50 (2005) APPLICATIONS OF MATHEMATICS No. 3, 291–308

COMPLEXITY OF AN ALGORITHM FOR SOLVING SADDLE-POINT SYSTEMS WITH SINGULAR BLOCKS ARISING IN WAVELET-GALERKIN DISCRETIZATIONS*

Radek Kučera, Ostrava

Abstract. The paper deals with fast solving of large saddle-point systems arising in wavelet-Galerkin discretizations of separable elliptic PDEs. The periodized orthonormal compactly supported wavelets of the tensor product type together with the fictitious domain method are used. A special structure of matrices makes it possible to utilize the fast Fourier transform that determines the complexity of the algorithm. Numerical experiments confirm theoretical results.

Keywords: wavelet-Galerkin discretization, fictitious domain method, saddle-point system, conjugate gradient method, circulant matrix, fast Fourier transform, Kronecker product

MSC 2000: 65T60, 65F10, 65T50, 65N30

1. Introduction

In this paper we propose a fast method for finding a pair $(\mathbf{u},\boldsymbol{\lambda}) \in \mathbb{R}^n \times \mathbb{R}^m$ that solves the linear system of algebraic equations called the *saddle-point system*:

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^\top \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix}, \tag{1}$$

where the $(n\times n)$ matrix $\mathbf{A}$ is symmetric positive semi-definite, the $(m\times n)$ matrix $\mathbf{B}$ has full row-rank and the vectors $\mathbf{f}$, $\mathbf{g}$ are of the order n, m, respectively. We will be interested especially in systems (1) with n large, $\mathbf{A}$ singular, $\mathbf{B}$ sparse and m much smaller than n. Moreover, we will assume that the defect of $\mathbf{A}$, i.e. $l = n - \operatorname{rank}\mathbf{A}$, is much smaller than m. Systems of this type arise e.g. when we want to solve quadratic programming problems with equality constraints [3], mixed formulations

* This work was supported by grant HPRNT-CT-2002-00286 and MSM 272400019.

of second-order elliptic problems [1], or if we use the fictitious domain method to Dirichlet problems [5].

The paper has been inspired by a class of saddle-point systems arising in wavelet-Galerkin discretizations of separable elliptic PDEs. It is well known that the orthonormal compactly supported wavelets are defined on a bounded interval (or a rectangular domain) via periodization [4]. Therefore they are a natural tool for solving problems with periodic boundary conditions. Other types of boundary conditions (e.g. Dirichlet or Neumann) can be treated by means of the *fictitious domain method* [6], [9]. In these cases, it is necessary to solve the saddle-point system (1), where the diagonal block $\mathbf{A}$ represents the PDE and the off-diagonal block $\mathbf{B}$ describes the geometry of the domain. More precisely, $\mathbf{A}$ is the stiffness matrix on a new fictitious domain where the periodic boundary conditions are considered. On the one hand, it can happen that the matrix $\mathbf{A}$ is singular, on the other hand, the periodic boundary conditions together with separability of the PDE lead to the block circulant structure of $\mathbf{A}$. Since circulant matrices are diagonalizable by the *discrete Fourier transform* (DFT), one can evaluate eigenvalues of $\mathbf{A}$ efficiently by the *fast Fourier transform* (FFT). This makes the situation much easier because it is possible to treat the singularity of $\mathbf{A}$ without great computational costs.

There are several basic approaches used for solving the saddle-point systems (1). We turn our attention to the class of methods called *primary* (the Schur complement methods, the range space methods [8] or static condensation [2], [1]). The key idea is based on eliminating the first unknown $\mathbf{u}$. If $\mathbf{A}$ is non-singular, we obtain a linear system in terms of the second unknown $\boldsymbol{\lambda}$ with a positive definite matrix. Then it is natural to use the *conjugate gradient method* (CGM) for computation of the solution.

The situation is not so easy if $\mathbf{A}$ is singular because the first unknown $\mathbf{u}$ can not be eliminated completely from (1). We obtain a (second) linear system in terms of $\boldsymbol{\lambda}$ and a new unknown, say $\boldsymbol{\alpha}$, that represents the correspondence of $\mathbf{u}$ to the null-space of $\mathbf{A}$. The new linear system has again the saddle-point structure and its diagonal block is non-singular in many practical situations. Therefore we can repeatedly eliminate the first unknown, now it is $\boldsymbol{\lambda}$, and obtain a (third) linear system in terms of $\boldsymbol{\alpha}$ with a positive definite matrix. The resulting linear system can be solved easily, e.g. by a direct method thanks to the small order. Let us point out that the CGM can be utilized during the elimination process in order to compute the matrix and the right-hand side vector of the third linear system without necessity to compute and store the diagonal block of the second linear system. Although this idea seems to be cumbersome, we will show its efficient realization leading to an algorithm with very small memory requirements. In advance, we propose a fast implementation for the saddle-point systems arising in the wavelet-Galerkin discretizations of PDEs mentioned above.

The paper is organized as follows. In Section 2, we describe the wavelet-Galerkin discretization of a model PDEs problem. In Section 3, we summarize theoretical results concerning existence and uniqueness of the solution to the saddle-point systems (1) and the elimination of the first unknown in the case of singular **A**. A general scheme of the algorithm is proposed in Section 4. A fast implementation based on the use of the FFT and the Kronecker product is described in Section 5. Finally, Section 6 presents results of numerical experiments.

2. Wavelet-Galerkin discretization of model problem

Let ω be a bounded domain in $\mathbb{R}^2$ with a Lipschitzian boundary $\partial\omega$. We consider the following *Dirichlet problem*:

$$-\Delta u + cu = f \quad \text{in } \omega, \tag{2}$$

$$u = g \quad \text{on } \partial\omega, \tag{3}$$

where f, g are sufficiently smooth functions defined on ω, $\partial\omega$, respectively, and $c \geqslant 0$ is a given constant.

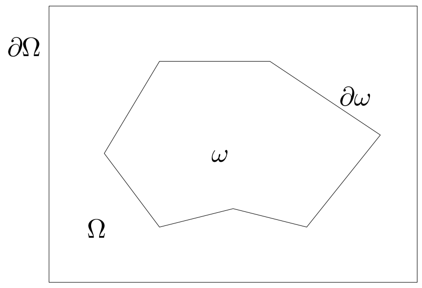

Figure 1. Fictitious domain.

We imbed ω in a larger rectangular domain Ω, i.e. $\overline{\omega} \subset \Omega$, with the boundary denoted by $\partial\Omega$; see Fig. 1. For the sake of simplicity, we shall assume that it is possible to take $\Omega = [0,1] \times [0,1]$. On Ω, we shall solve (2), (3) by means of the fictitious domain method with the boundary Lagrange multipliers; see [5]. To this end, we replace (2), (3) by the following *saddle-point problem*:

$$\begin{cases} \text{Find } (\tilde{u}, \lambda) \in V(\Omega) \times H^{-1/2}(\partial\omega) \quad \text{such that} \\ a_\Omega(\tilde{u}, v) = \displaystyle\int_\Omega \tilde{f} v \,\mathrm{d}x\,\mathrm{d}y + \langle \lambda, v \rangle \quad \forall\, v \in V(\Omega), \\ \langle \mu, \tilde{u} - g \rangle = 0 \quad \forall\, \mu \in H^{-1/2}(\partial\omega), \end{cases} \tag{4}$$

where $V(\Omega)$ is a well chosen subspace of $H^1(\Omega)$, $H^{-1/2}(\partial\omega)$ denotes the dual to $H^{1/2}(\partial\omega)$ with the duality pairing $\langle\cdot,\cdot\rangle$, $\tilde{f} \in L^2(\Omega)$ extends f from ω onto Ω and

$$a_\Omega(v,w) = \int_\Omega (\nabla v \cdot \nabla w + cvw)\,\mathrm{d}x\,\mathrm{d}y \quad \forall\, v,w \in V(\Omega).$$

It is well known that the saddle-point problem (4) has a unique solution $(\tilde{u},\lambda)$ and that $u = \tilde{u}_{|\omega}$ solves a weak formulation of (2), (3).

Let us specify $V(\Omega)$. Since we shall discretize the problem (4) by using periodized function spaces, it is natural to choose

$$V(\Omega) = H^1_{\text{per}}(\Omega), \tag{5}$$

i.e. $V(\Omega)$ is the subset of function from $H^1(\Omega)$ periodic at the boundary $\partial\Omega$.

Let V^J be the family of finite dimensional subspaces of $H^1_{\text{per}}(\Omega)$ defined by

$$V^J = \Big\{v\colon\ v = \sum_{k_x}\sum_{k_y} v_{k_x,k_y}\varphi^J_{k_x,k_y}(x,y),\ v_{k_x,k_y} = v_{k_x+n_x,k_y},\ v_{k_x,k_y} = v_{k_x,k_y+n_y}\Big\}$$

with $n := \dim V^J = n_x n_y$, where $J = (J_x, J_y)$ is a multiindex,

$$\varphi^J_{k_x,k_y}(x,y) = 2^{(J_x+J_y)/2}\varphi(2^{J_x}x - k_x)\varphi(2^{J_y}y - k_y)$$

are wavelet-scaling functions on the wavelet-levels J_x, J_y and $n_x = 2^{J_x}$, $n_y = 2^{J_y}$, respectively. We will assume that φ is an orthonormal compactly supported wavelet-scaling function with sufficiently high regularity; see [4].

Since the basis functions $\varphi^J_{k_x,k_y}(x,y)$ are of the tensor product type, their supports induce on Ω a dyadic rectangular mesh $\mathcal{D}^J$ consisting of rectangles

$$R_{k_x,k_y} = \big[2^{-J_x}(k_x-1), 2^{-J_x}k_x\big] \times \big[2^{-J_y}(k_y-1), 2^{-J_y}k_y\big].$$

We will restrict ourselves to the situations where the intersection between $\partial\Omega$ and any rectangle R_{k_x,k_y} is either empty or an edge. Then

$$\mathcal{D}^J = \{R_{k_x,k_y}\colon\ k_x = 1,\dots,n_x,\ k_y = 1,\dots,n_y\}.$$

Let $\{\Lambda^J\}$ be a family of finite dimensional spaces approximating $H^{-1/2}(\partial\omega)$. Before giving their definitions, we will approximate the boundary curve $\partial\omega$ by

$$[|\partial\omega|] = \bigcup_{R_{k_x,k_y}\in\mathcal{D}^J_{\partial\omega}} R_{k_x,k_y},$$

where $\mathcal{D}^J_{\partial\omega}$ is the subset of $\mathcal{D}^J$ defined by

$$\mathcal{D}^J_{\partial\omega} = \{R_{k_x,k_y} \in \mathcal{D}^J \colon\ R_{k_x,k_y} \cap \partial\omega \neq \emptyset\};$$

see Fig. 2. Let us now define

$$\Lambda^J = \{\mu\colon\ \mu_{|R_{k_x,k_y}} \in P_0(R_{k_x,k_y}) \quad \forall\, R_{k_x,k_y} \in \mathcal{D}^J_{\partial\omega}\},$$

i.e. Λ^J contains functions that are constant on any rectangle R_{k_x,k_y} intersected by the boundary curve $\partial\omega$. It is easily seen that $m := \dim \Lambda^J = \operatorname{card} \mathcal{D}^J_{\partial\omega}$.

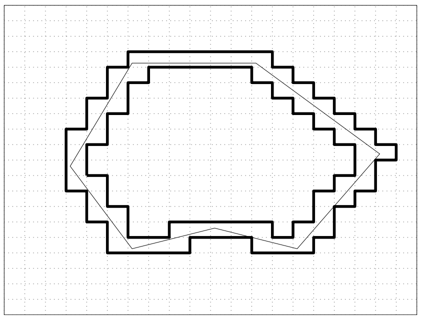

Figure 2. Approximation of the boundary $\partial\omega$ by $[|\partial\omega|]$.

If we replace $V(\Omega)$ by V^J and $H^{-1/2}(\partial\omega)$ by Λ^J in (4), we obtain the *wavelet-Galerkin discretization* of the problem (2), (3):

$$(6)\qquad \begin{cases} \text{Find } (u^J, \lambda^J) \in V^J \times \Lambda^J \quad \text{such that} \\ a_\Omega(u^J, v^J) = \displaystyle\int_\Omega \tilde{f} v^J \,\mathrm{d}x\,\mathrm{d}y + \langle\!\langle \lambda^J, v^J \rangle\!\rangle \quad \forall\, v^J \in V^J, \\ \langle\!\langle \mu^J, u^J - g^J \rangle\!\rangle = 0 \quad \forall\, \mu^J \in \Lambda^J, \end{cases}$$

where g^J is an appropriate extension of g from $\partial\omega$ to $[|\partial\omega|]$ and $\langle\!\langle \cdot,\cdot \rangle\!\rangle$ approximates the duality pairing $\langle \cdot,\cdot \rangle$ so that

$$\langle\!\langle \mu^J, v^J \rangle\!\rangle = \int_{[|\partial\omega|]} \mu^J v^J \,\mathrm{d}x\,\mathrm{d}y \quad \forall\, \mu^J \in \Lambda^J \ \ \forall\, v^J \in V^J.$$

If we rewrite (6) into the algebraic form, we obtain the saddle-point system (1). Its diagonal block $\mathbf{A}$ is singular provided $c = 0$ because it is the stiffness matrix of the Laplace operator on the domain Ω, where periodic boundary conditions are considered. Since V^J are represented by tensor product functions, the matrix $\mathbf{A}$ can be described by the Kronecker product as

$$(7)\qquad \mathbf{A} = \mathbf{A}_x \otimes \mathbf{I}_y + \mathbf{I}_x \otimes \mathbf{A}_y,$$

where $\mathbf{I}_x$, $\mathbf{I}_y$ and $\mathbf{A}_x$, $\mathbf{A}_y$ are of the order n_x, n_y, respectively. Moreover, $\mathbf{A}_x$, $\mathbf{A}_y$ are circulant matrices because of the presence of the periodic boundary condition on $\partial\Omega$. For more details about $\mathbf{A}$ and $\mathbf{B}$, we refer to [9].

3. Preliminaries

Let us denote the null-space and the range-space of $\mathbf{B}$ by

$$\mathcal{N}(\mathbf{B}) = \{\mathbf{v} \in \mathbb{R}^n : \ \mathbf{B}\mathbf{v} = \mathbf{o}\} \quad \text{and} \quad \mathcal{R}(\mathbf{B}) = \{\mu \in \mathbb{R}^m : \ \boldsymbol{\mu} = \mathbf{B}\mathbf{v}\},$$

respectively. The following statements are well known.

Lemma 1. *Let* $\mathbf{A}$ *be symmetric positive semi-definite. Then* $\mathbf{v} \in \mathcal{N}(\mathbf{A})$ *iff* $\mathbf{v}^\top \mathbf{A}\mathbf{v} = 0$.

Theorem 1. *The saddle-point system* (1) *has a unique solution iff*

$$\mathcal{N}(\mathbf{A}) \cap \mathcal{N}(\mathbf{B}) = \{\mathbf{o}\}. \tag{8}$$

We will assume that $\mathbf{A}$ in the saddle-point system (1) is singular with $l = \dim \mathcal{N}(\mathbf{A})$, $l \geqslant 1$. Consider an $(n \times l)$ matrix $\mathbf{N}$ whose columns span the null-space $\mathcal{N}(\mathbf{A})$ and denote by $\mathbf{A}^\dagger$ a generalized inverse to $\mathbf{A}$ that satisfies

$$\mathbf{A} = \mathbf{A}\mathbf{A}^\dagger\mathbf{A}. \tag{9}$$

Let us point out that $\mathbf{A}^\dagger$ is not determined by (9) uniquely. The following remark shows that we can easily find a symmetric positive semi-definite $\mathbf{A}^\dagger$.

R e m a r k 1. Any symmetric positive semi-definite matrix $\mathbf{A}$ can be factored into a product $\mathbf{L}\mathbf{D}\mathbf{L}^\top$ with a non-singular lower tri-diagonal $\mathbf{L}$ and a diagonal $\mathbf{D} = \operatorname{diag}(d_1, \ldots, d_n)$; see [7]. Let us define $\mathbf{D}^\dagger = \operatorname{diag}(d_1^\dagger, \ldots, d_n^\dagger)$, where $d_i^\dagger = 1/d_i$ if $d_i \neq 0$ and $d_i^\dagger = 0$ if $d_i = 0$. It may be verifed that $\mathbf{A}^\dagger = (\mathbf{L}^\top)^{-1}\mathbf{D}^\dagger\mathbf{L}^{-1}$ is symmetric positive semi-definite and satisfies (9).

Theorem 2. *Let us assume that* (8) *is satisfied and* $\mathbf{A}^\dagger$ *is symmetric positive semi-definite. The second component* $\boldsymbol{\lambda}$ *of the solution to* (1) *is the first component of the solution to the linear system*

$$\begin{pmatrix} \mathbf{B}\mathbf{A}^\dagger\mathbf{B}^\top & -\mathbf{B}\mathbf{N} \\ -\mathbf{N}^\top\mathbf{B}^\top & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} \mathbf{B}\mathbf{A}^\dagger\mathbf{f} - \mathbf{g} \\ -\mathbf{N}^\top\mathbf{f} \end{pmatrix}. \tag{10}$$

The first component $\mathbf{u}$ *of the solution to* (1) *is given by the formula*

$$\mathbf{u} = \mathbf{A}^\dagger(\mathbf{f} - \mathbf{B}^\top\boldsymbol{\lambda}) + \mathbf{N}\boldsymbol{\alpha}. \tag{11}$$

P r o o f. First we shall prove that there is a unique solution to (10). Under our assumptions, it is easy to show that $\mathbf{B}\mathbf{A}^\dagger\mathbf{B}^\top$ is symmetric positive semi-definite and $-\mathbf{N}^\top\mathbf{B}^\top$ has full row-rank. Therefore (10) is again a saddle-point system of the type (1). Let us have $\boldsymbol{\mu} \in \mathcal{N}(\mathbf{B}\mathbf{A}^\dagger\mathbf{B}^\top) \cap \mathcal{N}(-\mathbf{N}^\top\mathbf{B}^\top)$ and denote $\mathbf{v} = \mathbf{B}^\top\boldsymbol{\mu}$. Because of $\mathbf{N}^\top\mathbf{v} = \mathbf{o}$, $\mathbf{v}$ belongs to $\mathcal{R}(\mathbf{A})$ so that $\mathbf{v} = \mathbf{A}\mathbf{w}$. Furthermore, $\boldsymbol{\mu}^\top\mathbf{B}\mathbf{A}^\dagger\mathbf{B}^\top\boldsymbol{\mu} = 0$ yields

$$0 = \mathbf{v}^\top\mathbf{A}^\dagger\mathbf{v} = \mathbf{w}^\top\mathbf{A}\mathbf{A}^\dagger\mathbf{A}\mathbf{w} = \mathbf{w}^\top\mathbf{A}\mathbf{w}.$$

According to Lemma 1, we obtain $\mathbf{w} \in \mathcal{N}(\mathbf{A})$ so that $\mathbf{v} = \mathbf{o}$ or equivalently $\mathbf{B}^\top\boldsymbol{\mu} = \mathbf{o}$. The last equality gives $\boldsymbol{\mu} = \mathbf{o}$ because $\mathbf{B}$ has full-row rank. Hence $\mathcal{N}(\mathbf{B}\mathbf{A}^\dagger\mathbf{B}^\top) \cap \mathcal{N}(-\mathbf{N}^\top\mathbf{B}^\top) = \{\mathbf{o}\}$ and Theorem 1 implies that there is a unique solution to the saddle-point system (10). It remains to prove that the pair $(\mathbf{u}, \boldsymbol{\lambda})$ satisfying (10) and (11) is a solution to the saddle-point system (1). It may be directly verified by substituting (11) and then (10) into (1). □

The algorithm proposed in the next section is based on the previous theorem. We will confine ourselves to situations in which $\mathbf{B}\mathbf{A}^\dagger\mathbf{B}^\top$ is non-singular. A sufficient condition guaranteeing this property is proved in the following theorem.

Theorem 3. (i) *The matrix* $\mathbf{B}\mathbf{A}^\dagger\mathbf{B}^\top$ *in* (10) *is symmetric positive definite if*

$$\mathcal{N}(\mathbf{A}^\dagger) \cap \mathcal{R}(\mathbf{B}^\top) = \{\mathbf{o}\}. \tag{12}$$

(ii) *If* $\mathbf{A}^\dagger$ *is the Moore-Penrose pseudoinverse to* $\mathbf{A}$ *then* (12) *is equivalent to*

$$\mathcal{N}(\mathbf{A}) \cap \mathcal{R}(\mathbf{B}^\top) = \{\mathbf{o}\}. \tag{13}$$

P r o o f. Denote $\mathbf{v} = \mathbf{B}^\top\boldsymbol{\mu}$ for $\boldsymbol{\mu} \neq \mathbf{o}$. Because of $\mathbf{v} \neq \mathbf{o}$, the relation (12) yields $\mathbf{v} \notin \mathcal{N}(\mathbf{A}^\dagger)$. Using Lemma 1 for $\mathbf{A}^\dagger$, we obtain

$$\boldsymbol{\mu}^\top\mathbf{B}\mathbf{A}^\dagger\mathbf{B}^\top\boldsymbol{\mu} = \mathbf{v}^\top\mathbf{A}^\dagger\mathbf{v} > 0$$

so that the statement (i) holds. The statement (ii) follows from the fact that if $\mathbf{A}^\dagger$ is the Moore-Penrose pseudoinverse to $\mathbf{A}$, then $\mathcal{N}(\mathbf{A}) = \mathcal{N}(\mathbf{A}^\dagger)$. □

4. Algorithms

In this section we will propose an algorithm for solving the saddle-point system (1) with singular $\mathbf{A}$. First we recall the well known algorithm for the case of non-singular $\mathbf{A}$. We will assess computational costs of both algorithms by the number of floating point operations (flops) that are perfomed by matrix-vector multiplications.

Let us denote by $n_{A^\dagger}$ and m_B the number of flops needed for the evaluation of one of the matrix-vector products $\mathbf{A}^\dagger\mathbf{v}$, $\mathbf{N}^\top\mathbf{v}$, $\mathbf{N}\boldsymbol{\alpha}$ and $\mathbf{B}\mathbf{v}$, $\mathbf{B}^\top\boldsymbol{\mu}$, respectively.

4.1. Algorithm for the non-singular case

Let $\mathbf{A}$ be non-singular. Eliminating the first unknown $\mathbf{u}$ from the saddle-point system (1), we obtain $\mathbf{u} = \mathbf{A}^{-1}(\mathbf{f} - \mathbf{B}^\top\boldsymbol{\lambda})$ and the linear system $\mathbf{C}\boldsymbol{\lambda} = \mathbf{p}$, where $\mathbf{C} = \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^\top$ is positive definite and $\mathbf{p} = \mathbf{B}\mathbf{A}^{-1}\mathbf{f} - \mathbf{g}$. The algorithm can be divided into three steps:

Algorithm 4.1

Step 1: Assemble $\mathbf{p} = \mathbf{B}\mathbf{A}^{-1}\mathbf{f} - \mathbf{g}$.
Step 2: Solve the linear system $\mathbf{C}\boldsymbol{\lambda} = \mathbf{p}$ using the CGM.
Step 3: Assemble $\mathbf{u} = \mathbf{A}^{-1}(\mathbf{f} - \mathbf{B}^\top\boldsymbol{\lambda})$.

The computational costs of Step 1 and Step 3 are $2(n_{A^\dagger} + m_B)$ flops. The matrix-vector products $\mathbf{C}\boldsymbol{\mu}$ in the CGM are performed by successively evaluating the term $\mathbf{B}(\mathbf{A}^{-1}(\mathbf{B}^\top\boldsymbol{\mu}))$. Assuming that the CGM terminates after m iterations and one multiplication $\mathbf{C}\boldsymbol{\mu}$ is needed per iteration, the computational costs of the Step 2 are $m(n_{A^\dagger} + 2m_B)$ flops.

Lemma 2. *Algorithm* 4.1 *requires* $(m+2)n_{A^\dagger} + 2(m+1)m_B$ *flops.*

4.2. Algorithm for the singular case

Let $\mathbf{A}$ be singular and let (8) and (12) be satisfied. We will use Theorem 2 in order to propose the algorithm. First we compute the pair $(\boldsymbol{\lambda}, \boldsymbol{\alpha})$ solving the linear system (10) and then we evaluate $\mathbf{u}$ by means of (11). In order to simplify the presentation, we introduce new notation:

$$\begin{aligned} \mathbf{C} &= \mathbf{B}\mathbf{A}^\dagger\mathbf{B}^\top, & \mathbf{p} &= \mathbf{B}\mathbf{A}^\dagger\mathbf{f} - \mathbf{g}, \\ \mathbf{D} &= -\mathbf{N}^\top\mathbf{B}^\top, & \mathbf{q} &= -\mathbf{N}^\top\mathbf{f}, \end{aligned}$$

where $\mathbf{C}$ is an $(m \times m)$ positive definite matrix, $\mathbf{D}$ is an $(l \times m)$ full row-rank matrix, $\mathbf{p}$ is an m-vector and $\mathbf{q}$ is an l-vector. Then the linear system (10) reads as follows:

$$\begin{pmatrix} \mathbf{C} & \mathbf{D}^\top \\ \mathbf{D} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} \mathbf{p} \\ \mathbf{q} \end{pmatrix}. \tag{14}$$

Since $\mathbf{C}$ is non-singular, we can eliminate the first unknown $\boldsymbol{\lambda}$ from (14). We obtain

$$\boldsymbol{\lambda} = \mathbf{C}^{-1}(\mathbf{p} - \mathbf{D}^\top \boldsymbol{\alpha}) \tag{15}$$

and

$$\mathbf{E}\boldsymbol{\alpha} = \mathbf{r}, \tag{16}$$

where $\mathbf{E} = \mathbf{D}\mathbf{C}^{-1}\mathbf{D}^\top$ is a positive definite $(l \times l)$ matrix and $\mathbf{r} = \mathbf{D}\mathbf{C}^{-1}\mathbf{p} - \mathbf{q}$ is an l-vector.

Let us point out that, under our assumptions, $\mathbf{D}$, $\mathbf{E}$, $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{r}$ are relatively small. On the other hand, $\mathbf{C}$ is large and non-sparse and therefore we shall propose the algoritm so that it is not neccesary to assemble $\mathbf{C}$. The key idea is the same as in Algorithm 4.1 and consists in the fact that the matrix-vector products $\mathbf{C}\boldsymbol{\mu}$ can be computed in the CGM iterations by successively evaluating the term $\mathbf{B}(\mathbf{A}^\dagger(\mathbf{B}^\top\boldsymbol{\mu}))$.

Algorithm 4.2

```
Step 1.a: Assemble $\mathbf{D} = -\mathbf{N}^\top\mathbf{B}^\top$.
Step 1.b: Assemble $\mathbf{p} = \mathbf{B}\mathbf{A}^\dagger\mathbf{f} - \mathbf{g}$.
Step 1.c: Assemble $\mathbf{q} = -\mathbf{N}^\top\mathbf{f}$.
Step 1.d: Solve the linear systems $\mathbf{C}\mathbf{X} = \mathbf{D}^\top$ by the CGM.
Step 1.e: Solve the linear system $\mathbf{C}\mathbf{x} = \mathbf{p}$ by the CGM.
Step 1.f: Assemble $\mathbf{E} = \mathbf{D}\mathbf{X}$.
Step 1.g: Assemble $\mathbf{r} = \mathbf{D}\mathbf{x} - \mathbf{q}$.
Step 1.h: Solve the linear system $\mathbf{E}\boldsymbol{\alpha} = \mathbf{r}$.
Step 2:   Assemble $\boldsymbol{\lambda} = \mathbf{x} - \mathbf{X}\boldsymbol{\alpha}$.
Step 3:   Assemble $\mathbf{u} = \mathbf{A}^\dagger(\mathbf{f} - \mathbf{B}^\top\boldsymbol{\lambda}) + \mathbf{N}\boldsymbol{\alpha}$.
```

Let us point out that the formula in Step 2 is in fact (15) because $\mathbf{X}$ and $\mathbf{x}$ are computed in Step 1.d and Step 1.e so that $\mathbf{X} = \mathbf{C}^{-1}\mathbf{D}^\top$ and $\mathbf{x} = \mathbf{C}^{-1}\mathbf{p}$, respectively.

The computational costs of Algorithm 4.2 are determined above all by Step 1.a, Step 1.d and Step 1.e. The flops required by the other steps are not significant and therefore we do not count them. Step 1.a requires $mn_{A^\dagger}$ flops. Let us point out that Step 1.d represents l linear systems of order m solved by the CGM. Since one CGM requires $m(n_{A^\dagger}+2m_B)$ flops, Step 1.d and Step 1.e require $(l+1)m(n_{A^\dagger}+2m_B)$ flops.

Lemma 3. *Algorithm* 4.2 *requires* $\mathcal{O}((l+2)mn_{A^\dagger} + 2(l+1)mm_B)$ *flops.*

5. Fast implementation

In this section we will show how to evaluate efficiently the matrix-vector products $\mathbf{A}^{\dagger}\mathbf{v}$, $\mathbf{N}\alpha$ and $\mathbf{N}^{\top}\mathbf{v}$ in Algorithm 4.2 provided $\mathbf{A}$ is of the form (7), i.e.

$$\mathbf{A} = \mathbf{A}_x \otimes \mathbf{I}_y + \mathbf{I}_x \otimes \mathbf{A}_y, \tag{17}$$

where $\mathbf{A}_x$, $\mathbf{A}_y$ are circulant symmetric positive semi-definite, $\mathbf{I}_x$, $\mathbf{I}_y$ are identities and $\otimes$ stands for the Kronecker product. The subscripts x, y indicate that the corresponding matrix is of the order n_x, n_y, respectively. Since we will use the FFT, we assume that n_x, n_y are powers of two.

The multiplying procedures for $\mathbf{A}$ are based on properties of the circulant matrices.

5.1. Circulant matrices and the DFT

A matrix $\mathbf{A}$ is called the *circulant matrix* if

$$\mathbf{A} = \begin{pmatrix} a_1 & a_n & \dots & a_2 \\ a_2 & a_1 & \dots & a_3 \\ a_3 & a_2 & \dots & a_4 \\ \vdots & \vdots & \ddots & \vdots \\ a_n & a_{n-1} & \dots & a_1 \end{pmatrix},$$

i.e. each column of $\mathbf{A}$ is a cyclic shift of the preceding column to the bottom. Let us denote the first column of $\mathbf{A}$ by $\mathbf{a}$, i.e. $\mathbf{a} = (a_1, a_2, \dots, a_n)^{\top}$.

There are important conections between the circulant matrices and the DFT. A *DFT matrix* is defined by $\mathbf{F} = (\omega^{(k-1)(l-1)})_{k,l=1}^{n}$, $\omega = \mathrm{e}^{-\mathrm{i}2\pi/n}$. The DFT of the n-vector $\mathbf{v}$ is

$$\hat{\mathbf{v}} = \mathbf{F}\mathbf{v}.$$

If n is a power of two, then it is possible to evaluate this product by the FFT with $\mathcal{O}(n \log_2 n)$ flops [7]. In this case, we will use the notation $\hat{\mathbf{v}} := \mathtt{fft}(\mathbf{v})$. Let us point out that $\mathbf{F}$ is symmetric and fulfils $\mathbf{F}\overline{\mathbf{F}}^{\top} = n\mathbf{I}$ so that

$$\mathbf{v} = \frac{1}{n}\mathbf{F}\overline{\hat{\mathbf{v}}}, \tag{18}$$

if $\mathbf{v}$ has real entries. In view of (18), the inverse DFT can be evaluated again by the FFT with $\mathcal{O}(n \log_2 n)$ flops, which will be denoted by $\mathbf{v} := \mathtt{ifft}(\hat{\mathbf{v}})$.

Lemma 4. *Let* $\mathbf{A}$ *be a circulant matrix. Then* $\mathbf{A} = \mathbf{F}^{-1}\mathbf{\Lambda}\mathbf{F}$, *where* $\mathbf{\Lambda} = \operatorname{diag}(\hat{\mathbf{a}})$.

P r o o f. It is well known that the Fourier transform changes translation operators into modulation ones [10]. The equality of columns in $\mathbf{FA} = \mathbf{\Lambda F}$ represents this property. □

Let us point out that diagonal entries of $\mathbf{\Lambda}$ are eigenvalues of $\mathbf{A}$ and columns of $\mathbf{F}^{-1}$ are corresponding eigenvectors. Lemma 4 proves that the eigenvalues of a circulant matrix can be computed very cheaply by the FFT of its first column $\mathbf{a}$ and, moreover, the eigenvectors need not be computed at all because they are known apriori.

Lemma 5. *Let* $\mathbf{A}$ *be a circulant matrix,* $\mathbf{\Lambda} = \operatorname{diag}(\hat{\mathbf{a}})$ *and let* $\mathbf{\Lambda}^{\dagger}$ *be defined so that the non-vanishing entries of* $\mathbf{\Lambda}$ *are inverted. Then* $\mathbf{A}^{\dagger} = \mathbf{F}^{-1}\mathbf{\Lambda}^{\dagger}\mathbf{F}$ *is the Moore-Penrose pseudoinverse to* $\mathbf{A}$.

P r o o f. It is easy to verify that $\mathbf{A}^{\dagger}$ fulfils the relations defining the Moore-Penrose pseudoinverse, see e.g. [7]. □

Using Lemma 5, we can evaluate the matrix-vector product $\mathbf{A}^{\dagger}\mathbf{v}$ by the following three steps:

$$1^\circ \quad \mathbf{v} := \texttt{fft}(\mathbf{v}), \qquad 2^\circ \quad \mathbf{v} := \mathbf{\Lambda}^{\dagger}\mathbf{v}, \qquad 3^\circ \quad \mathbf{A}^{\dagger}\mathbf{v} := \texttt{ifft}(\mathbf{v}).$$

Since $\mathbf{\Lambda}^{\dagger}$ is diagonal, it requires $\mathcal{O}(2n\log_2 n + n)$ flops and the preliminary computation of $\hat{\mathbf{a}}$.

Using Lemma 4, we can evaluate the matrix-vector products $\mathbf{N}\alpha$ and $\mathbf{N}^{\top}\mathbf{v}$, where the columns of $\mathbf{N}$ span the null space of the circulant matrix $\mathbf{A}$. We can suppose that $\mathbf{N}$ consists of eigenvectors from $\mathbf{F}^{-1}$ corresponding to the positions of the vanishing eigenvalues of $\mathbf{A}$, i.e. to the vanishing entries of $\hat{\mathbf{a}}$. In order to determine the positions of the desirable columns in $\mathbf{F^{-1}}$, we introduce the operation $\texttt{ind}_{\hat{\mathbf{a}}}$,

$$\boldsymbol{\alpha} \in \mathbb{R}^l \Longleftrightarrow \mathbf{v}_{\boldsymbol{\alpha}} := \texttt{ind}_{\hat{\mathbf{a}}}(\boldsymbol{\alpha}) \in \mathbb{R}^n,$$

where the entries of $\boldsymbol{\alpha}$ are put in $\mathbf{v}_{\boldsymbol{\alpha}}$ to the positions of zeros in $\hat{\mathbf{a}}$ and the remaining entries of $\mathbf{v}_{\boldsymbol{\alpha}}$ vanish. Let us denote by $\texttt{ind}_{\hat{\mathbf{a}}}^{-1}$ the reverse operation to $\texttt{ind}_{\hat{\mathbf{a}}}$, i.e.

$$\boldsymbol{\alpha} := \texttt{ind}_{\hat{\mathbf{a}}}^{-1}(\mathbf{v}_{\boldsymbol{\alpha}}).$$

It is easily seen that

$$\begin{aligned} \mathbf{N}\boldsymbol{\alpha} &= \mathbf{F}^{-1}\texttt{ind}_{\hat{\mathbf{a}}}(\boldsymbol{\alpha}), \\ \mathbf{N}^{\top}\mathbf{v} &= \texttt{ind}_{\hat{\mathbf{a}}}^{-1}(\mathbf{F}^{-1}\mathbf{v}). \end{aligned}$$

Hence the matrix-vector products $\mathbf{N}\boldsymbol{\alpha}$ and $\mathbf{N}^\top\mathbf{v}$ can be evaluated by two steps:

$$1^\circ \quad \mathbf{v}_{\boldsymbol{\alpha}} := \mathtt{ind}_{\hat{\mathbf{a}}}(\boldsymbol{\alpha}), \qquad 2^\circ \quad \mathbf{N}\boldsymbol{\alpha} := \mathtt{ifft}(\mathbf{v}_{\boldsymbol{\alpha}})$$

and

$$1^\circ \quad \mathbf{v} := \mathtt{ifft}(\mathbf{v}), \qquad 2^\circ \quad \mathbf{N}^\top\mathbf{v} := \mathtt{ind}_{\hat{\mathbf{a}}}^{-1}(\mathbf{v}),$$

respectively. The computational costs are $\mathcal{O}(n\log_2 n)$ flops in both cases.

5.2. Kronecker product

Consider matrices $\mathbf{A}_x$ and $\mathbf{A}_y$. Their *Kronecker product* is defined by

$$\mathbf{A}_x \otimes \mathbf{A}_y = \begin{pmatrix} a^y_{11}\mathbf{A}_x & \dots & a^y_{1n_y}\mathbf{A}_x \\ \vdots & \ddots & \vdots \\ a^y_{n_y1}\mathbf{A}_x & \dots & a^y_{n_yn_y}\mathbf{A}_x \end{pmatrix},$$

where a^y_{kl} are the entries of $\mathbf{A}_y$. In other words, $\mathbf{A}_x \otimes \mathbf{A}_y$ is the $n \times n$ matrix (with $n = n_xn_y$) whose (k,l)th block is $a^y_{kl}\mathbf{A}_x$. The relationship between the Kronecker product and the matrix-matrix product is given by the equality [7]:

$$(\mathbf{A}_x \otimes \mathbf{A}_y)(\mathbf{B}_x \otimes \mathbf{B}_y) = \mathbf{A}_x\mathbf{B}_x \otimes \mathbf{A}_y\mathbf{B}_y. \tag{19}$$

Let us point out that (19) implies

$$(\mathbf{A}_x \otimes \mathbf{A}_y)^{-1} = \mathbf{A}_x^{-1} \otimes \mathbf{A}_y^{-1}. \tag{20}$$

It is a favourable feature of the Kronecker product that matrix-vector products $(\mathbf{A}_x \otimes \mathbf{A}_y)\mathbf{v}$ can be split into multiplications by the particular matrices $\mathbf{A}_x$ and $\mathbf{A}_y$. To this end, we introduce the operation `vec`,

$$\mathbf{V} = (\mathbf{v}_1, \dots, \mathbf{v}_{n_y}) \in \mathbb{R}^{n_x \times n_y} \Longleftrightarrow \mathtt{vec}(\mathbf{V}) =: \begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{n_y} \end{pmatrix} \in \mathbb{R}^{n_xn_y}.$$

Denote by $\mathtt{vec}^{-1}$ the reverse operation to `vec`, i.e. if $\mathbf{v} := \mathtt{vec}(\mathbf{V})$, then $\mathbf{V} := \mathtt{vec}^{-1}(\mathbf{v})$.

Lemma 6. *The identity*

$$(\mathbf{A}_x \otimes \mathbf{A}_y)\mathbf{v} = \mathtt{vec}(\mathbf{A}_x \mathbf{V} \mathbf{A}_y^\top)$$

holds, where $\mathbf{V} := \mathtt{vec}^{-1}(\mathbf{v})$.

P r o o f. The proof follows by direct evaluation of the products. $\square$

Using Lemma 6, we can evaluate the matrix-vector product $(\mathbf{A}_x \otimes \mathbf{A}_y)\mathbf{v}$ in the following way:

$$\mathbf{V} := \mathtt{vec}^{-1}(\mathbf{v}), \qquad \mathbf{V} := \mathbf{A}_x\mathbf{V}, \qquad \mathbf{V} := \mathbf{V}\mathbf{A}_y^\top, \qquad (\mathbf{A}_x \otimes \mathbf{A}_y)\mathbf{v} := \mathtt{vec}(\mathbf{V}).$$

Denote by n_{A_x}, n_{A_y} the number of flops needed for evaluation of the matrix-vector products $\mathbf{A}_x\mathbf{v}_x$, $\mathbf{A}_y\mathbf{v}_y$, respectively. Since vec and $\mathtt{vec}^{-1}$ do not require any flops, the computational costs of $(\mathbf{A}_x \otimes \mathbf{A}_y)\mathbf{v}$ are $n_y n_{A_x} + n_x n_{A_y}$ flops.

5.3. Multiplying procedures

We will combine the techniques from the previous two sections in order to obtain fast multiplying procedures for the matrix (17). We will use the following notation: $\mathbf{F}_x$, $\mathbf{F}_y$ denote the DFT matrices of the orders n_x, n_y, respectively; $\mathbf{a}_x$, $\mathbf{a}_y$ denote the first columns of the circulant matrices $\mathbf{A}_x$, $\mathbf{A}_y$, respectively.

Lemma 7. *Let* $\mathbf{A}_x$, $\mathbf{A}_y$ *be circulant matrices and* $\mathbf{A} = \mathbf{A}_x \otimes \mathbf{I}_y + \mathbf{I}_x \otimes \mathbf{A}_y$. *Then* $\mathbf{A} = \mathbf{F}^{-1}\mathbf{\Lambda}\mathbf{F}$, *where* $\mathbf{F} = \mathbf{F}_x \otimes \mathbf{F}_y$, $\mathbf{\Lambda} = \mathbf{\Lambda}_x \otimes \mathbf{I}_y + \mathbf{I}_x \otimes \mathbf{\Lambda}_y$, $\mathbf{\Lambda}_x = \operatorname{diag}(\hat{\mathbf{a}}_x)$ *and* $\mathbf{\Lambda}_y = \operatorname{diag}(\hat{\mathbf{a}}_y)$.

P r o o f. Using $\mathbf{A}_x = \mathbf{F}_x^{-1}\mathbf{\Lambda}_x\mathbf{F}_x$, $\mathbf{A}_y = \mathbf{F}_y^{-1}\mathbf{\Lambda}_y\mathbf{F}_y$ (see Lemma 4) and (19), we obtain

$$\begin{aligned}\mathbf{A} &= \mathbf{F}_x^{-1}\mathbf{\Lambda}_x\mathbf{F}_x \otimes \mathbf{F}_y^{-1}\mathbf{F}_y + \mathbf{F}_x^{-1}\mathbf{F}_x \otimes \mathbf{F}_y^{-1}\mathbf{\Lambda}_y\mathbf{F}_y \\ &= (\mathbf{F}_x^{-1} \otimes \mathbf{F}_y^{-1})(\mathbf{\Lambda}_x \otimes \mathbf{I}_y + \mathbf{I}_x \otimes \mathbf{\Lambda}_y)(\mathbf{F}_x \otimes \mathbf{F}_y).\end{aligned}$$

Then (20) proves the lemma. $\square$

Let us point out that Lemma 7 is formally the same as Lemma 4. Therefore a lemma analogous to Lemma 5 holds.

Lemma 8. *Let $\mathbf{A}_x$, $\mathbf{A}_y$ be circulant matrices, $\mathbf{\Lambda}_x = \operatorname{diag}(\hat{\mathbf{a}}_x)$, $\mathbf{\Lambda}_y = \operatorname{diag}(\hat{\mathbf{a}}_y)$, $\mathbf{F} = \mathbf{F}_x \otimes \mathbf{F}_y$, and let $\mathbf{\Lambda}^\dagger$ be defined so that the non-zero entries of $\mathbf{\Lambda} = \mathbf{\Lambda}_x \otimes \mathbf{I}_y + \mathbf{I}_x \otimes \mathbf{\Lambda}_y$ are inverted. Then $\mathbf{A}^\dagger = \mathbf{F}^{-1}\mathbf{\Lambda}^\dagger\mathbf{F}$ is the Moore-Penrose pseudoinverse to $\mathbf{A} = \mathbf{A}_x \otimes \mathbf{I}_y + \mathbf{I}_x \otimes \mathbf{A}_y$.*

Using this lemma, we can propose a multiplying procedure for the evaluation of the matrix-vector product $\mathbf{A}^\dagger\mathbf{v}$:

Procedure `Aplus_v`

```
Input: $\hat{\mathbf{a}}_x$, $\hat{\mathbf{a}}_y$, $\mathbf{V} := \mathtt{vec}^{-1}(\mathbf{v})$
1° $\mathbf{V} := \mathtt{fft}(\mathbf{V})$
2° $\mathbf{V} := \mathtt{fft}(\mathbf{V}^\top)^\top$
3° $\mathbf{V} := \mathtt{vec}^{-1}(\mathbf{\Lambda}^\dagger\mathtt{vec}(\mathbf{V}))$
4° $\mathbf{V} := \mathtt{ifft}(\mathbf{V})$
5° $\mathbf{V} := \mathtt{ifft}(\mathbf{V}^\top)^\top$
Output: $\mathbf{A}^\dagger\mathbf{v} := \mathtt{vec}(\mathbf{V})$
```

Here, we suppose that `fft` and `ifft` are independently performed for the individual columns of the matrices $\mathbf{V}$ or $\mathbf{V}^\top$.

Lemma 9. *The multiplying procedure* `Aplus_v` *requires $\mathcal{O}(2n\log_2 n + n)$ flops.*

P r o o f. Recall that $\mathbf{V}$ is an $(n_x \times n_y)$ matrix. The steps 1° and 2° involve

$$n_y\mathcal{O}(n_x\log_2 n_x) + n_x\mathcal{O}(n_y\log_2 n_y) = \mathcal{O}(n\log_2 n) \tag{21}$$

flops. The same flops are required by the steps 4° and 5°. Since $\mathbf{\Lambda}^\dagger$ is diagonal, the step 3° requires n flops. □

Lemma 10. *Let $\mathbf{A}_x$, $\mathbf{A}_y$ be symmetric positive semi-definite matrices with $l_x = \dim\mathcal{N}(\mathbf{A}_x)$, $l_y = \dim\mathcal{N}(\mathbf{A}_y)$ and $l_x \geqslant 1$, $l_y \geqslant 1$, respectively. Let $\mathbf{N}_x$, $\mathbf{N}_y$ be matrices whose columns span the null spaces of $\mathbf{A}_x$, $\mathbf{A}_y$, respectively. Then the columns of $\mathbf{N} = \mathbf{N}_x \otimes \mathbf{N}_y$ span the null space of $\mathbf{A} = \mathbf{A}_x \otimes \mathbf{I}_y + \mathbf{I}_x \otimes \mathbf{A}_y$.*

P r o o f. Using (19), we obtain

$$\mathbf{AN} = \mathbf{A}_x\mathbf{N}_x \otimes \mathbf{N}_y + \mathbf{N}_x \otimes \mathbf{A}_y\mathbf{N}_y = \mathbf{0} \otimes \mathbf{N}_y + \mathbf{N}_x \otimes \mathbf{0} = \mathbf{0}.$$

Since the number of columns in $\mathbf{N}$ is the same as $\dim\mathcal{N}(\mathbf{A})$, the lemma holds. □

If $\mathbf{A}_x$, $\mathbf{A}_y$ are cirulant matrices, we can identify $\mathbf{N}_x$, $\mathbf{N}_y$ with the columns of $\mathbf{F}_x^{-1}$, $\mathbf{F}_y^{-1}$ corresponding to the positions of the vanishing entries of $\hat{\mathbf{a}}_x$, $\hat{\mathbf{a}}_y$, respectively. To this end, we introduce the operation $\mathtt{Ind}_{\hat{\mathbf{a}}_x,\hat{\mathbf{a}}_y}$,

$$\boldsymbol{\alpha} \in \mathbb{R}^{l_x l_y} \Longleftrightarrow \mathbf{V}_{\boldsymbol{\alpha}} := \mathtt{Ind}_{\hat{\mathbf{a}}_x,\hat{\mathbf{a}}_y}(\boldsymbol{\alpha}) \in \mathbb{R}^{n_x \times n_y},$$

so that the entries of $\boldsymbol{\alpha}$ are taken as the entries of $\mathbf{V}_{\boldsymbol{\alpha}}$ with the row indices corresponding to the positions of zeros in $\hat{\mathbf{a}}_x$ and with the column indices corresponding to the positions of zeros in $\hat{\mathbf{a}}_y$. The remaining entries of $\mathbf{V}_{\boldsymbol{\alpha}}$ vanish. Let us denote by $\mathtt{Ind}^{-1}_{\hat{\mathbf{a}}_x,\hat{\mathbf{a}}_y}$ the reverse operation to $\mathtt{Ind}_{\hat{\mathbf{a}}_x,\hat{\mathbf{a}}_y}$, i.e.

$$\boldsymbol{\alpha} := \mathtt{Ind}^{-1}_{\hat{\mathbf{a}}_x,\hat{\mathbf{a}}_y}(\mathbf{V}_{\boldsymbol{\alpha}}).$$

Using Lemma 10 and Lemma 6, it is easy to verify that

$$\begin{aligned}\mathbf{N}\boldsymbol{\alpha} &= \mathtt{vec}(\mathbf{F}_x^{-1}\mathtt{Ind}_{\hat{\mathbf{a}}_x,\hat{\mathbf{a}}_y}(\boldsymbol{\alpha})\mathbf{F}_y^{-1}),\\ \mathbf{N}^\top\mathbf{v} &= \mathtt{Ind}^{-1}_{\hat{\mathbf{a}}_x,\hat{\mathbf{a}}_y}(\mathbf{F}_x^{-1}\mathtt{vec}^{-1}(\mathbf{v})\mathbf{F}_y^{-1}).\end{aligned}$$

Hence the matrix-vector products $\mathbf{N}\boldsymbol{\alpha}$ and $\mathbf{N}^\top\mathbf{v}$ can be evaluated by the following multiplying procedures:

```
Procedure N_alpha
Input: $\hat{\mathbf{a}}_x$, $\hat{\mathbf{a}}_y$, $\boldsymbol{\alpha}$
1° $\mathbf{V}_{\boldsymbol{\alpha}} := \mathtt{Ind}_{\hat{\mathbf{a}}_x,\hat{\mathbf{a}}_y}(\boldsymbol{\alpha})$
2° $\mathbf{V}_{\boldsymbol{\alpha}} := \mathtt{ifft}(\mathbf{V}_{\boldsymbol{\alpha}})$
3° $\mathbf{V}_{\boldsymbol{\alpha}} := \mathtt{ifft}(\mathbf{V}_{\boldsymbol{\alpha}}^\top)^\top$
Output: $\mathbf{N}\boldsymbol{\alpha} := \mathtt{vec}(\mathbf{V}_{\boldsymbol{\alpha}})$
```

```
Procedure Ntranspose_v
Input: $\hat{\mathbf{a}}_x$, $\hat{\mathbf{a}}_y$, $\mathbf{V} := \mathtt{vec}^{-1}(\mathbf{v})$
1° $\mathbf{V} := \mathtt{ifft}(\mathbf{V})$
2° $\mathbf{V} := \mathtt{ifft}(\mathbf{V}^\top)^\top$
3° $\mathbf{N}^\top\mathbf{v} := \mathtt{Ind}^{-1}_{\hat{\mathbf{a}}_x,\hat{\mathbf{a}}_y}(\mathbf{V})$
Output: $\mathbf{N}^\top\mathbf{v}$
```

Lemma 11. *The multiplying procedures* N_alpha *and* Ntranspose_v *require* $\mathcal{O}(n\log_2 n)$ *flops.*

Proof. The computational costs are determined by (21) because $\mathtt{Ind}_{\hat{\mathbf{a}}_x,\hat{\mathbf{a}}_y}$ and $\mathtt{Ind}^{-1}_{\hat{\mathbf{a}}_x,\hat{\mathbf{a}}_y}$ do not require any flops. □

5.4. Complexity of algorithms

We will assume the multiplying procedures `Aplus_v`, `N_alpha` and `Ntranspose_v` are used for evaluation of the matrix-vector product $\mathbf{A}^{\dagger}\mathbf{v}$, $\mathbf{N}\boldsymbol{\alpha}$ and $\mathbf{N}^{\top}\mathbf{v}$, respectively. Then

$$n_{A^{\dagger}} = \mathcal{O}(n \log_2 n)$$

in Lemma 2 and Lemma 3. Moreover,

$$m_B = \mathcal{O}(m)$$

since $\mathbf{B}$ is sparse. We obtain the following results.

Theorem 4. *Algorithm* 4.1 *for solving the saddle-point system* (1) *with* $\mathbf{A}$ *non-singular of the form* (17) *and with* $\mathbf{B}$ *sparse requires*

$$\mathcal{O}((m+2)n \log_2 n) \quad \textit{flops.}$$

Theorem 5. *Algorithm* 4.2 *for solving the saddle-point system* (1) *with* $\mathbf{A}$ *singular of the form* (17) *and with* $\mathbf{B}$ *sparse requires*

$$\mathcal{O}((l+2)mn \log_2 n) \quad \textit{flops.}$$

Let us point out that faster implementations can be achieved by using parallelizations which can be naturally realized on the level of the general scheme of the algorithm (parallelization of the CGMs performed in Step 1.d and Step 1.e of Algorithm 4.2) as well as on the level of its fast implementation (parallelization of the mutually independent FFTs of columns in `fft`($\mathbf{V}$) and `ifft`($\mathbf{V}$)).

6. Numerical experiments

We will assess complexities of Algorithm 4.1 and Algorithm 4.2 experimentaly. Let us consider the saddle-point linear system (1) arising in wavelet-Galerkin discretization of the problem (2), (3), where $\omega = \{(x,y) \in \mathbb{R}^2 \colon (x/0.2)^2 + (y/0.3)^2 \leqslant 1\}$, $f(x,y) = 1$ on $\langle -0.5, 0.5\rangle \times \langle -0.5, 0.5\rangle$, $f(x,y) = 0$ elsewhere, $g \equiv 0$ and $\Omega = \langle -1, 1\rangle \times \langle -1, 1\rangle$.

If $c > 0$, then $\mathbf{A}$ is non-singular so that we use Algorithm 4.1.

If $c = 0$, then $\mathbf{A}$ is singular with $l = 1$ $(= \dim \mathcal{N}(\mathbf{A}))$ and $\mathbf{N}$ has one column whose entries are non-vanishing and equal, e.g. "1". Since $\mathbf{B}$ describes the geometry of ω in Ω, we can deduce that $\mathbf{B}$ has vanishing columns. Therefore $\mathbf{N} \notin \mathcal{R}(\mathbf{B}^\top)$ so that (13) is satisfied and Algorithm 4.2 can be used.

All numerical experiments have been performed by Matlab 6 on Pentium(R)4, 3.00 GHz with 512MB RAM. We summarize them in Tab. 1. The relative tolerance terminating the CGM is 10^{-4} in all cases. With respect to the theoretical results of Theorem 4 and Theorem 5, the computational time of Algorithm 4.2 should be approximately three times longer than the computational time of Algorithm 4.1 (for the same n, m). We can see that the experimental results are better, since we have sped up the multiplying procedures `N_alpha` and `Ntranspose_v` using the apriori knowledge of $\mathbf{N}$.

Let us point out that the experimental results affirm high efficiency of the fast implementation of the proposed algorithm.

		$c = 1$		$c = 0$	
n	m	time	CGM steps	time	CGM steps
1024	64	0.03	13	0.06	$10 + 16$
2048	88	0.05	17	0.08	$14 + 21$
4096	128	0.06	17	0.13	$13 + 21$
8192	180	0.17	23	0.30	$20 + 29$
16384	256	0.28	21	0.48	$18 + 25$
32768	360	0.67	27	1.20	$25 + 33$
65536	512	2.27	27	5.58	$25 + 33$
131072	716	7.22	35	15.17	$33 + 43$
262144	1024	14.72	34	29.33	$29 + 38$
524288	1432	35.70	45	73.41	$43 + 53$
1048576	2048	65.56	41	133.75	$38 + 49$
2097152	2868	173.53	54	347.41	$50 + 62$
4194304	4096	337.95	48	655.00	$42 + 57$

Table 1. CPU time (in seconds) and the CGM steps.

References

[1] *F. Brezzi, M. Fortin*: Mixed and Hybrid Finite Element Methods. Springer-Verlag, New York, 1991.

[2] *P. G. Ciarlet*: The Finite Element Method for Elliptic Problems. North-Holland, Amsterdam, 1978.

[3] *M. Fortin, R. Glowinski*: Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems. North-Holland, Amsterdam, 1983.

[4] *I. Daubechies*: Ten Lectures on Wavelets. SIAM, Philadelphia, 1992.

[5] *R. Glowinski, T. Pan, and J. Periaux*: A fictitious domain method for Dirichlet problem and applications. Comput. Methods Appl. Mech. Eng. *111* (1994), 283–303.

[6] *R. Glowinski, T. Pan, R. O. Wells, X. Zhou*: Wavelets methods in computational fluid dynamics. In: Proc. Algorithms Trends in Computational Dynamics (1993) (M. Y. Hussaini, A. Kumar, and M. D. Salas, eds.). Springer-Verlag, New York, pp. 259–276.

[7] *G. H. Golub, C. F. Van Loan*: Matrix Computation. The Johns Hopkins University Press, Baltimore, 1996, 3rd ed.

[8] *N. I. M. Gould*: On practical conditions for the existence and uniqueness of solutions to the general equality quadratic programming problem. Math. Program. *32* (1985), 90–99.

[9] *R. Kučera*: Wavelet solution of elliptic PDEs. In: Proc. Matematyka v Naukach Technicznych i Przyrodniczych (2000) (S. Bialas, ed.). AGH Krakow, pp. 55–62.

[10] *W. Rudin*: Real and Complex Analysis. McGraw-Hill, New York, 1987, 3rd ed.

Author's address: *R. Kučera*, Department of Mathematics and Descriptive Geometry, VŠB-Technical University of Ostrava, Tř. 17. listopadu 15, CZ-708 33 Ostrava-Poruba, Czech Republic, e-mail: `radek.kucera@vsb.cz`.