hat{\phi}_i^2 = 0$ for $i = 1,2$. □

3.2. **From the fourth order operator $\mathcal{B}$ to the Stokes system.** After having found an optimal algorithm which converges in two steps for the fourth order operator $\mathcal{B}$ problem, we focus on the Stokes system (2.1)-(2.2) by translating this algorithm into an algorithm for the Stokes system. It suffices to replace the operator $\mathcal{B}$ by the Stokes system and $\phi$ by the last component $(F_2(\vec{u},p)^T)_3$ of the vector $F_2(\vec{u},p)^T$ in the boundary conditions, by using formula (2.5).

The algorithm reads:

**Algorithm 3.3.** *We choose the initial values $(\vec{u}_1^0, p_1^0)$ and $(\vec{u}_2^0, p_2^0)$ such that*

$$(F_2(\vec{u}_1^0, p_1^0)^T)_3 = (F_2(\vec{u}_2^0, p_2^0)^T)_3$$

*and*

$$\mathcal{L}_2(F_2(\vec{u}_1^0, p_1^0)^T)_3 = \mathcal{L}_2(F_2(\vec{u}_2^0, p_2^0)^T)_3$$

*on $\Gamma$. We compute $((\vec{u}_i^{n+1}, p_i^{n+1}))_{i=1,2}$ from $((\vec{u}_i^n, p_i^n))_{i=1,2}$ by the following iterative procedure:*

**Correction step**. *We compute the corrections $((\vec{\tilde{u}}_i^{n+1}, \tilde{p}_i^{n+1}))_{i=1,2}$ as the solution*

*of the homogeneous local problems:*

(3.6)
$$\begin{cases} \mathcal{S}_2(\vec{\tilde{u}}_1^{n+1}, \tilde{p}_1^{n+1}) = 0 \ \textit{in } \Omega_1, \\ \lim\limits_{|\vec{x}|\to\infty} |\vec{\tilde{u}}_1^{n+1}| = 0, \\ \dfrac{\partial (F_2(\vec{\tilde{u}}_1^{n+1}, \tilde{p}_1^{n+1})^T)_3}{\partial \vec{n}_1} = \gamma_1^n \ \textit{on } \Gamma, \\ \dfrac{\partial \mathcal{L}_2(F_2(\vec{\tilde{u}}_1^{n+1}, \tilde{p}_1^{n+1})^T)_3}{\partial \vec{n}_1} = \gamma_2^n \ \textit{on } \Gamma, \end{cases} \quad \begin{cases} \mathcal{S}_2(\vec{\tilde{u}}_2^{n+1}, \tilde{p}_2^{n+1}) = 0 \ \textit{in } \Omega_2, \\ \lim\limits_{|\vec{x}|\to\infty} |\vec{\tilde{u}}_2^{n+1}| = 0, \\ \dfrac{\partial (F_2(\vec{\tilde{u}}_2^{n+1}, \tilde{p}_2^{n+1})^T)_3}{\partial \vec{n}_2} = \gamma_1^n \ \textit{on } \Gamma, \\ \dfrac{\partial \mathcal{L}_2(F_2(\vec{\tilde{u}}_2^{n+1}, \tilde{p}_2^{n+1})^T)_3}{\partial \vec{n}_2} = \gamma_2^n \ \textit{on } \Gamma, \end{cases}$$

*where*

$$\gamma_1^n = -\frac{1}{2}\left(\frac{\partial (F_2(\vec{u}_1^n, p_1^n)^T)_3}{\partial \vec{n}_1} + \frac{\partial (F_2(\vec{u}_2^n, p_2^n)^T)_3}{\partial \vec{n}_2}\right),$$
$$\gamma_2^n = -\frac{1}{2}\left(\frac{\partial \mathcal{L}_2(F_2(\vec{u}_1^n, p_1^n)^T)_3}{\partial \vec{n}_1} + \frac{\partial \mathcal{L}_2(F_2(\vec{u}_2^n, p_2^n)^T)_3}{\partial \vec{n}_2}\right).$$

**Updating step.** *We update* $((\vec{u}_i^{n+1}, p_i^{n+1}))_{i=1,2}$ *by solving the local problems:*

$$\begin{cases} \mathcal{S}_2(\vec{u}_i^{n+1}, p_i^{n+1}) = \vec{g} \ \ \textit{in } \Omega_i, \\ \lim\limits_{|\vec{x}|\to\infty} |\vec{u}_i^{n+1}| = 0, \\ (F_2(\vec{u}_i^{n+1}, p_i^{n+1})^T)_3 = (F_2(\vec{u}_i^n, p_i^n)^T)_3 + \delta_1^{n+1} \ \textit{on } \Gamma, \\ \mathcal{L}_2(F_2(\vec{u}_i^{n+1}, p_i^{n+1})^T)_3 = \mathcal{L}_2(F_2(\vec{u}_i^n, p_i^n)^T)_3 + \delta_2^{n+1} \ \textit{on } \Gamma, \end{cases} \tag{3.7}$$

*where*

$$\delta_1^{n+1} = \frac{1}{2}[(F_2(\vec{\tilde{u}}_1^{n+1}, \tilde{p}_1^{n+1})^T)_3 + (F_2(\vec{\tilde{u}}_2^{n+1}, \tilde{p}_2^{n+1})^T)_3],$$
$$\delta_2^{n+1} = \frac{1}{2}[\mathcal{L}_2(F_2(\vec{\tilde{u}}_1^{n+1}, \tilde{p}_1^{n+1})^T)_3 + \mathcal{L}_2(F_2(\vec{\tilde{u}}_2^{n+1}, \tilde{p}_2^{n+1})^T)_3].$$

This algorithm seems quite complex since it involves third order derivatives of the unknowns in the boundary conditions on $(F_2(\vec{\tilde{u}}_i, \tilde{p}_i)^T)_3$. Writing $\vec{u}_i = (u_i, v_i)$ and using $(F_2(\vec{\tilde{u}}_i, \tilde{p}_i)^T)_3 = \tilde{v}_i$, it is possible to simplify it. By using the Stokes equations in the subdomains, we can lower the degree of the derivatives in the boundary conditions. In order to ease the presentation in Algorithm 3.4 we do not mention that the solutions tend to zero as $|\vec{x}| \to \infty$. If we denote the $k$-th component of the unit outward normal vector $\vec{n}_i$ of $\Omega_i$ by $n_{i,k}$, we obtain for two subdomains the following:

**Algorithm 3.4.** *We choose the initial values* $(u_1^0, v_1^0, p_1^0)$ *and* $(u_2^0, v_2^0, p_2^0)$ *such that* $v_1^0 = v_2^0$ *and*

$$\nu \frac{\partial u_1^0}{\partial \vec{n}_1} - p_1^0 n_{1,1} = -\left(\nu \frac{\partial u_2^0}{\partial \vec{n}_2} - p_2^0 n_{2,1}\right)$$

*on* $\Gamma$. *We compute* $((u_i^{n+1}, v_i^{n+1}, p_i^{n+1}))_{i=1,2}$ *from* $((u_i^n, v_i^n, p_i^n))_{i=1,2}$ *by the following iterative procedure:*

**Correction step**. *We compute the corrections* $((\tilde{u}_i^{n+1}, \tilde{v}_i^{n+1}, \tilde{p}_i^{n+1}))_{i=1,2}$ *as the solution of the homogeneous local problems:*

$$\begin{cases} \mathcal{S}_2(\tilde{u}_1^{n+1}, \tilde{v}_1^{n+1}, \tilde{p}_1^{n+1}) = 0 \ \textit{in } \Omega_1, \\ \nu \dfrac{\partial \tilde{v}_1^{n+1}}{\partial \vec{n}_1} = \gamma_1^n \ \textit{on } \Gamma, \\ \tilde{u}_1^{n+1} = \gamma_{2,1}^n \ \textit{on } \Gamma, \end{cases} \quad \begin{cases} \mathcal{S}_2(\tilde{u}_2^{n+1}, \tilde{v}_2^{n+1}, \tilde{p}_2^{n+1}) = 0 \ \textit{in } \Omega_2, \\ \nu \dfrac{\partial \tilde{v}_2^{n+1}}{\partial \vec{n}_2} = \gamma_1^n \ \textit{on } \Gamma, \\ \tilde{u}_2^{n+1} = \gamma_{2,2}^n \ \textit{on } \Gamma, \end{cases} \tag{3.8}$$

*where* $\gamma_1^n = -\frac{1}{2}\left(\nu\frac{\partial v_1^n}{\partial \vec{n}_1} + \nu\frac{\partial v_2^n}{\partial \vec{n}_2}\right)$ *and* $\gamma_{2,i}^n = (-1)^i\frac{1}{2}(u_1^n - u_2^n)$.

**Updating step.** *We update* $((u_i^{n+1}, v_i^{n+1}, p_i^{n+1}))_{i=1,2}$ *by solving the local problems:*

$$\begin{cases} \mathcal{S}_2(u_i^{n+1}, v_i^{n+1}, p_i^{n+1}) = \vec{g} \text{ in } \Omega_i, \\ \nu\dfrac{\partial u_i^{n+1}}{\partial \vec{n}_i} - p_i^{n+1}n_{i,1} = \nu\dfrac{\partial u_i^n}{\partial \vec{n}_i} - p_i^n n_{i,1} + \delta_{ij}^{n+1} \text{ on } \Gamma, \\ v_i^{n+1} = v_i^n + \dfrac{1}{2}(\tilde{v}_1^n + \tilde{v}_2^n) \text{ on } \Gamma, \end{cases} \tag{3.9}$$

*where* $\delta_{ij}^{n+1} = \frac{1}{2}\left(\nu\frac{\partial \tilde{u}_i^{n+1}}{\partial \vec{n}_i} - \tilde{p}_i^{n+1}n_{i,1}\right) - \frac{1}{2}\left(\nu\frac{\partial \tilde{u}_j^{n+1}}{\partial \vec{n}_j} - \tilde{p}_j^{n+1}n_{j,1}\right)$ *and* $j = 3 - i$.

**Lemma 3.5.** *Consider the model case* $\Omega = \mathbb{R}^2$, $\Omega_1 = \mathbb{R}^- \times \mathbb{R}$ *and* $\Omega_2 = \mathbb{R}^+ \times \mathbb{R}$. *We assume that all variables vanish at infinity. Then, the Algorithms* 3.3 *and* 3.4 *are equivalent.*

*Proof.* First, notice $(F_2(\vec{\tilde{u}}_i^n, \tilde{p}_i^n)^T)_3 = \tilde{v}_i^n$ and $(F_2(\vec{u}_i^n, p_i^n)^T)_3 = v_i^n$. Thus, the first interface conditions of (3.6), respectively, (3.7) are obviously the same as the first interface conditions of (3.8), respectively, the second one of (3.9).

To prove the complete equivalence between these algorithms, we start with the local problems in $\Omega_1$ by transforming the second interface condition of the correction step (3.6):

$$\partial_x\mathcal{L}_2\tilde{v}_1^{n+1} = -\frac{1}{2}\partial_x(\mathcal{L}_2 v_1^n - \mathcal{L}_2 v_2^n) \quad \text{on } \Gamma.$$

Using the second equation of the Stokes system,

$$\mathcal{L}_2(F_2(\vec{u}_i^n, p_i^n)^T)_3 = (-\nu\Delta + c)v_i^n = \partial_y p_i^n + f_2, \quad i = 1, 2,$$

we obtain

$$\begin{aligned} \partial_x(-\partial_y\tilde{p}_1^{n+1}) &= -\frac{1}{2}\partial_x((-\partial_y p_1^n + f_2) - (-\partial_y p_2^n + f_2)), \\ &= -\frac{1}{2}\partial_y(-\partial_x p_1^n + \partial_x p_2^n) \quad \text{on } \Gamma. \end{aligned}$$

Interchanging the partial derivatives and using the first equation of the Stokes system and the fact that all functions vanish at infinity, by integrating with respect to $y$ we get

$$\begin{aligned} \partial_y(\mathcal{L}_2\tilde{u}_1^{n+1}) = -\frac{1}{2}\partial_y(\mathcal{L}_2 u_1^n - \mathcal{L}_2 u_2^n) \quad \text{on } \Gamma \Leftrightarrow \\ \mathcal{L}_2\tilde{u}_1^{n+1} = -\frac{1}{2}(\mathcal{L}_2 u_1^n - \mathcal{L}_2 u_2^n) \quad \text{on } \Gamma. \end{aligned} \tag{3.10}$$

Differentiating the first interface condition (3.6) with respect to $y$ and using the incompressibility constraint ($\partial_y\tilde{v}_i^{n+1} = -\partial_x\tilde{u}_i^{n+1}$, $i = 1, 2$) yields

$$-\nu\partial_{xx}\tilde{u}_1^{n+1} = \frac{1}{2}\nu\partial_{xx}(u_1^n - u_2^n) \quad \text{on } \Gamma. \tag{3.11}$$

We subtract (3.11) from (3.10). Thus, we obtain

$$\begin{aligned} (-\nu\partial_{yy} + c)\tilde{u}_1^{n+1} = -\frac{1}{2}(-\nu\partial_{yy} + c)(u_1^n - u_2^n) \quad \text{on } \Gamma \Leftrightarrow \\ \tilde{u}_1^{n+1} = -\frac{1}{2}(u_1^n - u_2^n) \quad \text{on } \Gamma, \end{aligned}$$

which is exactly the second transmission condition (3.8) of the correction step.

Next, we consider the second interface condition of the updating step (3.7). Using again the second equation of the Stokes system we obtain:

$$\begin{aligned} \partial_y p_1^{n+1} &= \partial_y p_1^n + \frac{1}{2}\left(\partial_y \tilde{p}_1^{n+1} + \partial_y \tilde{p}_2^{n+1}\right) \quad \text{on } \Gamma \Leftrightarrow \\ p_1^{n+1} &= p_1^n + \frac{1}{2}\left(\tilde{p}_1^{n+1} + \tilde{p}_1^{n+1}\right) \quad \text{on } \Gamma. \end{aligned} \tag{3.12}$$

Of course, one could stop with boundary condition (3.12), but we will derive a more natural boundary condition. Therefore, we also use the second transmission condition of (3.7) and mix both conditions. Differentiating the first interface condition of (3.7) with respect to $y$ gives

$$\partial_y v_1^{n+1} = \partial_y v_1^n + \frac{1}{2}\partial_y\left(\tilde{v}_1^{n+1} + \tilde{v}_2^{n+1}\right) \quad \text{on } \Gamma.$$

Now, using the incompressibility constraint yields

$$-\nu\partial_x u_1^{n+1} = -\nu\partial_x u_1^n - \frac{1}{2}\nu\partial_x\left(\tilde{u}_1^{n+1} + \tilde{u}_2^{n+1}\right) \quad \text{on } \Gamma. \tag{3.13}$$

Adding (3.12) and (3.13) we end up with

$$\begin{aligned} -\nu\partial_x u_1^{n+1} + p_1^{n+1} &= -\nu\partial_x u_1^n + p_1^n \\ &\quad + \frac{1}{2}\left(-\nu\partial_x \tilde{u}_1^{n+1} + \tilde{p}_1^{n+1}\right) + \frac{1}{2}\left(-\nu\partial_x \tilde{u}_2^{n+1} + \tilde{p}_2^{n+1}\right), \end{aligned}$$

which is exactly the first transmission condition (3.9) of the updating step. The reformulation of the initial conditions can be done analogously.

The same computations can be performed for subdomain $\Omega_2$. □

*Remark* 3.6. The assumption that the pressure vanishes at infinity is artificial. If we only use that the derivatives of $p$ vanish, then the first interface condition of the updating step is determined only up to a constant. In practice, one could easily avoid this problem by providing an appropriate coarse space.

In order to write the resulting algorithm in an intrinsic form, we introduce the stress $\vec{\sigma}^i(\vec{u}, p)$ for each subdomain $\Omega_i$ on the interface for a velocity $\vec{u} = (u, v)$, a pressure $p$ and the normal vector $\vec{n}_i$. If $\partial n_i = \partial_x$, we have the following formula in cartesian coordinates:

$$\vec{\sigma}^i(\vec{u}, p) = (\nu\frac{\partial u}{\partial x} - p, \frac{\nu}{2}(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y})).$$

For any vector $\vec{u}$ its normal (resp. tangential) component on the interface is $u_{\vec{n}_i} = \vec{u}\cdot\vec{n_i}$ (resp. $\vec{u}_{\vec{\tau}_i} = (I - \vec{n}_i \otimes \vec{n}_i)\,\vec{u}$). We denote $\sigma^i_{\vec{n}_i} := \sigma^i_{\vec{n}_i}(\vec{u}_i, p_i)\cdot\vec{n}_i$ and $\vec{\sigma}^i_{\vec{\tau}_i} := (I - \vec{n}_i \otimes \vec{n}_i)\,\vec{\sigma}^i$ as the normal and tangential parts of $\vec{\sigma}^i$, respectively.

Moreover, we notice that boundary conditions (3.8) are equivalent to conditions (3.14). Indeed, it suffices to differentiate w.r.t the $y$ direction the boundary condition on the normal velocity in (3.8), multiply it by $\nu$ and add it to the boundary condition on the normal derivative of the tangential velocity in (3.8).

We can now generalize the previous algorithm to a more general decomposition into non-overlapping subdomains: $\bar{\Omega} = \bigcup_{i=1}^N \bar{\Omega}_i$ and denote by $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j$ the

interface between subdomains $\Omega_i$ and $\Omega_j$, $i \neq j$. The new algorithm for the Stokes system reads:

**Algorithm 3.7.** *Starting with an initial guess* $((\vec{u}_i^0, p_i^0))_{i=0}^N$ *satisfying* $\vec{u}^0_{i,\vec{\tau}_i} = \vec{u}^0_{j,\vec{\tau}_j}$ *and* $\sigma^i_{\vec{n}_i}(\vec{u}_i^0, p_i^0) = \sigma^j_{\vec{n}_j}(\vec{u}_j^0, p_j^0)$ *on* $\Gamma_{ij}$, $\forall i, j$, $i \neq j$, *the* ***correction step*** *is expressed as follows for* $1 \leq i \leq N$*:*

$$
\text{(3.14)} \quad \begin{cases} \mathcal{S}_2(\tilde{\vec{u}}_i^{n+1}, \tilde{p}_i^{n+1}) = 0 & \text{in } \Omega_i, \\ \tilde{u}^{n+1}_{i,\vec{n}_i} = -\dfrac{1}{2}(u^n_{i,\vec{n}_i} + u^n_{j,\vec{n}_j}) & \text{on } \Gamma_{ij}, \\ \vec{\sigma}^i_{\vec{\tau}_i}(\tilde{\vec{u}}_i^{n+1}, \tilde{p}_i^{n+1}) = -\dfrac{1}{2}(\vec{\sigma}^i_{\vec{\tau}_i}(\vec{u}_i^n, \tilde{p}_i^n) + \vec{\sigma}^j_{\vec{\tau}_j}(\vec{u}_j^n, \tilde{p}_j^n)) & \text{on } \Gamma_{ij}. \end{cases}
$$

*followed by an* **updating step** *for* $1 \leq i \leq N$*:*

$$
\text{(3.15)} \quad \begin{cases} \mathcal{S}_2(\vec{u}_i^{n+1}, p_i^{n+1}) = \vec{g} & \text{in } \Omega_i, \\ \vec{u}^{n+1}_{i,\vec{\tau}_i} = \vec{u}^n_{i,\vec{\tau}_i} + \dfrac{1}{2}(\tilde{\vec{u}}^{n+1}_{i,\vec{\tau}_i} + \tilde{\vec{u}}^{n+1}_{j,\vec{\tau}_j}) & \text{on } \Gamma_{ij}, \\ \sigma^i_{\vec{n}_i}(\vec{u}_i^{n+1}, p_i^{n+1}) = \sigma^i_{\vec{n}_i}(\vec{u}_i^n, p_i^n), & \\ \qquad + \dfrac{1}{2}(\sigma^i_{\vec{n}_i}(\tilde{\vec{u}}_i^{n+1}, \tilde{p}_i^{n+1}) + \sigma^j_{\vec{n}_j}(\tilde{\vec{u}}_j^{n+1}, \tilde{p}_j^{n+1})) & \text{on } \Gamma_{ij}. \end{cases}
$$

The boundary conditions in the correction step involve the normal velocity and the tangential stress, whereas in the updating step the tangential velocity and the normal stress are involved. As we will see in Section 5, in three dimensions the algorithm has the same definition.

**Proposition 3.8.** *For a domain* $\Omega = \mathbb{R}^2$ *divided into two non-overlapping half-planes, Algorithms* 3.3 *and* 3.7 *are equivalent and both converge in two iterations.*

*Proof.* The equivalence of both algorithms has already been shown. The convergence in two steps of Algorithm 3.7 is obvious, since the algorithm was derived directly from Algorithm 3.3 which converges in two steps. □

## 4. Extension to the Oseen equations

The next step is an extension of this technique to the Oseen equations

$$
\text{(4.1)} \quad \begin{cases} -\nu\Delta\vec{u} + \vec{b} \cdot \nabla\vec{u} + c\vec{u} + \nabla p = \vec{f} & \text{in } \Omega, \\ \nabla \cdot \vec{u} = 0 & \text{in } \Omega. \end{cases}
$$

In comparison to the Stokes equations we have added the convective term $\vec{b} \cdot \nabla\vec{u}$. Now, the equation is no longer symmetric. Standard linearization techniques for the incompressible Navier-Stokes equations lead to the Oseen problem. Therefore, the efficient numerical solution of the Oseen problem is very important. The Oseen operator is given by

$$
\mathcal{O}_d(\vec{u}, p) = \left(-\nu\Delta\vec{u} + \vec{b} \cdot \nabla\vec{u} + c\vec{u} + \nabla p, \nabla \cdot \vec{u}\right)^T, \; d = 2, 3.
$$

Our aim is to derive a domain decomposition method which is robust with respect to the viscosity $\nu$. To our knowledge up to now this is an unsolved problem. Here we just want to give a brief outline of how the Smith factorization can be used in order to derive a new domain decomposition method for the Oseen equations. For the details we refer to [5]. We only consider the two-dimensional case. Applying

the Smith factorization to the Fourier transform of $\mathcal{O}_2$ (in $y$ direction) yields $\hat{\mathcal{O}}_2 = \hat{E}_2^{\mathcal{O}} \hat{D}_2^{\mathcal{O}} \hat{F}_2^{\mathcal{O}}$. The diagonal matrix is given by the Fourier transform of

$$D_2^{\mathcal{O}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \mathcal{L}_2^{\mathcal{O}} \Delta \end{pmatrix}$$

with the second order differential operator $\mathcal{L}_2^{\mathcal{O}} u = -\nu \Delta u + \vec{b} \cdot \nabla u + cu$. Similarly to the Stokes case, we exhibit an iterative algorithm for the scalar fourth order problem given by the differential operator $\mathcal{L}_2^{\mathcal{O}} \Delta$, which converges in at most two steps in the case of $\Omega = \mathbb{R}^2$ and $\Omega_1 = \mathbb{R}^+ \times \mathbb{R}$ and $\Omega_2 = \mathbb{R}^- \times \mathbb{R}$. Our algorithm is given as follows:

**Algorithm 4.1.** *We choose the initial values $\phi_1^0$, $\phi_2^0$ such that*

$$\mathcal{L}_2^{\mathcal{O}} \phi_1^0 = \mathcal{L}_2^{\mathcal{O}} \phi_2^0, \qquad \phi_1^0 = \phi_2^0 \qquad \text{on } \Gamma = \partial\Omega_1 \cap \partial\Omega_2.$$

*Then, we obtain $(\phi_i^{n+1})_{i=1,2}$ from $(\phi_i^n)_{i=1,2}$ by the following procedure.*

**Correction step.** *We compute the corrections $(\tilde{\phi}_i^{n+1})_{i=1,2}$ as solutions of*

$$\text{(4.2)} \quad \begin{cases} \mathcal{L}_2^{\mathcal{O}} \Delta \tilde{\phi}_i^{n+1} = 0 \quad \text{in } \Omega_i, \\ \lim\limits_{|\vec{x}| \to \infty} \tilde{\phi}_i^{n+1} = 0, \\ \dfrac{\partial (\mathcal{L}_2^{\mathcal{O}} \tilde{\phi}_i^{n+1})}{\partial \vec{n}_i} = -\dfrac{1}{2} \left( \dfrac{\partial (\mathcal{L}_2^{\mathcal{O}} \phi_1^n)}{\partial \vec{n}_1} + \dfrac{\partial (\mathcal{L}_2^{\mathcal{O}} \phi_2^n)}{\partial \vec{n}_2} \right) \quad \text{on } \Gamma, \\ \left( \nu \dfrac{\partial}{\partial \vec{n}_i} - \dfrac{1}{2} \vec{b} \cdot \vec{n}_i \right) \tilde{\phi}_i^{n+1} = -\dfrac{1}{2} \nu \left( \dfrac{\partial \phi_1^n}{\partial \vec{n}_1} + \dfrac{\partial \phi_2^n}{\partial \vec{n}_2} \right) \quad \text{on } \Gamma. \end{cases}$$

**Updating step.** *We update $(\phi_i^{n+1})_{i=1,2}$ by solving the local problems:*

$$\text{(4.3)} \quad \begin{cases} \mathcal{L}_2^{\mathcal{O}} \Delta \phi_i^{n+1} = g \quad \text{in } \Omega_i, \\ \lim\limits_{|\vec{x}| \to \infty} \phi_i^{n+1} = 0, \\ \mathcal{L}_2^{\mathcal{O}} \phi_i^{n+1} = \mathcal{L}_2^{\mathcal{O}} \phi_i^n + \dfrac{1}{2} \left( \mathcal{L}_2^{\mathcal{O}} \tilde{\phi}_i^{n+1} + \mathcal{L}_2^{\mathcal{O}} \tilde{\phi}_2^{n+1} \right) \quad \text{on } \Gamma, \\ \phi_i^{n+1} = \phi_i^n + \dfrac{1}{2} (\tilde{\phi}_1^{n+1} + \tilde{\phi}_2^{n+1}) \quad \text{on } \Gamma. \end{cases}$$

Using the same technique as for the Stokes equations, we could derive the following algorithm, which converges in two steps for our model problem given by $\Omega = \mathbb{R}^2$, $\Omega_1 = \mathbb{R}^- \times \mathbb{R}$ and $\Omega_2 = \mathbb{R}^+ \times \mathbb{R}$. In order to write it in a more compact form we define the following quantity (which is very similar to the stress tensor)

$$\vec{\kappa}^i := \vec{\kappa}^i(\vec{u}, p) := \nu \frac{\partial \vec{u}}{\partial \vec{n}_i} - p \vec{n}_i.$$

As before, the normal (resp. tangential) component of the velocity on the interface is $u_{\vec{n}_i} = \vec{u} \cdot \vec{n}_i$ (resp. $\vec{u}_{\vec{\tau}_i} = (I - \vec{n}_i \otimes \vec{n}_i)\, \vec{u}$) and we denote by $\kappa^i_{\vec{n}_i} := \kappa^i(\vec{u}_i, p_i) \cdot \vec{n}_i$ and $\vec{\kappa}^i_{\vec{\tau}_i} := (I - \vec{n}_i \otimes \vec{n}_i)\, \vec{\kappa}^i$, the normal and tangential parts of $\vec{\kappa}^i$, respectively.

**Algorithm 4.2.** *Starting with an initial guess satisfying* $\vec{u}^0_{i,\vec{\tau}_i} = \vec{u}^0_{j,\vec{\tau}_j}$ *and* $\kappa^i_{\vec{n}_i} = \kappa^j_{\vec{n}_j}$ *on* $\Gamma_{ij}$*, the* **correction step** *is expressed as follows for* $1 \leq i \leq N$*:*

(4.4)
$$\begin{cases} \mathcal{O}_2(\vec{\tilde{u}}_i^{n+1}, \tilde{p}_i^{n+1}) = 0 \quad \text{in } \Omega_i, \\ \vec{\kappa}^i_{\vec{\tau}_i}(\vec{\tilde{u}}_i^{n+1}, \tilde{p}_i^{n+1}) - \frac{1}{2}(\vec{b}\cdot\vec{n}_i)\tilde{\vec{u}}^{n+1}_{i,\vec{\tau}_i} = -\frac{1}{2}(\vec{\kappa}^i_{\vec{\tau}_i}(\vec{u}_i^n, p_i^n) + \vec{\kappa}^j_{\vec{\tau}_j}(\vec{u}_j^n, p_j^n)) \quad \text{on } \Gamma_{ij}, \\ (-\nu\partial_{\vec{\tau}_i\vec{\tau}_i} + (\vec{b}\cdot\vec{\tau}_i)\partial_{\vec{\tau}_i} + c)\tilde{u}^{n+1}_{i,\vec{n}_i} + \frac{1}{2}(\vec{b}\cdot\vec{n}_i)\partial_{\vec{n}_i}\tilde{u}^{n+1}_{i,\vec{n}_i} = \gamma^n_{ij} \quad \text{on } \Gamma_{ij} \end{cases}$$

*with* $\gamma^n_{ij} := -\frac{1}{2}(-\nu\partial_{\vec{\tau}_i\vec{\tau}_i} + (\vec{b}\cdot\vec{\tau}_i)\partial_{\vec{\tau}_i} + c + \frac{1}{2}(\vec{b}\cdot\vec{n}_i)\partial_{\vec{n}_i})\left(u^n_{i,\vec{n}_i} + u^n_{j,\vec{n}_j}\right)$.

*The* **updating step** *is given by*

$$\text{(4.5)} \qquad \begin{cases} \mathcal{O}_2(\vec{u}_i^{n+1}, p_i^{n+1}) &=& \vec{f} \quad \text{in } \Omega_i, \\ \vec{u}^{n+1}_{i,\vec{\tau}_i} &=& \vec{u}^n_{i,\vec{\tau}_i} + \frac{1}{2}(\vec{\tilde{u}}^{n+1}_{i,\vec{\tau}_i} + \vec{\tilde{u}}^{n+1}_{j,\vec{\tau}_j}) \quad \text{on } \Gamma_{ij}, \\ \kappa^i_{\vec{n}_i}(\vec{u}_i^{n+1}, p_i^{n+1}) &=& \kappa^i_{\vec{n}_i}(\vec{u}_i^n, p_i^n) + \delta^{n+1} \text{ on } \Gamma_{ij} \end{cases}$$

*with* $\delta^{n+1} = \frac{1}{2}(\kappa^i_{\vec{n}_i}(\tilde{\vec{u}}_i^{n+1}, \tilde{p}_i^{n+1}) + \kappa^j_{\vec{n}_j}(\tilde{\vec{u}}_j^{n+1}, \tilde{p}_j^{n+1}))$.

This algorithm is more complicated than the one for the Stokes equations; but we would like to emphasize, that all interface conditions are intrinsic except the second interface condition in the correction step. There, some tangential derivatives are involved.

*Remark* 4.3. For $\vec{b}\cdot\vec{n}_i = 0$ the interface condition (4.4) can be further simplified. Using the fact that the interface condition is a second order ordinary differential equation in the tangential direction, it can be simply written as

$$\text{(4.6)} \qquad \tilde{u}^{n+1}_{i,\vec{n}_i} = -\frac{1}{2}(u^n_{i,\vec{n}_i} + u^n_{j,\vec{n}_j}) \quad \text{on } \Gamma_{ij}.$$

Thus, in the case $\vec{b} = 0$ we recover the intrinsic Algorithm 3.7 of the Stokes problem.

## 5. The three-dimensional case for the Stokes equations

As one can see, Algorithm 3.7 was derived using the structure of the two-dimensional Stokes operator. Thus, it is not clear what happens in the three-dimensional case. We will show that using the Smith factorization we also end up with the intrinsic Algorithm 3.7.

### 5.1. Smith factorization.

Performing a Fourier transform in $y$- and $z$-directions for the three-dimensional Stokes operator $\mathcal{S}_3$ (with dual variables $k$ and $\eta$), we obtain

$$\text{(5.1)} \qquad \hat{\mathcal{S}}_3 = \begin{pmatrix} \hat{\mathcal{L}}_3 & 0 & 0 & \partial_x \\ 0 & \hat{\mathcal{L}}_3 & 0 & ik \\ 0 & 0 & \hat{\mathcal{L}}_3 & i\eta \\ \partial_x & ik & i\eta & 0 \end{pmatrix}$$

where $\hat{\mathcal{L}}_3 := \nu(-\partial_{xx} + k^2 + \eta^2) + c$ is the Fourier transform of $\mathcal{L}_3 := -\nu\Delta + c$.

Applying the Smith Factorization yields

$$\hat{S}_3 = \hat{E}_3\hat{D}_3\hat{F}_3$$

with matrices

$$\hat{D}_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \hat{\mathcal{L}}_3 & 0 \\ 0 & 0 & 0 & (\partial_{xx} - k^2 - \eta^2)\hat{\mathcal{L}}_3 \end{pmatrix},$$

$$\hat{E}_3 = \hat{T}_3^{-1} \begin{pmatrix} ik\hat{\mathcal{L}}_3 & \nu\partial_{xxx} & -\nu i\eta\partial_x & -\nu\partial_x \\ 0 & \hat{T}_3 & 0 & 0 \\ 0 & i\eta(\nu(k^2+\eta^2)+c) & -\nu(k^2+\eta^2)+c & 0 \\ ik\partial_x & -\partial_{xx} & i\eta & 1 \end{pmatrix},$$

$$\hat{F}_3 = \begin{pmatrix} -\nu(\partial_{xx} - \eta^2) + c & \nu ik\partial_x & \nu i\eta\partial_x & \partial_x \\ 0 & \hat{\mathcal{L}}_3 & 0 & ik \\ 0 & -i\eta & ik & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

$T_3$ is the differential operator in the $y$ and $z$ direction with symbol $ik(\nu(k^2+\eta^2)+c)$.

We see analogously to the two-dimensional case that the Stokes operator $\mathcal{S}_3$ is determined by the diagonal matrix $D_3$. Therefore, it can be represented by the fourth order differential operator $\mathcal{L}_3\Delta$ and the second order differential operator $\mathcal{L}_3$.

5.2. **The three-dimensional algorithm.** Our starting point is the intrinsic Algorithm 3.7. We check in this section that indeed, also in three dimensions, Algorithm 3.7 converges in only two steps in the case of the whole space $\mathbb{R}^3$ divided into the two half spaces.

Let us consider the domain $\Omega := \mathbb{R}^3$ divided into $\Omega_1 := \{(x,y,z) \in \mathbb{R}^3 \mid x < 0\}$ and $\Omega_2 := \{(x,y,z) \in \mathbb{R}^3 \mid x > 0\}$. The common interface is given by $\Gamma := \{(x,y,z) \in \mathbb{R}^3 \mid x = 0\}$. For this special geometry the intrinsic Algorithm 3.7 can be simplified. We write $\vec{u} = (u,v,w)$ . We obtain the following algorithm:

**Algorithm 5.1.** *We start with an initial guess* $((\vec{u}_i^0, p_i^0))_{i=1,2}$ *satisfying*

$$\vec{u}^0_{1,\vec{\tau}_1} = \vec{u}^0_{2,\vec{\tau}_2}, \qquad \sigma^1_{\vec{n}_1}(\vec{u}_1^0, p_1^0) = \sigma^2_{\vec{n}_2}(\vec{u}_2^0, p_2^0) \qquad \text{on } \Gamma.$$

*Compute the following* **correction step** *for* $((\tilde{\vec{u}}_i^{n+1}, \tilde{p}_i^{n+1}))_{i=1,2}$*:*

$$(5.2)\quad \begin{cases} \mathcal{S}_3(\tilde{\vec{u}}_i^{n+1}, \tilde{p}_i^{n+1}) &=& 0 \quad \text{in } \Omega_i, \\ \vec{\sigma}^i_{\vec{\tau}_i}(\tilde{\vec{u}}_i^{n+1}, \tilde{p}_i^{n+1}) &=& -\dfrac{1}{2}(\vec{\sigma}^i_{\vec{\tau}_i}(\vec{u}_i^n, \tilde{p}_i^n) + \vec{\sigma}^j_{\vec{\tau}_j}(\vec{u}_j^n, \tilde{p}_j^n)) \quad \text{on } \Gamma, \\ \tilde{u}_i^{n+1} &=& -\dfrac{1}{2}\left(u_i^n - u_j^n\right) \quad \text{on } \Gamma. \end{cases}$$

*Then the* **updating step** *for* $((\vec{u}_i^{n+1}, p_i^{n+1}))_{i=1,2}$ *is given as follows:*

$$\begin{cases} \mathcal{S}_3(\vec{u}_i^{n+1}, p_i^{n+1}) &=& \vec{g} \quad \text{in } \Omega_i, \\ \vec{u}^{n+1}_{i,\vec{\tau}_i} &=& \vec{u}^n_{i,\vec{\tau}_i} + \dfrac{1}{2}(\tilde{\vec{u}}_{i,\vec{\tau}_i}{}^{n+1} + \tilde{\vec{u}}_{j,\vec{\tau}_j}{}^{n+1}) \quad \text{on } \Gamma, \\ \sigma^i_{\vec{n}_i}(u_i^{n+1}, p_i^{n+1}) &=& \sigma^i_{\vec{n}_i}(u_i^n, p_i^n) \\ &+& \dfrac{1}{2}\left(\sigma^i_{\vec{n}_i}(\tilde{u}_i^{n+1}, p_i^{n+1}) + \sigma^j_{\vec{n}_j}(\tilde{u}_j^{n+1}, p_j^{n+1})\right) \quad \text{on } \Gamma. \end{cases}$$

Algorithm 5.1 yields two completely uncoupled domain decomposition methods for scalar problems.

**Proposition 5.2.** *The decomposition is given by* $\Omega := \mathbb{R}^3$, $\Omega_1 := \{(x,y,z) \in \mathbb{R}^3 \mid x < 0\}$ *and* $\Omega_2 := \{(x,y,z) \in \mathbb{R}^3 \mid x > 0\}$. *Assume that the velocity components* $\vec{u}_i^n$, $\tilde{\vec{u}}_i^n$ *and the pressure components* $p_i^n, \tilde{p}_i^n$ *are given by Algorithm* 5.1. *Then the variables*

$$
\begin{array}{rcl}
v_i^n & = & (F_3(\vec{u}_i^n, p_i^n))_4\,, \qquad \tilde{v}_i^n = \left(F_3(\tilde{\vec{u}}_i^n, \tilde{p}_i^n)\right)_4\,, \\
\gamma_i^n & := & (F_3(\vec{u}_i^n, p_i^n))_3 = -\partial_z v_i^n + \partial_y w_i^n, \\
\tilde{\gamma}_i^n & := & \left(F_3(\tilde{\vec{u}}_i^n, \tilde{p}_i^n)\right)_3 = -\partial_z \tilde{v}_i^n + \partial_y \tilde{w}_i^n
\end{array}
\tag{5.3}
$$

*satisfy for* $i = 1, 2$ *the* **correction step**

$$
\left\{
\begin{array}{lcll}
\Delta \mathcal{L}_3 \tilde{v}_i^{n+1} & = & 0 & \text{in } \Omega_i, \\
\mathcal{L}_3 \tilde{\gamma}_i^{n+1} & = & 0 & \text{in } \Omega_i, \\
\nu \dfrac{\partial \tilde{\gamma}_i^{n+1}}{\partial \vec{n}_i} & = & -\dfrac{1}{2}\nu \left( \dfrac{\partial \gamma_1^n}{\partial \vec{n}_1} + \dfrac{\partial \gamma_2^n}{\partial \vec{n}_2} \right) & \text{on } \Gamma, \\
\dfrac{\partial (\mathcal{L}_3 \tilde{v}_i^{n+1})}{\partial \vec{n}_i} & = & -\dfrac{1}{2} \left( \dfrac{\partial (\mathcal{L}_3 v_i^n)}{\partial \vec{n}_1} + \dfrac{\partial (\mathcal{L}_3 v_2^n)}{\partial \vec{n}_2} \right) & \text{on } \Gamma, \\
\nu \dfrac{\partial \tilde{v}_i^{n+1}}{\partial \vec{n}_i} & = & -\dfrac{1}{2}\nu \left( \dfrac{\partial v_1^n}{\partial \vec{n}_1} + \dfrac{\partial v_2^n}{\partial \vec{n}_2} \right) & \text{on } \Gamma,
\end{array}
\right.
\tag{5.4}
$$

*and the* **updating step** $(i = 1, 2)$

$$
\left\{
\begin{array}{lcll}
\Delta \mathcal{L}_3 v_i^{n+1} & = & (E_3^{-1}\vec{g})_4 & \text{in } \Omega_i, \\
\mathcal{L}_3 \gamma_i^{n+1} & = & (E_3^{-1}\vec{g})_3 & \text{in } \Omega_i, \\
\gamma_i^{n+1} & = & \gamma_i^n + \dfrac{1}{2}(\tilde{\gamma}_1^{n+1} + \tilde{\gamma}_2^{n+1}) & \text{on } \Gamma, \\
\mathcal{L}_3 v_i^{n+1} & = & \mathcal{L}_3 v_i^n + \dfrac{1}{2}\left(\mathcal{L}_3 \tilde{v}_1^{n+1} + \mathcal{L}_3 \tilde{v}_2^{n+1}\right) & \text{on } \Gamma, \\
v_i^{n+1} & = & v_i^n + \dfrac{1}{2}(\tilde{v}_1^{n+1} + \tilde{v}_2^{n+1}) & \text{on } \Gamma.
\end{array}
\right.
\tag{5.5}
$$

Note that the algorithm decouples completely into two algorithms. One is defined for $v_i^n$ and $\tilde{v}_i^n$. The other one is defined for $\gamma_i^n$ and $\tilde{\gamma}_i^n$.

*Proof.* We only give the proof for $\Omega_1$. The proof of the iterations in $\Omega_2$ is similar. We start with the updating step. The last interface condition of (5.5) is a direct consequence of (5.3). We consider now the second interface condition of (5.3). Using the incompressibility constraint $(\partial_x u_i^{n+1} = -\partial_y v_i^{n+1} - \partial_z w_i^{n+1}$, $i = 1, 2)$ yields

$$
\begin{array}{l}
-\nu \dfrac{\partial}{\partial y} v_1^{n+1} - \nu \dfrac{\partial}{\partial z} w_1^{n+1} - p_1^{n+1} = -\nu \dfrac{\partial}{\partial y} v_1^n - \nu \dfrac{\partial}{\partial z} w_1^n - p_1^n \\
\qquad + \dfrac{1}{2}\left( -\nu \dfrac{\partial}{\partial y} \tilde{v}_1^{n+1} - \nu \dfrac{\partial}{\partial z} \tilde{w}_1^{n+1} - \tilde{p}_1^{n+1} \right) \\
-\dfrac{1}{2}\left( \nu \dfrac{\partial}{\partial y} \tilde{v}_2^{n+1} + \nu \dfrac{\partial}{\partial z} \tilde{w}_2^{n+1} + \tilde{p}_2^{n+1} \right).
\end{array}
\tag{5.6}
$$

Differentiating the first component of the first interface condition of (5.3) with respect to $y$ and the second component with respect to $z$, multiplying with $\nu$ and adding to (5.6) yield

$$
p_1^{n+1} = p_1^n + \frac{1}{2}\left(\tilde{p}_1^{n+1} - \tilde{p}_2^{n+1}\right).
$$

Now we differentiate with respect to $y$ and use the Stokes equations. We obtain exactly the second interface condition of (5.5):

$$\mathcal{L}_3 v_1^{n+1} = \mathcal{L}_3 v_1^n + \frac{1}{2}\left(\mathcal{L}_3 \tilde{v}_1^{n+1} + \mathcal{L}_3 \tilde{v}_2^{n+1}\right).$$

In order to derive the first interface condition of (5.5), we differentiate the second component of the first interface condition of (5.3) with respect to $y$ and the first component with respect to $z$. Subtracting both equations yields

$$-\partial_z v_1^{n+1} + \partial_y w_1^{n+1} = -\partial_z v_1^n - \frac{1}{2}\left(\partial_z \tilde{v}_1^{n+1} + \partial_z \tilde{v}_2^{n+1}\right) + \partial_y w_1^n + \frac{1}{2}\left(\partial_y \tilde{w}_1^{n+1} + \partial_y \tilde{w}_2^{n+1}\right)$$

on $\Gamma$ or, using the definitions for $\gamma_i^n, \tilde{\gamma}_i^n$ in (5.3),

$$\gamma_i^{n+1} = \gamma_i^n + \frac{1}{2}(\tilde{\gamma}_1^{n+1} + \tilde{\gamma}_2^{n+1}) \quad \text{on } \Gamma,$$

which is exactly the first interface condition of (5.5).

Next, we will prove the equivalence of the correction step for the two algorithms. By differentiating the second component of the first interface condition of (5.2) with respect to $y$ we obtain

$$\nu \partial_{xy} \tilde{w}_1^{n+1} = -\frac{1}{2}\nu(\partial_{xy} w_1^n - \partial_{xy} w_2^n) \quad \text{on } \Gamma.$$

Differentiating the first component of the first equation of (5.2) with respect to $z$ and subtracting it from the previous equation we get

$$\nu \partial_{xy} \tilde{w}_1^{n+1} - \nu \partial_{xz} \tilde{v}_1^{n+1} = -\nu \frac{1}{2}(\partial_{xy} w_1^n - \partial_{xy} w_2^n) + \nu \frac{1}{2}(\partial_{xz} v_1^n - \partial_{xz} v_2^n).$$

Using the definition (5.3) of $\gamma_i^n$ and $\tilde{\gamma}_i^n$ we obtain the first interface condition of (5.4).

Finally, we have to derive the second interface condition of (5.4). We start with the first interface condition of (5.2). Differentiating the second boundary condition of (5.2) w.r.t. $y$ and $z$, multiplying by $\nu$ and adding the results to the first interface condition of (5.2), we get

$$\nu \frac{\partial \tilde{\vec{u}}_{i,\vec{\tau}_i}^{\,n+1}}{\partial \vec{n}_i} = -\frac{1}{2}\left(\nu \frac{\partial \vec{u}_{i,\vec{\tau}_i}^n}{\partial \vec{n}_i} + \nu \frac{\partial \vec{u}_{j,\vec{\tau}_j}^n}{\partial \vec{n}_j}\right) \quad \text{on } \Gamma.$$

Differentiating the first component with respect to $y$ and the second one with respect to $z$ we obtain

$$\nu \partial_{xy} \tilde{v}_1^{n+1} + \nu \partial_{xz} \tilde{w}_1^{n+1} = -\frac{1}{2}\nu\left(\partial_{xy}(v_1^n - v_2^n)\right) - \frac{1}{2}\nu\left(\partial_{xz}(w_1^n - w_2^n)\right).$$

Next we insert the incompressibility condition:

$$-\nu \partial_{xx} \tilde{u}_1^{n+1} = \frac{1}{2}\nu \partial_{xx}(u_1^n - u_2^n). \tag{5.7}$$

Differentiating the second interface condition of (5.2) in tangential directions yields

$$(-\nu\partial_{yy} - \nu\partial_{zz} + c)\tilde{u}_1^{n+1} = -\frac{1}{2}(-\nu\partial_{yy} - \nu\partial_{zz} + c)(u_1^n - u_2^n).$$

Now we add equation (5.7). We get

$$\mathcal{L}_3 \tilde{u}_1^{n+1} = -\frac{1}{2}(\mathcal{L}_3 u_1^n - \mathcal{L}_3 u_2^n).$$

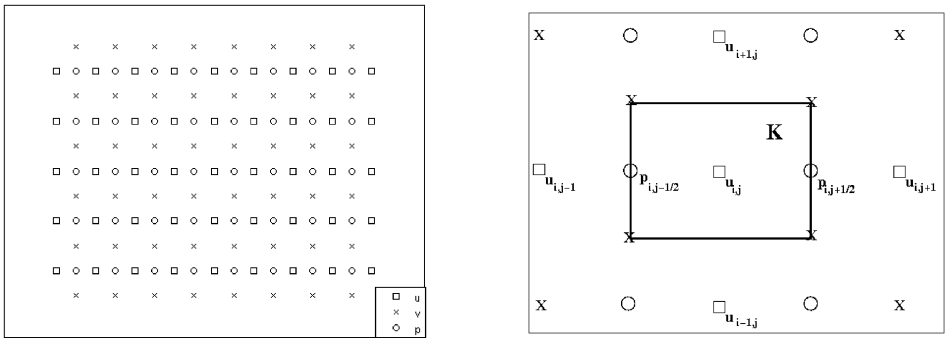


FIGURE 1. (a) Staggered grid, (b) a cell $K$ corresponding to the first velocity component $u$.

We use the Stokes equations and differentiate with respect to $y$:

$$\partial_y(\partial_x \tilde{p}_1^{n+1}) = -\frac{1}{2}\left(\partial_y(\partial_x p_1^n) - \partial_y(\partial_x p_2^n)\right).$$

Applying again the Stokes equations, we end up with the second interface condition of (5.4):

$$\partial_x \mathcal{L}_3 \tilde{v}_1^{n+1} = -\frac{1}{2}\nu(\partial_x \mathcal{L}_3 v_1^n - \partial_x \mathcal{L}_3 v_2^n).$$

Thus, everything is shown. □

*Remark* 5.3. The algorithm decouples into two scalar problems. Since one knows that each of these scalar algorithms converges in at most two steps, we obtain convergence in two steps for the three-dimensional case, too.

*Remark* 5.4. The new algorithm for the Stokes system is reminiscent of the hybrid approach presented in [10]. Indeed, in both cases, the interface conditions are mixed Dirichlet and Neumann type boundary conditions. But, our approach is different in two ways:

- It shows what is the good combination of stress and displacement for the interface conditions in both 2D and 3D.
- In the non-symmetric case, the complex interface condition (4.4) is not of the hybrid type as defined in [10].

## 6. Discretization

For the discretization of the two-dimensional case we choose a second order centered Finite Volume approach with a staggered grid (cf. [21]). In our numerical experiments we only consider the case, where the domain $\Omega$ is given by rectangles using regular grids. In Figure 1(a) a standard staggered grid for velocity $(u, v)$ and pressure $p$ is plotted. Each equation of the Stokes system is discretized by different control cells. In Figure 1(b) you see a typical interior control cell for the first equation. Let us study the discretization in more detail. We consider the first equation of the Stokes system for $(u, v, p)$ and integrate it over a cell $K_{ij}$ with

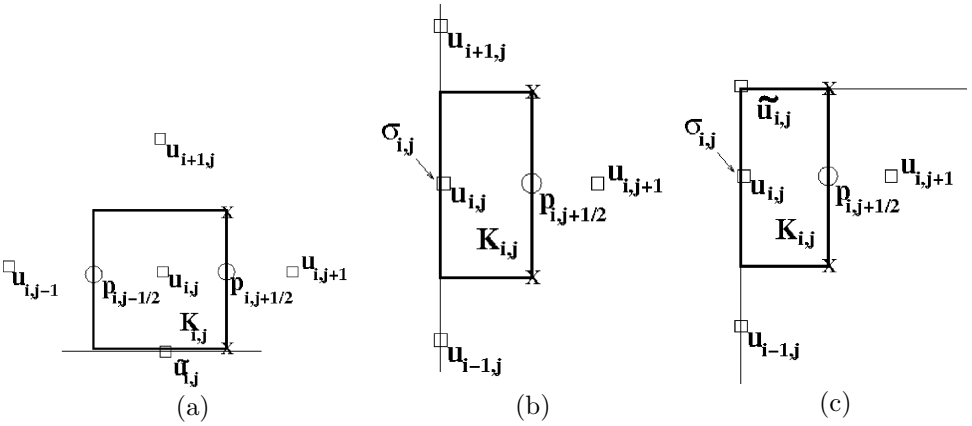


FIGURE 2. Boundary cells for $u$: (a) horizontal boundary cell, (b) vertical boundary cell, (c) corner cell.

center $x_{i,j}$ (position of $u_{i,j}$). Using integration by parts we obtain

$$\begin{aligned}\int_{K_{ij}} f_1 dx &= \int_{K_{ij}} \left(-\nu\Delta u + cu + \partial_x p\right) dx \\ &= \int_{\partial K_{ij}} \left(-\nu\partial_{\vec{n}_{K_{ij}}} u + pn_{ij,1}\right) ds + \int_{K_{ij}} cu dx\end{aligned}$$

where $n_{ij,k}$ is the $k$-th component of the outward normal $\vec{n}_{ij}$ of $K_{ij}$. Now this equation is discretized. We replace the derivatives of $u$ by corresponding central differences and approximate the remaining integrals by the midpoint rule. For the pressure we assume that it is constant along the edges. We denote the length of an interior cell $K_{ij}$ in $x$-direction by $\Delta x$ and the length in $y$-direction by $\Delta y$.

For an interior cell $K_{ij}$ we obtain the following equation:

$$\begin{aligned}\Delta x\Delta y\ f(x_{i,j}) = \Delta x\Delta y\ cu_{i,j} &+ \Delta y(-p_{i,j-1/2} + p_{i,j+1/2}) \\ &+ \frac{\Delta y}{\Delta x}\ \nu(2u_{i,j} - u_{i,j+1} - u_{i,j-1}) \\ &+ \frac{\Delta x}{\Delta y}\ \nu(2u_{i,j} - u_{i-1,j} - u_{i+1,j}).\end{aligned} \tag{6.1}$$

The different cells at the boundary are plotted in Figure 2. One has to distinguish between cells connected to horizontal boundaries or vertical boundaries and corner cells. Let us start with the cells that are connected to the horizontal boundaries. In the new domain decomposition method there are interface conditions for the normal stress. Since the normal stress on a boundary edge cannot be computed directly, we have to introduce an artificial value $\tilde{u}_{i,j}$. Then, the stress on the horizontal boundary can be approximated by $\nu\frac{\tilde{u}_{i,j}-u_{i,j}}{\Delta y/2}$. Therefore, we obtain for the cell in Figure 2(a) the following modification of equation (6.1):

$$\begin{aligned}&\Delta x\Delta y f(x_{i,j}) = \Delta x\Delta y\ cu_{i,j} + \Delta y(-p_{i,j-1/2} + p_{i,j+1/2}) \\ &+\frac{\Delta y}{\Delta x}\ \nu(2u_{i,j} - u_{i,j+1} - u_{i,j-1}) + \frac{\Delta x}{\Delta y}\ \nu(u_{i,j} - u_{i+1,j}) + \frac{\Delta x}{\Delta y/2}\ \nu(u_{i,j} - \tilde{u}_{i,j}).\end{aligned}$$

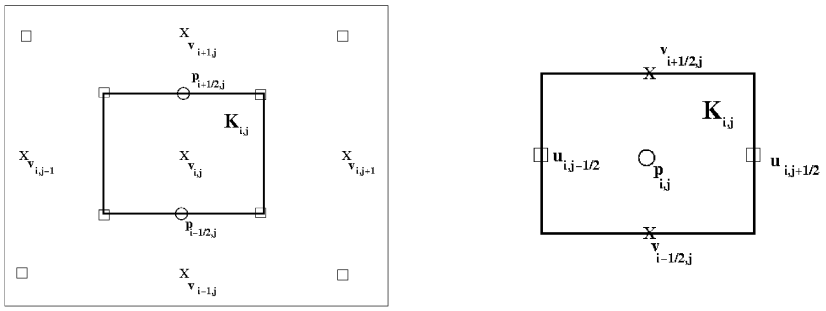


FIGURE 3. (a) Interior cell for the second velocity component $v$, (b) cell for the pressure $p$.

Next we consider a vertical boundary cell. Now, the cell $K_{ij}$ is given by a half cell; cf. Figure 2(b). We introduce on the boundary an artificial unknown $\sigma_{i,j}$ for the normal stress. Then the discretization is given by

$$\frac{\Delta x}{2}\Delta y f(x_{i,j}) = \frac{\Delta x}{2}\Delta y\ cu_{i,j} + \Delta y\ p_{i,j+1/2}$$
$$+\frac{\Delta y}{\Delta x}\ \nu(u_{i,j} - u_{i,j+1}) + \Delta y\ \sigma_{i,j} + \frac{\Delta x/2}{\Delta y}\ \nu(2u_{i,j} - u_{i-1,j} - u_{i+1,j}).$$

The corner cells are the combination of horizontal and vertical cells; cf. Figure 2 (c):

$$\frac{\Delta x}{2}\Delta y f(x_{i,j}) = \frac{\Delta x}{2}\Delta y\ cu_{i,j} + \Delta y\ p_{i,j+1/2}$$
$$+\frac{\Delta y}{\Delta x}\ \nu(u_{i,j} - u_{i,j+1}) + \Delta y\ \sigma_{i,j} + \frac{\Delta x/2}{\Delta y}\ \nu(u_{i,j} - u_{i-1,j}) + \frac{\Delta x/2}{\Delta y/2}\ \nu(u_{i,j} - \tilde{u}_{i,j}).$$

Thus, for each cell of $u$ we obtain one equation.

For the equation of the second velocity component $v$ we proceed in a similar manner. The center of the cells for $v$ are always given by the second velocity component. In Figure 3(a) an interior cell is plotted and in Figure 4 you can see, how the boundary cells can be treated.

The third equation is discretized with the help of the pressure nodes. Considering the cells centered by the pressure nodes, we observe that all cells can be handled in the same way; cf. Figure 3(b). Integrating over an arbitrary cell $K_{ij}$ yields

$$0 = \int_{\partial K_{ij}} un_{ij,1} + vn_{ij,2} ds,$$

where $n_{ij,k}$ is the $k$-th component of the outward normal $\vec{n}_{ij}$ of $K_{ij}$. Thus, the discretization is given by

$$0 = \Delta y(-u_{i,j-1/2} + u_{i,j+1/2}) + \Delta x(-v_{i-1/2,j} + v_{i+1/2,j}).$$

*Remark* 6.1. In the correction step the pressure is only determined up to a constant. In order to avoid singular problems, we regularize the pressure equation by

$$0 = \Delta y(-u_{i,j-1/2} + u_{i,j+1/2}) + \Delta x(-v_{i-1/2,j} + v_{i+1/2,j}) + \epsilon p_{i,j}$$

using a small value $\epsilon > 0$. In the numerical experiments we have chosen $\epsilon = 10^{-3}$.

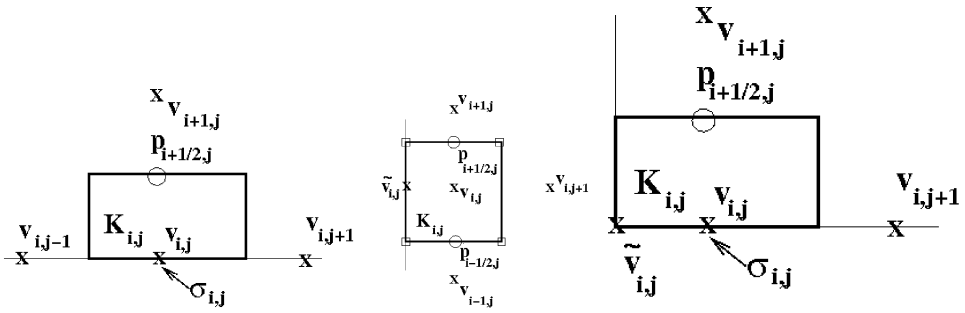


FIGURE 4. Boundary cells for $v$: (a) horizontal boundary cell, (b) vertical boundary cell, (c) corner cell.

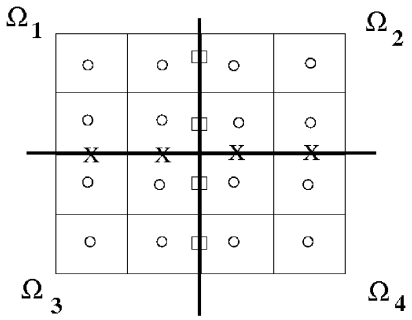


FIGURE 5. A $2 \times 2$ decomposition with pressure cells.

Finally, we discuss, how boundary conditions are imposed. Again, we restrict ourselves to the case of the first velocity component $u$. The boundary conditions for $v$ are imposed analogously. On vertical boundaries Dirichlet conditions, respectively, Neumann conditions are imposed by simply setting the nodes for $u$, respectively, $c\sigma_{i,j}$ on the interface. For horizontal boundaries Dirichlet conditions are imposed by setting the artificial values $\tilde{u}_{i,j}$. A Neumann condition $\nu\partial_{\vec{n}}u = g$ is discretized by setting

$$g(x_{i,j}) = \nu\frac{\partial u}{\partial \vec{n}}(x_{i,j}) \approx \nu\frac{\tilde{u}_{i,j} - u_{i,j}}{\Delta y/2} \tag{6.2}$$

for all nodes $x_{i,j}$ corresponding to the artificial unknowns $\tilde{u}_{i,j}$ (cf. Figure 2(b)).

For the domain decomposition we split the global rectangle $\Omega$ into local rectangles $\Omega_i$ in such a way that we retrieve local subdomains with the above pattern. This means that the subdomains consist of the union of cells of the pressure nodes. In Figure 5 an example for a $2 \times 2$ decomposition is shown. For the implementation of the domain decomposition algorithm a discretization of the interface conditions is needed. Fortunately, all interface conditions are of Dirichlet- or Neumann-type. For the sake of simplicity only the case of vertical interfaces is described. For horizontal interfaces the role of the first and the second velocity component has to

be switched. Thus, in the correction step (3.14) a Dirichlet condition

$$\tilde{u}_i^{n+1} = -\frac{1}{2}(u_i^n - u_j^n) \tag{6.3}$$

for the first velocity component $u$ and a Neumann condition

$$\sigma^i_{\vec{\tau}_i}(\vec{\tilde{u}}_i^{n+1}, \tilde{p}_i^{n+1}) = -\frac{1}{2}(\sigma^i_{\vec{\tau}_i}(\vec{\tilde{u}}_i^{n}, p_i^{n}) + \sigma^j_{\vec{\tau}_j}(\vec{u}_j^{n}, \tilde{p}_j^{n})) \tag{6.4}$$

for the second component $v$ has to be imposed in subdomain $\Omega_i$. Neumann conditions can be imposed following the line of (6.2), where normal derivatives of the right hand side of (6.4) can be computed by finite differences. For the Dirichlet conditions we just set the values on the interface to the corresponding value using the interface Dirichlet data of adjacent subdomains.

In the update step (3.15) we have a Neumann condition for the first velocity component and a Dirichlet condition for the second component. Imposing the Neumann condition for the first component is simple. One just sets the artificial stress $\sigma_{i,j}$ on the interface to the given value using the artificial stresses on the interface of the correction step. For the Dirichlet condition the artificial unknowns of the second velocity component on the interface are used.

We consider two different types of domain decomposition methods. First, we apply directly the discrete version of Algorithm 3.7. In the second version we have accelerated the algorithm using a Krylov method. Due to the non-symmetric structure of the boundary conditions we apply the GMRES method [25].

## 7. Numerical results

In this section we will analyze the performance of the new algorithm. It will be compared with the standard Schur complement approach using a Neumann-Neumann preconditioner (without coarse space); cf. [26]. We will extend the preliminary results of [4], where we made some numerical experiments for the two subdomains case, using standard inf-sup stable $P_2/P_1$-Taylor-Hood elements on triangles.

We consider the domain $\Omega = [0.2, 1.2] \times [0.1, 1.1]$ decomposed into two or more subdomains of equal or different sizes. We choose the right hand side $\vec{f}$ such that the exact solution is given by $u(x,y) = \sin(\pi x)^3 \sin(\pi y)^2 \cos(\pi y)$, $v(x,y) = -\sin(\pi x)^2 \sin(\pi y)^3 \cos(\pi x)$ and $p(x,y) = x^2 + y^2$. The viscosity $\nu$ is always 1. We solve the problem for various values of the reaction coefficient $c$, which can arise for example, when one applies an implicit time discretization of the unsteady Stokes problem ($c = 1/\Delta t$).

First, the interface system is solved by a purely iterative method (denoted respectively by $it_{New}$ and $it_{NN}$ for the new algorithm and the Neumann-Neumann preconditioner) and then accelerated by GMRES (denoted respectively by $ac_{New}$ and $ac_{NN}$ for the new algorithm and the Neumann-Neumann preconditioner). In all tables we count the number of iterations needed to reduce the $L^\infty$ norm of the error by the factor $TOL = 10^{-6}$:

$$\max_{i=1,\dots,N} \|u_k^i - u_h\|_{L^\infty(\Omega_i)} \leq 10^{-6},$$

where $u_k^i$ is the discrete solution of iteration step $k$ in subdomain $\Omega_i$ and $u_h$ is the global discrete solution computed by a direct solver applied to the global problem.

TABLE 1. Two subdomains case: (a) Influence of the reaction parameter on the convergence ($h = 1/96$). (b) Influence of the mesh size for $c = 10^{-5}$.

| $c$ | $it_{New}$ | $it_{NN}$ | $ac_{New}$ | $ac_{NN}$ |
|---|---|---|---|---|
| $10^{2}$ | 2 | 15 | 1 | 6 |
| $10^{0}$ | 2 | 15 | 1 | 6 |
| $10^{-3}$ | 2 | 15 | 1 | 6 |
| $10^{-5}$ | 2 | 15 | 1 | 6 |

| $h$ | $it_{New}$ | $it_{NN}$ | $ac_{New}$ | $ac_{NN}$ |
|---|---|---|---|---|
| 1/24 | 2 | 14 | 1 | 6 |
| 1/48 | 2 | 15 | 1 | 6 |
| 1/96 | 2 | 15 | 1 | 6 |

TABLE 2. Two subdomains case: Influence of the length of the first domain: (a) $c = 10^{-5}$, $h = 1/100$, (b) $c = 1.0$, $h = 1/100$.

| $L_1/L$ | $it_{New}$ | $it_{NN}$ | $ac_{New}$ | $ac_{NN}$ |
|---|---|---|---|---|
| 0.1 | - | - | 7 | 8 |
| 0.2 | 22 | 22 | 5 | 7 |
| 0.3 | 5 | 16 | 3 | 6 |
| 0.4 | 5 | 15 | 3 | 6 |
| 0.5 | 2 | 15 | 1 | 6 |

| $L_1/L$ | $it_{New}$ | $it_{NN}$ | $ac_{New}$ | $ac_{NN}$ |
|---|---|---|---|---|
| 0.1 | - | - | 7 | 8 |
| 0.2 | 15 | 18 | 5 | 7 |
| 0.3 | 5 | 16 | 3 | 6 |
| 0.4 | 5 | 15 | 3 | 7 |
| 0.5 | 2 | 15 | 1 | 6 |

Since the interface problem is ill-conditioned especially in the presence of cross-points, the reduction of the Euclidean norm of the residual is not a good indicator for the convergence of the algorithm. The case where the algorithm does not converge within 100 steps is denoted by $-$.

7.1. **Two subdomains case.** We first consider a decomposition into two subdomains of same width and study the influence of the reaction parameter and of the mesh size on the convergence. We can see in Table 1(a) that the convergence of the new algorithm is optimal. For the iterative version convergence is reached in two iterations. Since in this case the preconditioned operator for the corresponding Krylov method reduces in theory to the identity, the Krylov method converges in one step. This is also valid numerically. Moreover, both algorithms are completely insensitive with respect to the reaction parameter. The advantage in comparison to the Neumann-Neumann algorithm is obvious.

In Table 1(b) we fix the reaction parameter $c = 10^{-5}$ and vary the mesh size. The conclusions are similar: both algorithms converge independently of the mesh size and, again, we observe a clearly better convergence behavior of the new algorithm. The same kind of results are valid for different values of $c$ (not presented here).

Next, we consider a decomposition into two subdomains where the first subdomain is thinner than the second one. We study the influence of the ratio between the length $L_1$ of the first subdomain and the global domain $L$ for three values of $c$ (see Tables 2, 3).

We observe that the iterative counterparts of the algorithms are very sensitive to the size of the first subdomain (it might not even converge when the parameter $c$ is very small), but as expected not the accelerated one. Second, when the parameter $c$ is sufficiently large (which corresponds to small time steps when using a time

TABLE 3. Influence of the length of the first subdomain ($c = 10^2$, $h = 1/100$).

| $L_1/L$ | $it_{New}$ | $it_{NN}$ | $ac_{New}$ | $ac_{NN}$ |
|---|---|---|---|---|
| 0.1 | 16 | 40 | 5 | 7 |
| 0.2 | 7 | 16 | 3 | 6 |
| 0.3 | 5 | 15 | 3 | 6 |
| 0.4 | 4 | 16 | 2 | 7 |
| 0.5 | 2 | 15 | 1 | 6 |

TABLE 4. Influence of the number of subdomains for $h = 1/96$; $c = 10^{-5}$ (left), $c = 1$ (right).

| N | $it_{New}$ | $it_{NN}$ | $ac_{New}$ | $ac_{NN}$ |
|---|---|---|---|---|
| 2 | 2 | 15 | 1 | 6 |
| 4 | - | - | 8 | - |
| 6 | - | - | 15 | - |
| 8 | - | - | 21 | - |

| N | $it_{New}$ | $it_{NN}$ | $ac_{New}$ | $ac_{NN}$ |
|---|---|---|---|---|
| 2 | 2 | 15 | 1 | 6 |
| 4 | - | - | 7 | 13 |
| 6 | - | - | 12 | 25 |
| 8 | - | - | 17 | 31 |

TABLE 5. Influence of the number of subdomains ($c = 10^2$, $h = 1/96$).

| N | $it_{New}$ | $it_{NN}$ | $ac_{New}$ | $ac_{NN}$ |
|---|---|---|---|---|
| 2 | 2 | 15 | 1 | 6 |
| 4 | 34 | - | 5 | 9 |
| 6 | - | - | 7 | 15 |
| 8 | - | - | 10 | 21 |

discretization scheme), or of order 1, we have only small variations of iteration numbers in the case of thinner subdomains.

7.2. **Multi-domain case.** Now we analyze the case of a decomposition into more than two subdomains. Two cases are considered: strip-wise decompositions (with subdomains of the same size or with a variable length, see Tables 4, 6) and more general decompositions with cross points.

7.2.1. *Strip-wise decomposition.* First of all we fix the mesh size $h = 1/96$ and for different values of $c$ we vary the number of subdomains. In the case of a strip-wise decomposition into $N$ subdomains, due to the ill-conditioning of the Neumann-Neumann preconditioner this algorithm does not reach the given tolerance of $10^{-6}$. A suitable coarse space will cure this. These results show clearly that the new algorithm is more robust w.r.t. the absence of a coarse space. For large $c$ (cf. Table 5) the behavior of the two domains case is conserved. Using the new algorithm, the number of iteration steps is almost reduced by a factor of two.

Next, we consider a $4 \times 1$ strip-wise decomposition into subdomains of variable length (here $[m_1,\ m_2,\ m_3,\ m_4]$ denotes the number of discretization points in $x$-direction per subdomain). Again, we can conclude, that the new algorithm shows clearly better results.

TABLE 6. Influence of the size of subdomains ($h = 1/96$).

| c | N | $it_{New}$ | $it_{NN}$ | $ac_{New}$ | $ac_{NN}$ |
|---|---|---|---|---|---|
| $10^{-5}$ | [16, 32, 16, 32] | - | - | 9 | - |
| | [16, 48, 16, 16] | - | - | 10 | - |
| | [48, 16, 16, 16] | - | - | 12 | - |
| $10^{0}$ | [16, 32, 16, 32] | - | - | 8 | 14 |
| | [16, 48, 16, 16] | - | - | 10 | 13 |
| | [48, 16, 16, 16] | - | - | 12 | 17 |
| $10^{2}$ | [16, 32, 16, 32] | 74 | - | 5 | 12 |
| | [16, 48, 16, 16] | - | - | 6 | 11 |
| | [48, 16, 16, 16] | - | - | 6 | 14 |

TABLE 7. Influence of the number of subdomains: $h = 1/96$.

| c | $N \times N$ | $it_{New}$ | $it_{NN}$ | $ac_{New}$ | $ac_{NN}$ |
|---|---|---|---|---|---|
| $10^{-5}$ | 2x2 | - | - | 9 | 13 |
| | 3x3 | - | - | 28 | - |
| | 4x4 | - | - | 40 | - |
| $10^{0}$ | 2x2 | - | - | 9 | 13 |
| | 3x3 | - | - | 30 | 28 |
| | 4x4 | - | - | 39 | 39 |
| $10^{2}$ | 2x2 | 61 | - | 7 | 11 |
| | 3x3 | - | - | 22 | 21 |
| | 4x4 | - | - | 27 | 27 |

7.2.2. *General decomposition.* The final test cases treat general decompositions into $N \times N$ subdomains. Two values for the reaction coefficient $c$ are analyzed.

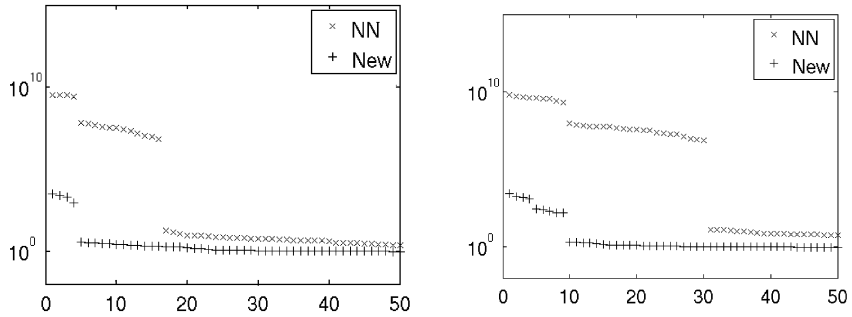


FIGURE 6. Singular values of the preconditioned interface operator for a $4 \times 4$ decomposition (left), $5 \times 5$ decomposition (right).

As seen in Tables 4, 6, 7, contrarily to the new algorithm, the Neumann-Neumann algorithm fails to converge when $c$ is small. Actually, the residual of the GMRES iterates decreases but not the error in $L^{\infty}$-norm with respect to the solution computed by a direct method. This is due to the fact that for small $c$ the preconditioned problem is ill-conditioned. In Figure 6 we plot the largest 50 singular values

TABLE 8. Number of singular values which are larger than 10 for a $N \times N$ decomposition.

| $N \times N$ | NN | New |
|---|---|---|
| 2x2 | 4 | 0 |
| 3x3 | 9 | 1 |
| 4x4 | 20 | 4 |
| 5x5 | 36 | 9 |
| 6x6 | 54 | 16 |
| 7x7 | 76 | 25 |
| 8x8 | 102 | 36 |
| 9x9 | 135 | 49 |
| 10x10 | 167 | 64 |

of the singular value decomposition (SVD) of the upper Hessenberg matrix $H$ computed by the first 200 GMRES iterations for both methods. We see that for the Neumann-Neumann algorithm the largest eigenvalues are much higher than for the new algorithm. A possible cure is adding a suitable coarse space whose size is at least equal to the number of largest SVD eigenvalues. We see in Table 8 that the coarse space for the Neumann-Neumann algorithm must be much larger. For the new algorithm one degree of freedom for each inner sudomain seems to be sufficient.

## 8. Conclusion

In this paper we have shown that the Smith factorization is a powerful tool in order to derive new domain decomposition methods for vector valued partial differential equations. Compared to the classical algorithm, the new algorithm is much more robust w.r.t. small reaction terms which correspond to using a large time step in an implicit scheme for the unsteady Stokes equations. Of course, the convergence of both methods is not completely satisfactory in the multi-domain case with cross points. But the number of needed iteration steps can be dramatically decreased by using an appropriate coarse space. A suitable choice of a coarse space for our new approach is a subject of further research. An experimental convergence analysis shows that the size of the coarse space for the new algorithm will be much smaller than the one needed by the classical algorithm; see also [10].

Moreover, we outlined, how this approach can be used in order to derive a domain decomposition method for the Oseen equations. We expect that the proposed algorithm will be robust with respect to the viscosity $\nu$. To our knowledge this would be the first one showing this behavior. The power of the Smith factorization can also be seen from the fact that one of the authors used it in order to design perfectly matched layers (PML) for the compressible Euler equations (cf. [17]).

## 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*, 184:145–170, 2000. MR1764189 (2002a:76110)

[2] M. Ainsworth and S. Sherwin. Domain decomposition preconditioners for $p$ and $hp$ finite element approximations of Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 175:243–266, 1999. MR1702213 (2000h:76116)

[3] V. Dolean and F. Nataf. A new domain decomposition method for the compressible Euler equations. *M2AN Math. Model. Numer. Anal.*, 40:689–703, 2006. MR2274774 (2007g:76155)
[4] V. Dolean, F. Nataf, and G. Rapin. New constructions of domain decomposition methods for systems of PDEs. *C.R. Acad. Sci. Paris, Ser I*, 340:693–696, 2005. MR2139279
[5] V. Dolean, F. Nataf, and G. Rapin. A New Domain Decomposition Method for the Oseen Equations, 2006. In Preperation.
[6] Ch. Farhat and F.-X. Roux. A method of finite element tearing and interconnecting and its parallel solution algorithm. *Internat. J. Numer. Methods Engrg.*, 32:1205–1227, 1991.
[7] L. Gerardo-Giorda, P. Le Tallec, and F. Nataf. A Robin-Robin preconditioner for advection-diffusion equations with discontinuous coefficients. *Comput. Methods Appl. Mech. Engrg.*, 193:745–764, 2004. MR2037041 (2004k:65203)
[8] V. Girault and P.A. Raviart. *Finite Element Methods for Navier-Stokes Equations*. Springer, Heidelberg-Berlin, 1986. MR851383 (88b:65129)
[9] R. Glowinski, Y.A. Kuznetsov, G. Meurant, J. Periaux, and O.B. Widlund, editors. *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, Philadelphia, 1991. SIAM. MR1106444 (92a:65023)
[10] P. Gosselet and Chr. Rey. Non-overlapping domain decomposition methods in structural mechanics. *Arch. Comput. Methods Engrg.*, 13(4):515–572, 2006. MR2303317
[11] J. Li. A dual-primal FETI method for incompressible Stokes equations. *Numer. Math.*, 102:257–275, 2005. MR2206465 (2007e:65123)
[12] J. Li and O. Widlund. BDDC algorithms for incompressible Stokes equations, 2006. submitted.
[13] J. Mandel. Balancing domain decomposition. *Comm. on Applied Numerical Methods*, 9:233–241, 1992. MR1208381 (94b:65158)
[14] J. Mandel and M. Brezina. Balancing domain decomposition: Theory and performance in two and three dimensions. UCD/CCM report 2, 1993.
[15] J. Mandel, C.R. Dohrmann, and R. Tezaur. An algebraic theory for primal and dual substructuring methods by constraints. *Appl. Numer. Math.*, 54:167–193, 2005. MR2148040 (2006a:65151)
[16] F. Nataf. Interface conditions for domain decomposition methods for 2D and 3D Oseen equations. *C. R. Acad. Sci., Paris*, Ser. I 324:1155–1160, 1997. MR1451940 (98c:76084)
[17] F. Nataf. A new construction of perfectly matched layers for the linearized Euler equations. *J. Computational Phys.*, 214:757–772, 2006. MR2216613 (2006k:76106)
[18] F. Nataf and G. Rapin. Construction of a New Domain Decomposition Method for the Stokes Equations. In O.B. Widlund and D.E. Keyes, editors, *Domain Decomposition Methods in Science and Engineering XVI*, pages 247–254. Springer, 2007. MR2334110
[19] F.-C. Otto and G. Lube. A nonoverlapping domain decomposition method for the Oseen equations. *Math. Models Methods Appl. Sci.*, 8:1091–1117, 1998. MR1646527 (99i:65102)
[20] F.-C. Otto, G. Lube, and L Müller. An iterative substructuring method for div-stable finite element approximations of the Oseen problem. *Computing*, 67:91–117, 2001. MR1867355 (2003a:65110)
[21] S.V. Patankar. *Numerical heat transfer and fluid flow*. MC Graw-Hill, New York, 1980.
[22] L.F. Pavarino and O.B. Widlund. Balancing Neumann-Neumann methods for incompressible stokes equations. *Comm. Pure Appl. Math.*, 55:302–335, 2002. MR1866366 (2002h:76048)
[23] Y.H. De Roeck and P. Le Tallec. Analysis and Test of a Local Domain Decomposition Preconditioner. In *R. Glowinski et al.* [9], 1991. MR1106455
[24] E. Ronquist. A domain decomposition solver for the steady Navier-Stokes equations. In A. Ilin and L. Scott, editors, *Proc. of ICOSAHOM.95*, pages 469–485. Houston Journal of Mathmatics, 1996.
[25] Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986. MR848568 (87g:65064)
[26] P . Le Tallec and A. Patra. Non-overlapping domain decomposition methods for adaptive *hp* approximations of the Stokes problem with discontinuous pressure fields. *Comput. Methods Appl. Mech. Engrg.*, 145:361–379, 1997. MR1456020 (98e:76067)
[27] P. Le Tallec, J. Mandel, and M. Vidrascu. A Neumann-Neumann domain decomposition algorithm for solving plate and shell problems. *SIAM J. Numer. Anal.*, 35:836–867, 1998. MR1618907 (99f:65192)

[28] A. Toselli and O. Widlund. *Domain Decomposition Methods—Algorithms and Theory*. Springer, Berlin-Heidelberg, 2005. MR2104179 (2005g:65006)
[29] J.T. Wloka, B. Rowley, and B. Lawruk. *Boundary Value Problems for Elliptic Systems*. Cambridge University Press, Cambridge, 1995. MR1343490 (96f:35003)

Laboratoire J.A. Dieudonné, CNRS UMR 6621, Université de Nice Sophia-Antipolis, 06108 Nice Cedex 02, France
*E-mail address*: dolean@math.unice.fr

Laboratoire J.L. Lions, CNRS UMR 7598, Université Pierre et Marie Curie, 75252 Paris Cedex 05, France
*E-mail address*: nataf@ann.jussieu.fr

Department of Mathematics, NAM, University of Göttingen, D-37083, Germany
*E-mail address*: grapin@math.uni-goettingen.de