Radek Kučera

Complexity and memory requirements of an algorithm for solving saddle-point linear systems with singular blocks

In: Jan Chleboun and Petr Přikryl and Karel Segeth (eds.): Programs and Algorithms of Numerical Mathematics, Proceedings of Seminar. Dolní Maxov, June 6-11, 2004. Institute of Mathematics AS CR, Prague, 2004.
pp. 131–135.

Persistent URL: http://dml.cz/dmlcz/702785

COMPLEXITY AND MEMORY REQUIREMENTS OF AN ALGORITHM FOR SOLVING SADDLE-POINT LINEAR SYSTEMS WITH SINGULAR BLOCKS *

Radek Kučera

Abstract

The paper deals with fast solution 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 possible to use the fast Fourier transform that determines the complexity of the algorithm. Numerical experiments confirm theoretical results.

1. Formulation of the problem

We shall 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* [1], [2]:

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

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 shall be interested especially in systems (1) with n large, $\mathbf{A}$ singular, $\mathbf{B}$ sparse and m much smaller than n. Moreover we shall assume that the defect of $\mathbf{A}$, i.e. $l = n - rank\mathbf{A}$, is much smaller than m.

2. Model PDE problem

Let ω be a bounded domain in $\mathbb{R}^2$ with a smooth boundary $\partial\omega$. We shall consider the following model 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 $\omega, \partial\omega$, respectively, and $c \geq 0$ is a given constant. We imbed ω in a larger rectangular domain Ω so that $\overline{\omega} \subset \Omega$ and denote by $\partial\Omega$ the boundary of Ω. On Ω, we shall solve (2), (3) by means of

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

the fictitious domain method with the boundary Lagrange multipliers; see [4]. We replace (2), (3) by the following *saddle-point problem*:

$$\left.\begin{aligned} &\text{Find } (u,\lambda) \in H^1_{per}(\Omega) \times H^{-1/2}(\partial\omega) \text{ such that} \\ &a_\Omega(u,v) = \int_\Omega \tilde{f} v \, dx + \langle \lambda, v \rangle \quad \forall v \in H^1_{per}(\Omega), \\ &\langle \mu, u - g \rangle = 0 \quad \forall \mu \in H^{-1/2}(\partial\omega), \end{aligned}\right\} \tag{4}$$

where $H^1_{per}(\Omega)$ denotes the subset of functions from $H^1(\Omega)$ that are periodic on the opposite sides of the rectangle $\partial\Omega$, $H^{-1/2}(\partial\omega)$ is dual to $H^{1/2}(\partial\omega)$ with the duality pairing denoted by $\langle \cdot, \cdot \rangle$, $\tilde{f} \in L^2(\Omega)$ extends f from ω onto Ω and $a_\Omega(v,w) = \int_\Omega (\nabla v \cdot \nabla w + cvw)\, dx$.

It is well-known that (4) has a unique solution (u,λ) and that the restriction of u on ω is a solution to a weak formulation of (2), (3); see [4].

We introduce $H^1_{per}(\Omega)$ because we discretize (4) by the periodized orthogonal compactly supported wavelets of the tensor product type; see [3]. After discretization, we obtain the saddle-point linear system (1). The diagonal block $\mathbf{A}$ is singular if $c = 0$ and non-singular if $c > 0$. Since we use the tensor product basis functions on the rectangular domain Ω, we can represent $\mathbf{A}$ using the Kronecker product as

$$\mathbf{A} = \mathbf{A}_x \otimes \mathbf{I}_y + \mathbf{I}_x \otimes \mathbf{A}_y, \tag{5}$$

where $\mathbf{I}_x$, $\mathbf{I}_y$ and $\mathbf{A}_x$, $\mathbf{A}_y$ are of the order n_x, n_y, respectively, and $n = n_x n_y$. Here, $\mathbf{A}_x$, $\mathbf{A}_y$ 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 [6]

3. Algorithm

Let us consider an $n \times l$ matrix $\mathbf{N}$ whose columns span the null-space of $\mathbf{A}$ and denote by $\mathbf{A}^\dagger$ a generalized inverse to $\mathbf{A}$. The first component $\mathbf{u}$ of the solution to (1) is given by

$$\mathbf{u} = \mathbf{A}^\dagger(\mathbf{f} - \mathbf{B}^\top \boldsymbol{\lambda}) + \mathbf{N}\boldsymbol{\alpha}, \tag{6}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^l$. Inserting (6) into the second equation in (1) and using the orthogonality $\mathbf{f} - \mathbf{B}^\top \boldsymbol{\lambda} \perp Ker\, \mathbf{A}$, we obtain

$$\begin{pmatrix} \mathbf{C} & \mathbf{D}^\top \\ \mathbf{D} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} \mathbf{p} \\ \mathbf{q} \end{pmatrix}, \tag{7}$$

where $\mathbf{C} = \mathbf{B}\mathbf{A}^\dagger\mathbf{B}^\top$, $\mathbf{p} = \mathbf{B}\mathbf{A}^\dagger\mathbf{f} - \mathbf{g}$, $\mathbf{D} = -\mathbf{N}^\top\mathbf{B}^\top$, $\mathbf{q} = -\mathbf{N}^\top\mathbf{f}$. Assume that $\mathbf{C}$ is positive definite. Then we can eliminate $\boldsymbol{\lambda}$ from (7) so that

$$\boldsymbol{\lambda} = \mathbf{C}^{-1}(\mathbf{p} - \mathbf{D}^\top\boldsymbol{\alpha}), \tag{8}$$

and

$$\mathbf{E}\boldsymbol{\alpha} = \mathbf{r}, \tag{9}$$

where $\mathbf{E} = \mathbf{D}\mathbf{C}^{-1}\mathbf{D}^\top$ is positive definite and $\mathbf{r} = \mathbf{D}\mathbf{C}^{-1}\mathbf{p} - \mathbf{q}$.

ALGORITHM 2.1 (GENERAL SCHEME)

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

Computational costs are determined above all by the steps 1.a, 1.d and 1.e. Let us point out that $\mathbf{C}$ is not assembled. The matrix-vector products $\mathbf{C}\boldsymbol{\mu}$ in the conjugate gradients iterations are computed by succesively evaluating the term $\mathbf{B}(\mathbf{A}^\dagger(\mathbf{B}^\top \boldsymbol{\mu}))$. Assuming that the conjugate gradients terminate after m iterations and denoting by $n_{A^\dagger}$ the number of floating point operations (flops) involved by the matrix-vector products $\mathbf{A}^\dagger \mathbf{v}$ and $\mathbf{N}^\top \mathbf{v}$, the computational costs of the algorithm are $\mathcal{O}((l+2)mn_{A^\dagger})$ flops.

4. Fast implementation

We can utilize properties of circulant matrices and the Kronecker product in order to obtain a fast implementation of ALGORITHM 2.1.

A matrix $\mathbf{A}_x$ is called *circulant* if each column is a cyclic shift of the column above to the bottom, i.e. if $\mathbf{a}_x = (a_1, \ldots, a_{n_x})^\top$ is the first column of $\mathbf{A}_x$ then the j-th column $\mathbf{a}_x^j$, $2 \le j \le n_x$, is given by $\mathbf{a}_x^j = (a_{n_x-j+2}, \ldots, a_{n_x}, a_1, \ldots, a_{n_x-j+1})^\top$. It is well-known that the spectral decompositon of $\mathbf{A}_x$ can be expressed by the *discrete Fourier transform* (DFT) matrix $\mathbf{F}_x$ so that

$$\mathbf{A}_x = \mathbf{F}_x^{-1} \boldsymbol{\Lambda}_x \mathbf{F}_x, \tag{10}$$

where $\boldsymbol{\Lambda}_x = diag(\widehat{\mathbf{a}}_x)$ and $\widehat{\mathbf{a}}_x = \mathbf{F}_x \mathbf{a}_x$; see [5]. In other words, the eigenvalues of $\mathbf{A}_x$ are entries of the DFT of $\mathbf{a}_x$ and the corresponding eigenvectors are columns of $\mathbf{F}_x^{-1}$. If n_x is a power of two, then the matrix-vector products $\mathbf{A}_x^\dagger \mathbf{v}_x$ can be evaluated by $\mathcal{O}(n_x \log_2 n_x)$ flops so that

$$\mathbf{A}_x^\dagger \mathbf{v}_x := \texttt{ifft}(\boldsymbol{\Lambda}_x^\dagger \texttt{fft}(\mathbf{v}_x)),$$

where `fft` and `ifft` denote the fast Fourier transform and its inverse, respectively. Moreover, the $n_x \times l_x$ matrix $\mathbf{N}_x$ whose columns span the null-space of $\mathbf{A}_x$ can be identified with the columns of $\mathbf{F}_x^{-1}$ corresponding to the positions of vanishing entries of $\widehat{\mathbf{a}}_x$. To this end, we introduce the operation $\texttt{ind}_{\widehat{\mathbf{a}}_x}$,

$$\boldsymbol{\alpha}_x \in \mathbb{R}^{l_x} \quad \Longleftrightarrow \quad \mathbf{v}_{\boldsymbol{\alpha}_x} := \texttt{ind}_{\widehat{\mathbf{a}}_x}(\boldsymbol{\alpha}_x) \in \mathbb{R}^{n_x},$$

so that the entries of $\boldsymbol{\alpha}_x$ are put in $\mathbf{v}_{\boldsymbol{\alpha}_x}$ onto the positions of zeros in $\widehat{\mathbf{a}}_x$ and the remaining entries of $\mathbf{v}_{\boldsymbol{\alpha}_x}$ vanish. Let us denote by $\mathtt{ind}^{-1}_{\widehat{\mathbf{a}}_x}$ the reverse operation to $\mathtt{ind}_{\widehat{\mathbf{a}}_x}$. It is easy to verify that

$$\mathbf{N}_x\boldsymbol{\alpha}_x := \mathbf{F}_x^{-1}\mathtt{ind}_{\widehat{\mathbf{a}}_x}(\boldsymbol{\alpha}_x), \tag{11}$$

$$\mathbf{N}_x^\top\mathbf{v}_x := \mathtt{ind}^{-1}_{\widehat{\mathbf{a}}_x}(\mathbf{F}_x^{-1}\mathbf{v}_x), \tag{12}$$

so that the matrix-vector products $\mathbf{N}_x\boldsymbol{\alpha}_x$ and $\mathbf{N}_x^\top\mathbf{v}_x$ can be evaluated again by $\mathcal{O}(n_x\log_2 n_x)$ flops.

Let us return to $\mathbf{A}$ of the form (5). Substituting (10) and the analogous formula for $\mathbf{A}_y$, we obtain after simple manipulations

$$\mathbf{A} = \mathbf{F}^{-1}\boldsymbol{\Lambda}\mathbf{F},$$

where $\mathbf{F} = \mathbf{F}_x \otimes \mathbf{F}_y$, $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}_x \otimes \mathbf{I}_y + \mathbf{I}_x \otimes \boldsymbol{\Lambda}_y$ with $\boldsymbol{\Lambda}_x = diag(\widehat{\mathbf{a}}_x)$, $\boldsymbol{\Lambda}_y = diag(\widehat{\mathbf{a}}_y)$ and $\widehat{\mathbf{a}}_x = \mathbf{F}_x\mathbf{a}_x$, $\widehat{\mathbf{a}}_y = \mathbf{F}_y\mathbf{a}_y$, respectively. Since $\mathbf{F}$ is the 2D DFT matrix and $\boldsymbol{\Lambda}$ is diagonal, the matrix-vector products $\mathbf{A}^\dagger\mathbf{v}$ can be evaluated by $\mathcal{O}(n\log_2 n)$ flops so that

$$\mathbf{A}^\dagger\mathbf{v} := \mathtt{ifft2d}(\boldsymbol{\Lambda}^\dagger\mathtt{fft2d}(\mathbf{v})),$$

where `fft2d` and `ifft2d` denote the 2D FFT and its inverse, respectively. Moreover, since $\mathbf{N} = \mathbf{N}_x \otimes \mathbf{N}_y$, formulas analogous to (11), (12) are valid and the matrix-vector products $\mathbf{N}\boldsymbol{\alpha}$ and $\mathbf{N}^\top\mathbf{v}$ can be performed again by means of $\mathcal{O}(n\log_2 n)$ flops.

Theorem 4.1 [7] ALGORITHM 2.1 *for solving (1) with* $\mathbf{A}$ *singular of the form (5) and with* $\mathbf{B}$ *sparse requires* $\mathcal{O}((l+2)mn\log_2 n)$ *flops.*

Theorem 4.2 ALGORITHM 2.1 *for solving (1) with* $\mathbf{A}$ *singular of the form (5) and with* $\mathbf{B}$ *sparse requires a memory space for (at most)* $3n+km$ *floating point numbers, where* k *denotes the maximal number of non-vanishing entries in rows of* $\mathbf{B}$.

Proof. Memory requirements are determined by $\mathbf{f}$, $\mathbf{B}$, $\mathbf{u}$ and by the diagonal of $\boldsymbol{\Lambda}^\dagger$ (it keeps information of $\mathbf{A}$). Since we assume $n \gg m \gg l$, the memory requirements of the other matrices and vectors are not significant. Moreover, these matrices and vectors can be stored in the memory reserved for $\mathbf{u}$ because $\mathbf{u}$ is computed in the last step of the algorithm. □

5. Numerical experiments

Let us consider the problem (4) with $\omega = \{(x,y) \in \mathbb{R}^2 : (x/0.2)^2 + (y/0.3)^2 \le 1\}$, $f(x,y) = 1$ on $\langle -0.5, 0.5\rangle \times \langle -0.5, 0.5\rangle$ and $f(x,y) = 0$ elsewhere, $g \equiv 0$ and $\Omega = \langle -1, 1\rangle \times \langle -1, 1\rangle$.

Numerical experiments are performed by Matlab 6 on Pentium(R)4, 3GHz with 512MB RAM; see Tab. 1. If $c = 0$, then $\mathbf{A}$ is singular with the defect $l = 1$ so that two runs of the conjugate gradient method in ALGORITHM 2.1 are computed. The relative tolerance terminating the conjugate gradient method is 10^{-4} in all cases.

		$c = 1$		$c = 0$	
n	m	time	CG steps	time	CG 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

Tab. 1: *CPU time (in seconds) and the conjugate gradients steps.*

References

[1] M. Fortin, R. Glowinski: *Augmented Lagrangian methods: Applications to the numerical solution of boundary-value problems*. Amsterdam, North-Holland 1983.

[2] F. Brezzi, M. Fortin: *Mixed and hybrid finite element methods*. Berlin, Springer-Verlag 1991.

[3] I. Daubechies: *Ten lectures on wavelets*. Philadelphia, SIAM 1992.

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

[5] G.H. Golub, C.F. Van Loan: *Matrix computations*, third edition. The Johns Hopkins Universtiy Press, Baltimore, 1996.

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

[7] R. Kučera: *Complexity of an algorithm for solving saddle-point systems with singular blocks arising in wavelet-Galerkin discretizations.* Accepted in Appl. Math., 2004.