UNIVERSITÄT SALZBURG

Fast Algorithm for the Fourth-Order Elliptic Problem Based on Orthogonal Matrix Decomposition

Paolo Di Stolfo Marián Vajteršic

Technical Report 2015-08 June 2015

Department of Computer Sciences

Jakob-Haringer-Straße 2
5020 Salzburg
Austria
www.cosy.sbg.ac.at

Technical Report Series

Fast Algorithm for the Fourth-Order Elliptic Problem Based on Orthogonal Matrix Decomposition

Paolo Di Stolfo[1] and Marián Vajteršic[2,3]

[1] Fachbereich Mathematik, Paris-Lodron-Universität Salzburg, Austria
paolo.distolfo@sbg.ac.at
[2] Fachbereich Computerwissenschaften, Paris-Lodron-Universität Salzburg, Austria
[3] Institute of Mathematics, Slovak Academy of Sciences, Bratislava, Slovakia
marian@cosy.sbg.ac.at

Abstract. A fast algorithm for solving the first biharmonic boundary problem on a rectangular domain is presented. It is based on splitting the fourth-order problem into a coupled system of two-order problems, whose finite-difference approximations are solved iteratively with linear algebra routines. There, a crucial role plays the orthogonal eigenvalue decomposition of the iteration matrix, which leads to a reduction of the operational count for one iteration to the asymptotically optimal value. This approach is extensively tested from the point of view of the total computational time, number of iterations and the solution error. It is shown, that these values cope with the theoretical assumptions and scale convincingly up to a problem with 100 million grid points.

1 Introduction

The biharmonic boundary value problem is of importance in several areas of computational practice, e.g. in aviation industry, civil engineering, etc. [8]. An example of a concrete problem is the deflection of a loaded plate. There is a number of numerical approaches developed for solving this problem. Most of them are based on its approximation by finite differences [8], finite elements [5] and most recently, by finite volumes [9]. Our attention will be devoted to the finite-difference approximations of a 2D problem, which lead either to one block five-diagonal linear system or to a split system of two block tri-diagonal systems, both with sparse and regularly structured matrices. The latter one arises by replacing the original problem by a coupled pair of discretized Poisson equations, which need to be solved iteratively. This is the so-called splitting principle (e.g. [1], [4], [7], [8]) and it has advantages to direct solvers in respect to the computational complexity and accuracy, particularly, when large grids are under consideration [3].

For a rectangular domain, covered by an $N \times N$ grid, the splitting approach requires in its original form $O(N^{5/2} \log^2 N)$ operations, when the second-order approximation to the Laplacian is considered and the accuracy of the solution

is set to $O(N^{-2})$ [4]. From this count, $O(N^{1/2} \log N)$ is the number of required iterations and $O(N^2 \log N)$ is the computational cost for one iteration. In paper [7], an algorithm was proposed, which reaches the optimal asymptotic complexity $O(N^2)$ for the evaluation of one iteration. This $O(\log N)$ improvement has been achieved by the exploitation of an orthogonal decomposition of the iteration matrix belonging to the splitting process. This gain was not paid off by an asymptotic increase of the number of iterations for the process with the transformed matrix. It remained unchanged, while the increase was only by a constant factor, due to a stronger convergence criterion.

In our previous work [7], this approach was tested on small grids only. A purpose of this work was to test it on modern computing platforms and to check whether the theoretical results will be confirmed also by computational experiments for large values of N. Our focus was particularly at the total computational time, the number of iterations needed and the accuracy of the obtained approximative solution.

The paper begins in Section 2 with a formulation of the problem, with its discretization and a description of the splitting approach. Section 3 presents the modified iteration process, where the orthogonal decomposition is exploited efficiently, together with the algorithm for its realization. The gravity point of our contribution brings Section 4, where the experiments are summarized and analyzed. Results for solving a problem on grids up to size 10000×10000 are presented. They manifest clear coincidence to theoretical estimations in all aspects under observation. The paper concludes with outlooks for further work.

2 Problem

The first boundary problem for the biharmonic equation (further, the biharmonic problem) on a rectangular region Ω is formulated as follows.

Find a solution $u \in C^4(\Omega) \cap C^1\left(\overline{\Omega}\right)$ with piecewise continuous second derivatives on the boundary $\partial\Omega$ of Ω such that

$$\begin{aligned} \Delta^2 u &= f \text{ in } \Omega \\ u &= g_1 \text{ on } \partial\Omega \\ u_n &= g_2 \text{ on } \partial\Omega \ , \end{aligned} \tag{1}$$

where the term u_n denotes the derivative of u in the direction of the outer normal.

The biharmonic equation (1) can be replaced by a coupled pair of second-order boundary value problems, introducing an unknown function v and a parameter $\gamma \neq 0$ [6]:

$$\begin{aligned} \Delta u &= v && \text{in } \Omega \\ u &= g_1 && \text{on } \partial\Omega \\ \Delta v &= f && \text{in } \Omega \\ v &= \Delta u - \gamma(u_n - g_2) && \text{on } \partial\Omega \ . \end{aligned}$$

Iterative schemes that make use of this replacement are called splitting methods. A possible iterative scheme for the above system of equations introduces new unknown functions U and V that incorporate a smoothing process for u and v:

$$\begin{aligned} u^{(t+1)} &= g_1 && \text{on } \partial\Omega \\ \Delta u^{(t+1)} &= V^{(t)} && \text{in } \Omega \\ U^{(t+1)} &= (1-\omega_1)U^{(t)} + \omega_1 u^{(t+1)} && \text{in } \Omega \\ v^{(t+1)} &= \Delta U^{(t+1)} - \gamma\big(\partial_n U^{(t+1)} - g_2\big) && \text{on } \partial\Omega \\ \Delta v^{(t+1)} &= f && \text{in } \Omega \\ V^{(t+1)} &= (1-\omega_2)V^{(t)} + \omega_2 v^{(t+1)} && \text{in } \Omega \ . \end{aligned} \tag{2}$$

For this scheme, starting values $U^{(0)}$, $V^{(0)}$, and suitable smoothing parameters ω_1, ω_2 have to be defined. Given suitable smoothing parameters and $\gamma \neq 0$, the scheme converges to the solution of the biharmonic equation (1) (see e.g. [6]).

In our further explanations, the rectangular region Ω will be simplified to the unit square $[0,1]^2$. Using the mesh size $h = 1/(N+1)$ for some positive integer N, $\Omega \cup \partial\Omega$ is replaced by the set $\Omega_h \cup \partial\Omega_h$ of grid points. The discrete function $U^{(t+1)}$ on Ω_h is given as $U_{i,j}^{(t+1)} = U^{(t+1)}(ih, jh)$ for $1 \le i, j \le N$. The values of $U^{(t+1)}$ are arranged as a vector $U^{(t+1)} = \left(U_1^{(t+1)}, \ldots, U_N^{(t+1)}\right)^\top$, where $U_j^{(t+1)} = \left(U_{j,1}^{(t+1)}, \ldots, U_{j,N}^{(t+1)}\right)^\top$.

Let us consider the discretized version of the functions u, U, v, V on $\Omega_h \cup \partial\Omega_h$ and discrete analogues for Δ, ∂_n in (2) by applying the standard five-point difference approximation for the Laplacian. This yields the discretization error of $O\big(N^{-3/2}\big)$ for sufficiently large N. From these approximations, the following iterative system studied by Ehrlich [4] can be derived:

$$\begin{aligned} Gu^{(t+1)} &= -h^2 V^{(t)} + b \\ U^{(t+1)} &= (1-\omega_1)U^{(t)} + \omega_1 u^{(t+1)} \\ Gv^{(t+1)} &= \tfrac{2}{h^2} M U^{(t+1)} + c \\ V^{(t+1)} &= (1-\omega_2)V^{(t)} + \omega_2 v^{(t+1)} \ . \end{aligned} \tag{3}$$

Here, G denotes the block-tridiagonal matrix $G = (-I, A, -I) \in \mathbb{R}^{N^2 \times N^2}$, where $A = (-1, 4, -1) \in \mathbb{R}^{N \times N}$ is a tridiagonal matrix. Moreover, the matrix $M = (I + D, D, \ldots, D, I + D) \in \mathbb{R}^{N^2 \times N^2}$ is block-diagonal, where $D = \mathrm{diag}(1, 0, \ldots, 0, 1) \in \mathbb{R}^{N \times N}$ and $I \in \mathbb{R}^{N \times N}$ is the identity matrix. The vectors b and c are constant and depend only on the boundary conditions and the right-hand side.

3 Algorithm

Instead of iterating the scheme in (3), the equations may be combined into a single three-term recursion formula for $U^{(t+1)}$, eliminating the vectors $u^{(t)}, v^{(t)}$,

and $V^{(t)}$ [7]:

$$U^{(t+1)} = (2-\omega_1-\omega_2)U^{(t)} - 2\omega_1\omega_2 G^{-2}\big(MU^{(t)}\big) - (1-\omega_1)(1-\omega_2)U^{(t-1)} + \omega_1\omega_2 d \ , \tag{4}$$

where $U^{(0)} = O$, $U^{(1)} = \omega_1 G^{-1} b$, and $d = G^{-1}b - h^2 G^{-2} c$.

As the computation of $G^{-2}MU^{(t)}$ in (4) is the most expensive step, it requires special attention and is handled using the matrix decomposition (MD) algorithm which takes advantage of the eigendecomposition of a symmetric matrix [2].

The matrix $A \in \mathbb{R}^{N\times N}$ is real and symmetric and, therefore, can be factored into $A = V^T \Lambda V$, where V is an orthogonal matrix. The diagonal values of $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ are the eigenvalues of A, i.e.,

$$\lambda_j = 4 - 2\cos\frac{j\pi}{N+1} \ , \text{ for } j = 1, \ldots, N \ .$$

The corresponding eigenvectors form the columns of the symmetric matrix $V = (V_{i,j}) \in \mathbb{R}^{N\times N}$, where

$$V_{i,j} = \sqrt{\frac{2}{N+1}} \sin\frac{ij\pi}{N+1} \ , \text{ for } i,j = 1, \ldots, N \ . \tag{5}$$

In the following, we apply the MD algorithm to compute $y = G^{-2}z$ with $z = MU^{(t)}$ and include the computational complexity of each step:

1. Compute the vectors $\widetilde{z}_j = V z_j$, for $j = 1, \ldots, N$. Due to the sparse structure of z, this requires only $O\big(N^2\big)$ operations in total.
2. Create the column vectors $\hat{z}_i$, i.e., taking the ith column of $\widetilde{z}$.
3. Solve the systems $M_i \hat{y}_i = \hat{z}_i$, for $i = 1, \ldots, N$, where $M_i = (-1, \lambda_j, -1) \in \mathbb{R}^{N\times N}$. Each tridiagonal system may be solved in $O(N)$ operations. Thus, this step takes $O\big(N^2\big)$ operations in total.
4. Solve the systems $M_i \hat{\hat{y}}_i = \hat{y}_i$, $i = 1, \ldots, N$ in $O\big(N^2\big)$ operations.
5. Create the vectors $\widetilde{\widetilde{y}}_j$, $j = 1, \ldots, N$ that are the rows of $\widetilde{\widetilde{y}}$ by using $\hat{\hat{y}}_i$ as their columns.
6. Compute the solution vectors $y_j = V\widetilde{\widetilde{y}}_j$, $j = 1, \ldots, N$. Note that y_j are dense vectors in general. Using the matrix $F = \mathrm{diag}(V, \ldots, V) \in \mathbb{R}^{N^2\times N^2}$, one may also write $\widetilde{\widetilde{y}} = Fy$. While the classical matrix-vector multiplication takes $O\big(N^3\big)$ steps, the FFT algorithm reduces this number to $O\big(N^2 \log N\big)$.

Note that the vectors $\widetilde{b} = FG^{-1}b$ and $\widetilde{\widetilde{c}} = FG^{-2}c$ may be computed analogously using the above algorithm. For $\widetilde{b}$, steps 1–3 need to be performed using b instead of $MU^{(t)}$, for $\widetilde{\widetilde{c}}$, steps 1–5 need to be performed using c instead of $MU^{(t)}$.

We refer to the combination of (4) with the above steps for computing $G^{-2}MU^{(t)}$ as modified Ehrlich's algorithm (*ME*). In total, the ME algorithm costs $O\big(N^2 \log N\big)$ computational steps per iteration. To improve the computational complexity of each iteration to an asymptotically optimal value of $O\big(N^2\big)$,

the final multiplication of F in step 6 needs to be eliminated. Therefore, a new iteration sequence $\overline{U}^{(t+1)}$ is defined by $\overline{U}^{(t+1)} = \frac{1}{\omega_1} F U^{(t+1)}$. Premultiplying the recurrence formula for $U^{(t+1)}$ by $\frac{1}{\omega_1} F$, one gets the equivalent formula

$$\overline{U}^{(t+1)} = (2-\omega_1-\omega_2)\overline{U}^{(t)} - 2\omega_2 F G^{-2}\big(MU^{(t)}\big) - (1-\omega_1)(1-\omega_2)\overline{U}^{(t-1)} + \omega_2 F d \ , \tag{6}$$

in which $\overline{U}^{(0)} = 0$, $\overline{U}^{(1)} = \widetilde{b}$, and $\widetilde{\widetilde{d}} = \widetilde{b} + h^2 \widetilde{\widetilde{c}}$. Since $\widetilde{\widetilde{d}}$ is assumed to be constant, each iteration can be performed in $O\big(N^2\big)$ operations. We refer to the combination of (6) and the steps 1–5 from above as the algorithm *FV*.

As a test for convergence, one may check for a given $\epsilon > 0$ whether

$$\max_{i,j} \left| U_{i,j}^{(t+1)} - U_{i,j}^{(t)} \right| < \epsilon \tag{7a}$$

for ME, and

$$\max_{i,j} \left| \overline{U}_{i,j}^{(t+1)} - \overline{U}_{i,j}^{(t)} \right| < \frac{\epsilon}{\|V\|_\infty \, \omega_1} \tag{7b}$$

for FV. Since the iteration matrices corresponding to the algorithms ME and FV are similar, the same optimal smoothing parameters $\omega_1 = \omega_2 = O\big(N^{-1/2}\big)$ as developed in [4] can be used. From this, one gets the same rate of convergence $R_\infty = O\big(N^{-1/2}\big)$ for FV, so that the number of iterations required for $\epsilon = O\big(h^2\big)$ is

$$\log \frac{\omega_1 \, \|V\|_\infty}{\epsilon} \frac{1}{R_\infty} = O\Big(N^{1/2} \log N\Big). \tag{8}$$

Due to the stronger convergence criterion (7b), FV may need a higher number of iterations than ME up to a constant multiple.

In total, the computational complexity amounts to $O\big(N^{5/2} \log N\big)$ for FV and to $O\big(N^{5/2} \log^2 N\big)$ for the ME algorithm (called *ME-FFT* in short) when $\epsilon = O\big(h^2\big)$. When using classical matrix-vector multiplications as a replacement for the FFT, the computational complexity is $O\big(N^{7/2} \log N\big)$. We refer to this variant as *ME-Matmul*.

4 Experiments

In this section, we study the performance of the algorithm FV with various mesh sizes h. Main measures of interest are the total running time and the resulting error. From this perspective, we also compare it to ME.

4.1 Experimental Setup

The algorithm FV (explained in detail in the previous section) was implemented in the C programming language. The ME algorithm was implemented in the variants ME-FFT and ME-Matmul. Experiments comparing all three algorithms are described in Sect. 4.2.

The computations were performed on the Doppler cluster at the University of Salzburg that runs mostly AMD Opteron processors of clock rates between 2.2 GHz and 2.8 GHz. (Clearly, comparative experiments were conducted on nodes having the same clock rates.)

As an experimental problem, we use the biharmonic equation (1) with the solution (see e.g. [7])

$$u(x, y) = x^3 - 3y^2 + 2xy \text{ on } [0, 1]^2 \ . \tag{9}$$

All figures and tables presented in this section are related to problem (1) with solution (9). The parameter N in figures ranges from 7 to 5000 and the logarithmic scale is used on both axes.

4.2 Performance

In this section, detailed results for running times are given. Firstly, the relationship between the running times for the FV algorithm and its computational complexity is investigated. Later on, FV is compared to the two variants ME-FFT and ME-Matmul of the modified Ehrlich's algorithm.

Total Computational Time. For sizes $N \leq 31$, the problem (1) with solution (9) was analyzed in [7]. These results are reproduced and results for larger-sized systems are given in the following.

As a tolerance, the typical choice of $\epsilon = h^2$ is used. Fig. 1 depicts the running time for several problem sizes confirming the expected complexity of $O\big(N^{5/2} \log N\big)$. The constant for the dominant complexity term, which results from our experiments, can be taken equal to 10^{-6}.

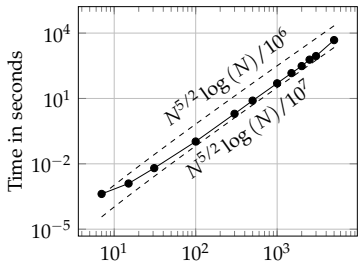

Fig. 1. Running times for $\epsilon = h^2$. The actual running time (solid line) is compared to the theoretical complexity estimation with the constants 10^{-6} and 10^{-7} (dashed lines).

Number of Iterations. In order to achieve a factor of error reduction β, the expected number of iterations is $t(N) \geq \frac{\log 1/\beta}{\log 1/\left(1-\sqrt{h}\right)}$, where $h = \frac{1}{N+1}$ [3]. In

comparison to the actual number of iterations used, the curve $t(N)$ is plotted in Fig. 2. The curve $t(N)$ conforms to the actual number of iterations, slightly overestimating it for smaller N. In addition, the curve $N^{1/2}\log N$ is plotted which is an approximation to $t(N)$ in this case. For a tolerance $\epsilon = h^2 \approx N^{-2}$, the number of iterations is at least $2N^{1/2}\log N$. In the present case, this is a fairly good approximation for the actual number of iterations. Moreover, the approximation $2N^{1/2}\log N$ can be shown to be almost identical to the curve $t(N)$ for larger N. Thus, it seems that the estimate $O\big(N^{1/2}\log N\big)$ might be satisfactory in practice.

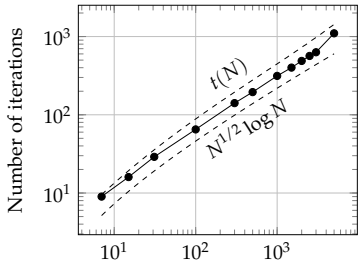

Fig. 2. The number of iterations for $\epsilon = h^2$ (solid line). The dashed curves $t(N)$ and $N^{1/2}\log N$ depict estimates for the number of iterations.

Comparison of FV with ME-FFT and ME-Matmul.

- FV: This algorithm has a complexity of $O\big(N^2\big)$ per iteration, but a possibly higher number of iterations than the variants ME-FFT and ME-Matmul due to different criteria for convergence (see Sect. 2).
- ME-FFT: This variant, as described in (4), performs with a complexity of $O\big(N^2\log N\big)$ per iteration.
- ME-Matmul: Instead of optimizing the matrix-vector multiplications by means of FFT, the classical algorithm is used. Its drawback is the higher complexity of $O\big(N^3\big)$ per iteration.

For our comparisons, the same problem (1) for (9) is solved for $\epsilon = h^2$ by all three variants. We note that for achieving this demanded tolerance, the number of iterations is $O\big(N^{1/2}\log N\big)$. Therefore, the total computational complexity amounts to $O\big(N^{5/2}\log N\big)$ for FV, to $O\big(N^{5/2}\log^2 N\big)$ for ME-FFT, and to $O\big(N^{7/2}\log N\big)$ for ME-Matmul. The actual results for these algorithms are depicted in Fig. 3.

From Fig. 3, it is evident that the running times of FV and ME-FFT follow similar patterns since their theoretical computational complexities differ only by a factor of $O(\log N)$. When comparing FV and ME-FFT, the possibly larger

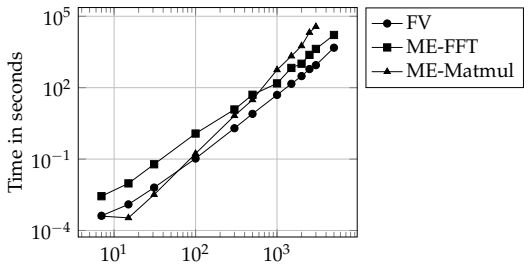

Fig. 3. The total running time of each of the three algorithms for $\epsilon = h^2$

number of iterations performed by FV has to be taken into account. Nonetheless, even for large problem sizes, FV is approximately three times faster than ME-FFT.

Table 1 contains a detailed list of results per algorithm for some representative sizes of N. Two immediate conclusions can be drawn from the figures. Firstly, the number of iterations for FV is indeed higher than the number of iterations for ME-FFT due to the more strict termination criterion, which may cause an iteration count increased by a constant factor. Secondly, although competitive for small N, the variant ME-Matmul using classical matrix-vector multiplications becomes infeasible for large N.

Table 1. The number of iterations and running times for $\epsilon = h^2$

	FV		ME-FFT		ME-Matmul	
N	Iterations	Time [s]	Iterations	Time [s]	Iterations	Time [s]
7	9	0.0004	7	0.0028	7	0.0004
31	29	0.0064	21	0.0605	21	0.0032
500	195	7.9508	152	50.4387	152	30.7433
1,000	314	49.5345	239	151.2021	239	560.1155
2,000	488	306.3109	369	1,015.1991	369	5,689.0126
3,000	630	883.7871	488	4,248.0112	488	36,534.9337
5,000	1,240	4,778.6709	662	16,376.5583	–	–

For a second series of tests, the same setups are used as before except the tolerance ϵ is fixed at $\epsilon = 10^{-6}$. Table 2 lists corresponding results for representative values of N. In comparison to the results for $\epsilon = h^2$, it can be seen that for smaller N, the number of iterations is higher. It is because for small N, the tolerance h^2 is larger than 10^{-6} and thus, more iterations have to be performed until an iteration error of 10^{-6} is reached. However, starting at a size of $N = 1000$, the solution takes less time for $\epsilon = 10^{-6}$. Additionally to Tab. 1,

results for $N = 10000$ are provided, showing that FV is still faster than ME-FFT for large systems (in this case, by a factor of approximately 5).

Table 2. The number of iterations and running times for the tolerance $\epsilon = 10^{-6}$

	FV		ME-FFT		ME-Matmul	
N	Iterations	Time [s]	Iterations	Time [s]	Iterations	Time [s]
7	18	0.0005	17	0.0061	17	0.0002
31	42	0.0079	37	0.1038	37	0.0075
500	216	8.9500	162	53.6371	162	41.7075
1,000	314	56.0913	239	152.0935	239	537.9254
2,000	466	294.5305	336	922.8362	336	6,769.0631
3,000	575	1,122.8605	423	3,678.5816	423	28,381.2390
5,000	791	3,436.3500	575	13,916.6963	–	–
10,000	1,164	19,156.2446	824	106,102.6260	–	–

4.3 Errors

In this subsection, the error values related to FV are examined. These are the termination error, given as the maximum norm of the difference of two consecutive iterations defined by (7b), and the solution error, which is the maximum norm of the difference between the computed solution and the actual solution. It is known that the approximations used yield a discretization error of $O\big(N^{-3/2}\big)$. Thus, if the actual error is below the discretization error, the solution is sufficiently accurate.

In the following plots and tables, the same set-up configurations are used as in the previous subsection, which was devoted to examinations of running times. In Fig. 4, a comparison of the actual error to the function $N^{-3/2}$ is plotted.

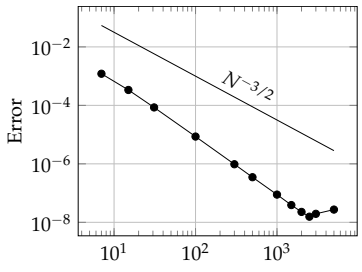

Fig. 4. The solution error compared to the theoretical estimate $N^{-3/2}$ with $\epsilon = 10^{-6}$

Figure 4 shows, that the error caused by the approximation attains the order of the discretization error, as expected. For the sake of completeness, the termination and solution errors are listed in Tab. 3.

Table 3. The termination errors and solution errors for $\epsilon = 10^{-6}$

N	Termination error	Solution error
7	0.00000040	0.00120771
31	0.00000042	0.00008502
500	0.00000031	0.00000035
1,000	0.00000025	0.00000009
2,000	0.00000030	0.00000002
3,000	0.00000030	0.00000002
5,000	0.00000038	0.00000003

5 Conclusions and Outlooks

An efficient algorithm for solving the model biharmonic problem has been presented. It has been shown, that it is possible to get a numerical solution on an $N \times N$ grid with the accuracy of $O(N^{-2})$ in a time proportional to $O(N^{5/2} \log^2 N)$. It was experimentally confirmed that the actual times for N ranging from 7 up to 5000 scale perfectly with this estimate and lie in a corridor marked by lines, which correspond to the dominant complexity term with constants 10^{-6} and 10^{-7} respectively. Similarly, the estimation $O(N^{1/2} \log N)$ for the number of required iterations coincides in a scalable way with the reality, whereby the absolute error of the solution is always below the discretization error.

A comparison of this algorithm with a modification of the original approach of Ehrlich [4] confirms, that the estimated theoretical $O(\log N)$ gain in total time is valid also for large values of N, despite the number of iterations needed is slightly higher. Experiments for N up to 10000 show that against Ehrlich's approach with FFT (used for matrix-multiplication) the presented algorithm needs up to 5-times less total time and when the product of matrices is computed classically, then this factor is even higher, making this approach for large parameters on serial computer not feasible.

There is a space left for further research. Another interesting comparison of FV against the modification of Ehrlich's approach offers an application of a multigrid Poisson solver, because it would also lead to the optimal complexity count for computing one iteration. While our presentation is restricted to $N \times N$ grids on a rectangle, a further extension would encompass grids with different number of points in each direction and also the case of non-rectangular domains. Here, the estimation of smoothing parameters would be an important issue. The

algorithm bears potential for a parallelization. Therefore, the investigation of its behavior when examined on large HPC platforms is also an open question.

Acknowledgments. The second author was supported by the VEGA grant no. 2/0026/14.

References

1. Bialecki, B.: A fourth order finite difference method for the Dirichlet biharmonic problem. Numerical Algorithms, 61.3 (2012): 351–375.
2. Buzbee, B. L., Golub G. H., and Nielson, C. W.: On direct methods for solving Poisson's equations. SIAM Journal on Numerical analysis 7.4 (1970): 627-656.
3. Di Stolfo, P.: Splitting-based fast algorithm for solving the rectangular biharmonic problem: A sequential implementation. Master's Thesis, Paris-Lodron-Universität Salzburg (2015).
4. Ehrlich, L. W.: Solving the biharmonic equation as coupled finite difference equations. SIAM Journal on Numerical Analysis 8.2 (1971): 278–287.
5. Falk, R. S.: Approximation of the biharmonic equation by a mixed finite element method. SIAM Journal on Numerical Analysis, 15.3 (1978): 556–567.
6. McLaurin, J. W.: A general coupled equation approach for solving the biharmonic boundary value problem. SIAM Journal on Numerical Analysis 11.1 (1974): 14-33.
7. Vajteršic, M.: A fast algorithm for solving the first biharmonic boundary value problem. Computing 23.2 (1979): 171–178.
8. Vajteršic, M.: Algorithms for Elliptic Problems. Kluwer Academic Publisher, Dordrecht (1993).
9. Wang, T.: A mixed finite volume element method based on rectangular mesh for biharmonic equations, J. Comput. Appl. Math., 172.1 (2004): 117–130.