Efficient computation of matrix power-vector products: application for space-fractional diffusion problems

Ferenc Izsák[a,*], Béla J. Szekeres[b]

[a]*Department of Applied Analysis and Computational Mathematics, & MTA ELTE NumNet Research Group, Eötvös Loránd University, Pázmány P. stny. 1C, 1117 - Budapest, Hungary*
[b]*Department of Numerical Analysis, Eötvös Loránd University, Faculty of Informatics, Pázmány P. stny. 1C, 1117 - Budapest, Hungary*

Abstract

A novel algorithm is proposed for computing matrix-vector products $A^\alpha \mathbf{v}$, where A is a symmetric positive semidefinite sparse matrix and $\alpha > 0$. The method can be applied for the efficient implementation of the matrix transformation method to solve space-fractional diffusion problems. The performance of the new algorithm is studied in a comparison with the conventional MATLAB subroutines to compute matrix powers.

Keywords: matrix powers, matrix transformation method, binomial series, space-fractional diffusion
2010 MSC: 65N12, 65N15, 65N30

1. Introduction

In the last two decades, the numerical simulation of (space-) fractional diffusion became an important topic in the numerical PDE's starting with the poineering paper [1]. Fractional dynamics was observed in a wide range of life sciences [2], earth sciences [3] and financial processes [4]. A frequently used and meaningful mathematical model [5] of these phenomena is offered by the fractional Laplacian operator [6]. For solving these problems numerically, the so-called matrix transformation method was proposed in [7] and [8]. Accordingly, a mathematical analysis was presented to prove the convergence of this simple approach in case of finite difference [9] and finite element discretization.

The bottleneck of the matrix transformation approach is the computation of the fractional power of the matrices arising from the finite element or the finite difference discretization of the negative Laplacian operators. This is why many authors use a more technical approach [6].

Several algorithms were proposed to compute this matrix power. We refer to the latest approach in [10] and its reference list on the earlier developments. Due to its importance, MATLAB has also a built-in subroutine `mpower.m`.

To bypass this costly procedure, the aim of the present article is to develop a simple and fast numerical method to compute matrix-vector products of type $A^\alpha \mathbf{v}$. This idea was also used in [11] by developing the frequently used Matlab subroutine `expmv.m` to compute matrix exponential-vector products.

1.1. Mathematical preliminaries

The main motivation of our study is the efficient numerical solution of the initial-boundary value problem

$$\begin{cases} \partial_t u(t,\mathbf{x}) = -(-\Delta)^\alpha u(t,\mathbf{x}) \quad t \in (0,T),\ \mathbf{x} \in \Omega \\ u(0,\mathbf{x}) = u_0(\mathbf{x}) \quad t \in (0,T),\ \mathbf{x} \in \Omega, \end{cases} \tag{1}$$

where $\Delta = \Delta_{\mathcal{D}}$ or $\Delta = \Delta_{\mathcal{N}}$ denotes the Dirichlet or the Neumann Laplacian on the bounded Lipschitz domain Ω and $u_0 \in L_2(\Omega)$ is given. Note that $-\Delta_{\mathcal{D}}^{-1} : L_2(\Omega) \to L_2(\Omega)$ and $-\Delta_{\mathcal{N}}^{-1} : L_2(\Omega)/\mathbb{R} \to L_2(\Omega)$ are compact and self-adjoint, so that its fractional power makes sense.

*Corresponding author
Email addresses: `izsakf@cs.elte.hu` (Ferenc Izsák), `szbpagt@cs.elte.hu` (Béla J. Szekeres)

According to the *matrix transformation method*, (1) can be semidiscretized as

$$\partial_t \mathbf{u}(t) = -(A)^\alpha \mathbf{u}(t) \quad t \in (0,T) \tag{2}$$

where A is the discretization of the operator $-\Delta_{\mathcal{D}}$ or $-\Delta_{\mathcal{N}}$ either with finite differences or with finite elements.

Accordingly, $A \in \mathbb{R}^{n\times n}$ is a symmetric positive semidefinite sparse matrix, $\mathbf{v} \in \mathbb{R}^n$ is an arbitrary vector and we consider the case of $\alpha \in (0,1)$. The computational challenge is then to compute the products $A^\alpha \mathbf{v}$, when the solution of (2) is approximated with some explicit time stepping.

If the spectral decomposition of A is known, such that $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ are the eigenvectors and $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ the corresponding eigenvalues of A then A^α has the same eigenvectors, while the corresponding eigenvalues are $\lambda_1^\alpha, \lambda_2^\alpha, \dots, \lambda_n^\alpha$. Accordingly, if $\mathbf{v} = a_1\mathbf{x}_1 + a_2\mathbf{v}_2 + \cdots + a_n\mathbf{x}_n$ then

$$A^\alpha \mathbf{v} = a_1\lambda_1^\alpha \mathbf{x}_1 + a_2\lambda_2^\alpha \mathbf{v}_2 + \cdots + a_n\lambda_n^\alpha \mathbf{x}_n. \tag{3}$$

2. Main results

2.1. Motivation and main idea

For the first glance, the easiest way to approximate the matrix power A^α is to truncate the power series

$$A^\alpha = (I + A - I)^\alpha = \sum_{j=0}^{\infty} \binom{\alpha}{j} (A-I)^j,$$

which is satisfied for $\alpha > 0$ if $\sigma(A-I) \le 1$, where σ denotes the spectral radius. To ensure this, we rather compute

$$A^\alpha = \left(\frac{\sigma(A)}{2}\right)^\alpha \left(\frac{2A}{\sigma(A)}\right)^\alpha = \left(\frac{\sigma(A)}{2}\right)^\alpha \sum_{j=0}^{\infty} \binom{\alpha}{j} \left(\frac{2A}{\sigma(A)} - I\right)^j. \tag{4}$$

Using the relation $A \ge 0$, for an arbitrary eigenvalue $\tilde{\lambda}$ of $\frac{2A}{\sigma(A)} - I$ we have that $-1 =\le \tilde{\lambda} \le \frac{\sigma(2A)}{\sigma(A)} - 1 = 1$, so that the binomial series in (4) converges for non-singular matrices A. The convergence, however, is very slow due to the smallest and largest eigenvalues of A, which are transformed near to -1 and 1, respectively.

The basic idea is then to apply a spectral decomposition to A: in the subspace of $\mathbb{R}^n$ corresponding to the smallest and largest eigenvalues of A, we apply the direct spectral definition in (3), while in the orthocomplement, the expansion (4) will converge satisfactorily.

2.2. Method

The decomposition can be formulated by introducing an orthogonal projection $Q : \mathbb{R}^n \to \mathbb{R}^k$, which maps to the subspace span $\left\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_j, \mathbf{x}_{n-(k-j-1)}, \dots, \mathbf{x}_{n-1}, \mathbf{x}_n\right\}$ with $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_j \le \lambda_{n-(k-j-1)} \le \cdots \le \lambda_n$, where j and k are some parameters. Also, henceforth we focus to the computation of $A^\alpha\mathbf{v}$, which can be given as

$$A^\alpha \mathbf{v} = \left(\frac{\sigma(A)}{2}\right)^\alpha \left(\frac{2A}{\sigma(A)}\right)^\alpha \mathbf{v} = \left(\frac{\sigma(A)}{2}\right)^\alpha \left(\left(\frac{2A}{\sigma(A)}\right)^\alpha Q\mathbf{v} + \left(\frac{2A}{\sigma(A)}\right)^\alpha (\mathbf{v} - Q\mathbf{v})\right). \tag{5}$$

2.2.1. Computation of the first term in (5)

The projection matrix Q in (5) is defined with $Q = XX^T$ and the projection $Q\mathbf{v}$ can be given as

$$Q\mathbf{v} = XX^T\mathbf{v} = a_1\mathbf{x}_1 + \cdots + a_j\mathbf{x}_j + a_{n-(k-j-1)}\mathbf{x}_{n-(k-j-1)} + \cdots + a_n\mathbf{x}_n \tag{6}$$

with some coefficients $a_1, a_2, \dots, a_n$, where the matrix

$$X = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_j, \mathbf{x}_{n-(k-j-1)}, \dots, \mathbf{x}_{n-1}, \mathbf{x}_n] \in \mathbb{R}^{n\times k} \tag{7}$$

is composed from the eigenvectors corresponding to the largest and smallest eigenvalues of A. Observe that, according to the spectral definition of the matrix power in (3), we have

$$A^\alpha Q\mathbf{v} = \lambda_1^\alpha a_1\mathbf{x}_1 + \cdots + \lambda_j^\alpha a_j\mathbf{x}_j + \lambda_{n-(k-j-1)}^\alpha a_{n-(k-j-1)}\mathbf{x}_{n-(k-j-1)} + \cdots + \lambda_n^\alpha a_n\mathbf{x}_n = X\Lambda X^T\mathbf{v}, \tag{8}$$

where Λ is a diagonal matrix with the entries $\lambda_1, \dots, \lambda_j, \lambda_{n-(k-j-1)}, \dots, \lambda_n$. This gives the first term in (5).

2.2.2. Computation of the second term in (5)

We should simply apply here (4) to get

$$\left(\frac{2A}{\sigma(A)}\right)^{\alpha}(\mathbf{v}-Q\mathbf{v})=\sum_{n=0}^{\infty}\binom{\alpha}{n}\left(\frac{2A}{\sigma(A)}-I\right)^{n}(\mathbf{v}-Q\mathbf{v})\approx\sum_{n=0}^{K}\binom{\alpha}{n}\left(\frac{2A}{\sigma(A)}-I\right)^{n}(\mathbf{v}-Q\mathbf{v}), \tag{9}$$

where the vector $Q\mathbf{v}$ was already computed in (6). To accelerate the computation in (9) further, we observe that

$$\binom{\alpha}{n+1}\left(\frac{2A}{\sigma(A)}-I\right)^{n+1}(\mathbf{v}-Q\mathbf{v})=\frac{\alpha-j}{n+1}\left(\frac{2A}{\sigma(A)}-I\right)\cdot\left[\binom{\alpha}{n}\left(\frac{2A}{\sigma(A)}-I\right)^{n}(\mathbf{v}-Q\mathbf{v})\right],$$

i.e., a new term in (9) can be computed using only one sparse matrix-vector multiplication.

2.3. Analysis: Estimation for the number of terms in the Taylor expansion (9)

The accuracy of the approximation in (9) will be given using the error indicator

$$\mu_{\max}=\left\|\left(\frac{2A}{\sigma(A)}-I\right)(I-Q)\right\|=\max\left\{\left|\frac{2\lambda_{j+1}}{\sigma(A)}-1\right|;\left|\frac{2\lambda_{n-k+j}}{\sigma(A)}-1\right|\right\}=\max\left\{1-\frac{2\lambda_{j+1}}{\sigma(A)};\frac{2\lambda_{n-k+j}}{\sigma(A)}-1\right\},$$

which measures the error of the truncation in (9) and can be decreased by choosing suitable (large enough) parameters j and k.

Theorem 1. *Assume that the eigenvalues for X in (7) and the eigenvectors for Λ in (8) are computed exactly. Then an upper bound for computational error in $A^{\alpha}v$ is the term*

$$\alpha\left(\frac{\sigma(A)}{2}\right)^{\alpha}\frac{\exp\left(-(K+1)(1-\mu_{\max})\right)}{(K+1)(1-\mu_{\max})}\|\mathbf{v}\|.$$

In practice, to ensure a precision ε of $A^{\alpha}\mathbf{v}$, the number K of the terms in (9) *should satisfy*

$$K\leq\frac{1}{1-\mu_{\max}}\mathcal{G}\left(\frac{\varepsilon}{\alpha}\left(\frac{2}{\sigma(A)}\right)^{\alpha}\right)-1,$$

where $\mathcal{G}$ is the inverse of the function $\mathbb{R}^{+}\ni s\mapsto\frac{e^{-s}}{s}$.

Proof: The computational error in $A^{\alpha}\mathbf{v}$ can be expressed in the following form.

$$\begin{aligned}&A^{\alpha}\mathbf{v}-\left(A^{\alpha}Q\mathbf{v}+\left(\frac{\sigma(A)}{2}\right)^{\alpha}\sum_{j=0}^{K}\binom{\alpha}{j}\left(\frac{2A}{\sigma(A)}-I\right)^{j}(\mathbf{v}-Q\mathbf{v})\right)\\&=\left(A^{\alpha}-\left(\frac{\sigma(A)}{2}\right)^{\alpha}\sum_{j=0}^{K}\binom{\alpha}{j}\left(\frac{2A}{\sigma(A)}-I\right)^{j}\right)(\mathbf{v}-Q\mathbf{v})=\left(\frac{\sigma(A)}{2}\right)^{\alpha}\sum_{j=K+1}^{\infty}\binom{\alpha}{j}\left(\frac{2A}{\sigma(A)}-I\right)^{j}(I-Q)\mathbf{v}.\end{aligned} \tag{10}$$

Therefore, we first derive an upper bound for the matrix series in the last line. Observe that

$$\left(\frac{2A}{\sigma(A)}-I\right)^{j}(I-Q)\mathbf{v}=\left(\frac{2A}{\sigma(A)}-I\right)^{j}(I-Q)^{j}\mathbf{v}=\left(\left(\frac{2A}{\sigma(A)}-I\right)(I-Q)\right)^{j}\mathbf{v}. \tag{11}$$

Since we use symmetric matrices, the matrix norm $\|\cdot\|$ in the following refers both to the spectral norm and the l_2-norm. Since $0<\alpha<1$, we easily get the inequality

$$\left|\binom{\alpha}{j}\right|=\left|\binom{\alpha}{j-1}\right|\left|\frac{\alpha-j+1}{j}\right|\leq\left|\binom{\alpha}{j-1}\right|\frac{j-1}{j}\leq\left|\binom{\alpha}{j-2}\right|\frac{j-1}{j}\frac{j-2}{j-1}\leq\cdots\leq\left|\binom{\alpha}{0}\right|\frac{\alpha}{j}=\frac{\alpha}{j},$$

which can be used in (10) with (11) to get the following estimate

$$\begin{aligned}
&\left(\frac{\sigma(A)}{2}\right)^{\alpha}\left\|\sum_{j=K+1}^{\infty}\binom{\alpha}{j}\left(\frac{2A}{\sigma(A)}-I\right)^{j}(I-Q)\right\| \leq \left(\frac{\sigma(A)}{2}\right)^{\alpha}\frac{\alpha}{K+1}\left\|\sum_{j=K+1}^{\infty}\left(\frac{2A}{\sigma(A)}-I\right)^{j}(I-Q)\right\| \\
&\leq \left(\frac{\sigma(A)}{2}\right)^{\alpha}\frac{\alpha}{K+1}\sum_{j=K+1}^{\infty}\left\|\left(\frac{2A}{\sigma(A)}-I\right)(I-Q)\right\|^{j} \leq \left(\frac{\sigma(A)}{2}\right)^{\alpha}\frac{\alpha}{K+1}\sum_{j=K+1}^{\infty}\mu_{\max}^{j} \\
&= \left(\frac{\sigma(A)}{2}\right)^{\alpha}\frac{\alpha}{K+1}\cdot\frac{\mu_{\max}^{K+1}}{1-\mu_{\max}} = \left(\frac{\sigma(A)}{2}\right)^{\alpha}\frac{1}{(K+1)(1-\mu_{\max})}\left((1-(1-\mu_{\max}))^{\frac{1}{1-\mu_{\max}}}\right)^{(K+1)(1-\mu_{\max})} \\
&\leq \left(\frac{\sigma(A)}{2}\right)^{\alpha}\frac{(K+1)}{(1-\mu_{\max})}\exp\left(-(K+1)(1-\mu_{\max})\right) \leq \alpha\left(\frac{\sigma(A)}{2}\right)^{\alpha}\frac{\exp\left(-(K+1)(1-\mu_{\max})\right)}{(K+1)(1-\mu_{\max})},
\end{aligned} \tag{12}$$

as stated in the theorem. □

Example. If A is the five-point discretization of $\Delta_{\mathcal{N}}$ on $(0,1)^2$, then for $h=\frac{1}{60}$

$$\frac{\sigma(A)}{2} \approx \frac{\frac{8}{h^2}}{2} = \frac{4}{h^2} \approx 14590.$$

Ignoring the first eight eigenvalues, λ_9 is approximately $(2\pi)^2+(2\pi)^2 \approx 79$ so that $1-\mu_{\max} \leq \lambda_9\frac{2}{\sigma(A)} \approx 0.005412$. Therefore, the condition

$$\alpha\left(\frac{\sigma(A)}{2}\right)^{\alpha}\frac{\exp\left(-(K+1)(1-\mu_{\max})\right)}{(1-\mu_{\max})(K+1)} \leq 10^{-5}$$

gives with $\alpha=0.3$ that for this precision, we should have

$$\frac{\exp\left(-0.005412(K+1)\right)}{0.005412(K+1)} \leq 10^{-5}\frac{1}{0.3\cdot 14590^{0.3}} \approx 1.878\cdot 10^{-6},$$

so that $0.005412(K+1) \approx 10.81$, i.e. we need $K \approx 1996$. This corresponds to line 3 (right) in Table 1.

2.4. Numerical experiments

In the experiments, the matrix A is the five-point finite difference approximation of $-\Delta_{\mathcal{N}}$ on $(0,1)^d$ using N uniformly distributed grid points in one direction. We measure the CPU time and the memory usage of the different approaches. To investigate the accuracy, we use the error indicator $r=\|A^{1-\alpha}A^{\alpha}\mathbf{v}-A\mathbf{v}\|_{\max}$ with a random vector $\mathbf{v}$. The CPU time and the error indicator are computed from the average of ten independent runs.

2.4.1. Experimental analysis of our method

The main parameters in the algorithm of Section 2.2 are the following. N: the number of the grid points in one direction; k: the number of the eigenvectors used to the computation in the first term (6); j: the number of the smallest eigenvectors among the k extreme ones; K: the number of the terms in the Taylor approximation in (9); `res`: the residual in `chdav.m` for approximating the eigenvectors and eigenvalues (the default value is 10^{-8}).

Table 1: Performance of the method in Section 2.2 using the error indicator r with $N=60$, $d=2$ and $\alpha=0.3$.

k	j	K	time [s]	r	`res`	k	j	K	time [s]	r	`res`
20	10	20	0.544	13.3	10^{-8}	40	20	2000	1.21	$1.88\cdot10^{-4}$	10^{-8}
20	10	200	0.526	$9.75\cdot10^{-3}$	10^{-8}	40	20	20000	1.78	$2.36\cdot10^{-4}$	10^{-8}
20	10	2000	0.568	$1.88\cdot10^{-4}$	10^{-8}	16	8	2000	0.44	$1.72\cdot10^{-5}$	10^{-14}
20	10	20000	1.23	$2.00\cdot10^{-4}$	10^{-8}	12	6	2000	0.36	$2.05\cdot10^{-5}$	10^{-14}
20	10	200000	8.88	$1.96\cdot10^{-4}$	10^{-8}	16	8	20000	1.16	$1.91\cdot10^{-10}$	10^{-14}

The experiments confirm some of our expectations:
It is necessary to compute both the smallest and the largest eigenvalues. If the set $\{2\lambda_j/\sigma(A) - 1 : j = 1, 2, \dots, n\}$ is symmetric, then the optimal choice is $j = \frac{k}{2}$.

The convergence of the Taylor series in (9) is slow: we need at least a few hundred terms to get an acceptable accuracy. See lines 1-4 (left) in Table 1. The result in line 3 (left) confirms our estimation in the example.

The optimal number of k is a few tens. For larger values of k, the computation becomes slow and the inaccurate eigenstructure results in a relatively large error contributions to (8). The decomposition in (5) becomes then also inaccurate, such that increasing k further gives an even worse approximation. See the lines for $k = 40$ in Table 1.

A cornerstone of the algorithm in Section 2.2 is an efficient eigensolver. We compared the performance of the built-in Matlab subroutine `eigs.m` and the special subroutines `jdcg.m` (see [12]), `chdav.m` (see [13] and its improved version `bchdav.m`. For matrices of moderate sizes (up to a few thousands of non-zero elements), all subroutines perform equally well. For large 2-dimensional cases, `eigs.m` became the most efficient with an appropriate setting of the tolerance and the iteration number in the algorithm. For large 3-dimensional cases, if multiple eigenvalues are expected, we advice to use the `bchdav.m` subroutine. An experimental comparison of the effect of these eigensolvers can be found in Table 2.

Table 2: Performance of the method in Section 2.2 using different eigensolvers using the error indicator r with $\alpha = 0.3$.

N	`eigs.m`		`jdcg.m`		`chdav.m`		`bchdav.m`	
	r	time [s]	r	time [s]	r	time [s]	r	time [s]
2-dimensional case								
20	$2.9 \cdot 10^{-12}$	0.26	$1.9 \cdot 10^{-12}$	0.27	$2 \cdot 10^{-12}$	0.25	$1.8 \cdot 10^{-12}$	0.28
40	$8.6 \cdot 10^{-12}$	0.44	$1.1 \cdot 10^{-11}$	1.4	$1.2 \cdot 10^{-12}$	0.94	$1.1 \cdot 10^{-11}$	0.76
100	$6.7 \cdot 10^{-12}$	4.9	$5.3 \cdot 10^{-3}$	4.3	$1.7 \cdot 10^{-10}$	10	$1.3 \cdot 10^{-10}$	11.2
3-dimensional case								
20	$3.1 \cdot 10^{-12}$	2.3	$3.7 \cdot 10^{-12}$	3.9	$3.7 \cdot 10^{-12}$	2.4	$3.7 \cdot 10^{-12}$	2
30	$1 \cdot 10^{-11}$	13.8	$9.9 \cdot 10^{-11}$	18.9	$7.6 \cdot 10^{-12}$	11.8	$8.9 \cdot 10^{-12}$	15
40	$1.8 \cdot 10^{-11}$	86	$1.9 \cdot 10^{-11}$	69	$1.8 \cdot 10^{-11}$	43	$1.9 \cdot 10^{-11}$	38

2.4.2. Comparison with the standard method in 3-dimensional case

Since the real computational challenge is to deal with 3-dimensional problems, we focus now on this case. Our approach to compute $A^\alpha \mathbf{v}$ directly is compared now with the standard way: computing A^α first (over time t1) using the `mpower.m` subroutine and multiplying A^α with $\mathbf{v}$ (over time t2) for $\alpha = 0.3$. Results are displayed in Table 3.

Table 3: Performance of the method in Section 2.2 with different parameters using the error indicator r with $d = 3$ and $\alpha = 0.3$.

	conventional method				proposed method			
N	t1[s]	t2[s]	memory[MB]	precision	j	time[s]	memory[MB]	precision
10	0.28	0.002	16	$2.1 \cdot 10^{-11}$	8	0.132	0.12	$5.2 \cdot 10^{-12}$
20	145.9	0.80	$1 \cdot 10^3$	$5.9 \cdot 10^{-10}$	8	0.84	0.99	$2.6 \cdot 10^{-11}$
25	1251	128	$2.6 \cdot 10^3$	$4.5 \cdot 10^{-11}$	8	3.32	5	$7.9 \cdot 10^{-12}$
30	$7.6 \cdot 10^5$	370	$1.5 \cdot 10^4$	$1.1 \cdot 10^{-11}$	8	6.75	8	$8.2 \cdot 10^{-11}$
35	-	-	overflow	-	8	8.84	16	$5.85 \cdot 10^{-11}$
40	-	-	overflow	-	8	20.68	30	$6.83 \cdot 10^{-7}$

3. Conclusions

The present algorithm to compute matrix power-vector products completes the favor of using matrix transformation methods, which delivers an elegant and easily accessible scheme and optimal convergence rate for the finite element and finite difference discretization of space-fractional diffusion problems. Also, as pointed out in the present work, a fast numerical procedure can be constructed to solve the fully discrete schemes using explicit time stepping.

Acknowledgments

The first author was supported by the Hungarian Research Fund OTKA (grant 112157). The project has been supported by the European Union, co-financed by the European Social Fund. EFOP-3.6.1-16-2016-0023.

References

[1] M. M. Meerschaert, C. Tadjeran, Finite difference approximations for fractional advection-dispersion flow equations, J. Comput. Appl. Math. 172 (1) (2004) 65–77. doi:10.1016/j.cam.2004.01.033.

[2] C. Bucur, E. Valdinoci, Nonlocal diffusion and applications, Vol. 20 of Lecture Notes of the Unione Matematica Italiana, Springer, [Cham]; Unione Matematica Italiana, Bologna, 2016. doi:10.1007/978-3-319-28739-3.

[3] Y. Zhang, H. Sun, H. H. Stowell, M. Zayernouri, S. E. Hansen, A review of applications of fractional calculus in Earth system dynamics, Chaos Solitons Fractals 102 (2017) 29–46. doi:10.1016/j.chaos.2017.03.051.

[4] S. Z. Levendorskiĭ, Pricing of the American put under Lévy processes, Int. J. Theor. Appl. Finance 7 (3) (2004) 303–335. doi:10.1142/S0219024904002463.

[5] F. Izsák, B. J. Szekeres, Models of space-fractional diffusion: a critical review, Appl. Math. Lett. 71 (2017) 38–43. doi:10.1016/j.aml.2017.03.006.

[6] R. H. Nochetto, E. Otárola, A. J. Salgado, A PDE approach to fractional diffusion in general domains: a priori error analysis, Found. Comput. Math. 15 (2015) 733–791. doi:10.1007/s10208-014-9208-x.

[7] M. Ilić, F. Liu, I. Turner, V. Anh, Numerical approximation of a fractional-in-space diffusion equation. I, Fract. Calc. Appl. Anal. 8 (3) (2005) 323–341.

[8] M. Ilić, F. Liu, I. Turner, V. Anh, Numerical approximation of a fractional-in-space diffusion equation. II. With nonhomogeneous boundary conditions, Fract. Calc. Appl. Anal. 9 (4) (2006) 333–349.

[9] B. J. Szekeres, F. Izsák, Finite difference approximation of space-fractional diffusion problems: The matrix transformation method, Comput. Math. Appl. 73 (2) (2017) 261–269. doi:10.1016/j.camwa.2016.11.021.

[10] N. J. Higham, L. Lin, An improved Schur-Padé algorithm for fractional powers of a matrix and their Fréchet derivatives, SIAM J. Matrix Anal. Appl. 34 (3) (2013) 1341–1360. doi:10.1137/130906118.

[11] A. H. Al-Mohy, N. J. Higham, Computing the action of the matrix exponential, with an application to exponential integrators, SIAM J. Sci. Comput. 33 (2) (2011) 488–511. doi:10.1137/100788860.

[12] Y. Notay, Combination of Jacobi-Davidson and conjugate gradients for the partial symmetric eigenproblem, Numer. Linear Algebra Appl. 9 (1) (2002) 21–44.

[13] Y. Zhou, Y. Saad, A Chebyshev–Davidson algorithm for large symmetric eigenproblems, SIAM J. Matrix Anal. Appl. 29 (3) (2007) 954–971. doi:10.1137/050630404.