Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 715–720
DOI: 10.15439/2025F2449
ISSN 2300-5963 ACSIS, Vol. 43

An application of BURA solver to fractional super-diffusion problems

Nikola Kosturski, Ivan Lirkov
0000-0002-2215-9835, 0000-0002-5870-2588
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Acad. G. Bonchev, bl. 25A
1113 Sofia, Bulgaria
Email: kosturski@parallel.bas.bg, ivan.lirkov@iict.bas.bg
http://parallel.bas.bg/~kosturski/, http://parallel.bas.bg/~ivan/

Marcin Paprzycki
0000-0002-8069-2152
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw, Poland,
Email: paprzyck@ibspan.waw.pl
http://www.ibspan.waw.pl/~paprzyck/

***Abstract*—In this contribution, the numerical solution of the spectral fractional elliptic equation with power $\alpha \in (1,2)$ is studied. After discretization, the problem is reduced to solving a system of linear algebraic equations. We apply the Best Uniform Rational Approximation (BURA) method and focus on the numerical aspects, related to its implementation. An extensive experimental study is conducted to evaluate the accuracy of the proposed approach. The numerical results confirm the theoretical analysis and demonstrate the effectiveness of using the BURA method for the fractional super-diffusion problems.**

I. Introduction

LET US CONSIDER the spectral fractional elliptic equation with power $\alpha \in (1,2)$, given by

$$\mathcal{A}^\alpha u(x) = f(x), \tag{1}$$

where $\mathcal{A}$ is a self-adjoint elliptic operator in Ω, satisfying homogeneous Dirichlet boundary conditions. The non-local operator $\mathcal{A}^\alpha$ is defined via the spectral decomposition of $\mathcal{A}$, namely

$$\mathcal{A}^\alpha u = \sum_{j=1}^{\infty} \lambda_j^\alpha (u, \psi_j)\psi_j,$$

where $\lambda_j > 0$ are the eigenvalues, respectively, ψ_j are the corresponding normalized eigenfunctions of $\mathcal{A}$, and $(.,.)$ is the L^2 inner product.

Let ω_h be a uniform rectangular mesh, while the Finite Difference Method (FDM) is applied to approximate the operator $\mathcal{A}$. In this case, the FDM discretization of (1) leads to a system of linear algebraic equations with respect to the mesh functions $\mathbf{u}$ and $\mathbf{f}$, in the form

$$A^\alpha \mathbf{u} = \mathbf{f}. \tag{2}$$

Here, $A \in \mathbb{R}^{N\times N}$ is a sparse, symmetric and positive definite (SPD) matrix, and

$$A^\alpha \mathbf{u} = \sum_{j=1}^{N} \lambda_{j,h}^\alpha (\mathbf{u}, \Psi_{j,h})\Psi_{j,h}. \tag{3}$$

As in the continuous case, $\{\lambda_{j,h}\}_{j=1}^N$ is the spectrum of A, and the eigenvectors $\Psi_{j,h}$ are normalized with respect to the Euclidean dot product $(.,.)$.

When $\alpha \in (0,1)$, the problem (1), corresponding to the fractional sub-diffusion, has been extensively studied in the past decade (see, for instance, [1]). When $\alpha \in (1,2)$, the problem (1) is related to the fractional super-diffusion [2], which is the focus of this study. Notably, many numerical methods developed for the sub-diffusion case rely directly on the condition $\alpha \in (0,1)$. Therefore they do not extend naturally to the super-diffusion case. For $\alpha \in (1,2)$, the BURA has one positive pole and one positive zero, making one of the resulting linear systems non-positive definite. To address this, we employ the BURA method, based on the Best Uniform Rational Approximation $r_{\alpha,k}(t)$ of degree k of t^α in the interval $[0,1]$. Specifically, we investigate the numerical behavior of the BURA method [3], [4] when applied to the fractional super-diffusion problems.

II. The BURA Method

The abbreviation BURA stands for Best Uniform Rational Approximation. Let us consider the min-max problem: find $r_{\alpha,k} \in \mathcal{R}(k,k)$ such that

$$\max_{t\in[0,1]} |t^\alpha - r_{\alpha,k}(t)| = \min_{r_k(t)\in\mathcal{R}(k,k)} \max_{t\in[0,1]} |t^\alpha - r_k(t)|, \quad \alpha \in (0,1), \tag{4}$$

where $r_k(t) = P_k(t)/Q_k(t)$, P_k and Q_k are polynomials of degree k. Then the error $E_{\alpha,k}$ of the k-BURA element $r_{\alpha,k}$ is defined as

$$E_{\alpha,k} := \max_{t\in[0,1]} |t^\alpha - r_{\alpha,k}(t)|. \tag{5}$$

A sharp estimate of $E_{\alpha,k}$ has been derived in [5] and has the form:

$$E_{\alpha,k} = 4^{\alpha+1} \sin(\alpha\pi) e^{-2\pi\sqrt{\alpha k}}. \tag{6}$$

Following [6] we introduce the approximation of $A^{-\alpha}$ in the form

$$A^{-\alpha} \approx \lambda_{1,h}^{-\alpha} r_{\alpha,k}\left(\lambda_{1,h} A^{-1}\right),$$

and then

$$\mathbf{u}_k = \lambda_{1,h}^{-\alpha} r_{\alpha,k}\left(\lambda_{1,h} A^{-1}\right)\mathbf{f}, \tag{7}$$

where $\mathbf{u}_k$ is the BURA numerical solution of the linear algebraic system (2). In other words, $\mathbf{u}_k$ is the BURA approximation of the solution $u(x)$ of the fractional elliptic equation (1). Scaling A^{-1} by $\lambda_{1,h}$, above, ensures that all eigenvalues of the matrix are in the interval $(0, 1]$ where the BURA error is uniform.

III. An application of the BURA method

In this work we consider the original problem (2) in the form:

$$A^{\alpha_1} A^{\alpha_2}\mathbf{u} = \mathbf{f},$$

where $\alpha = \alpha_1 + \alpha_2$, $\alpha_{1,2} \in (0, 1]$. Now, the BURA method can be used for solving the following sub-diffusion problems:

$$A^{\alpha_1}\mathbf{w} = \mathbf{f}, \tag{8}$$

$$A^{\alpha_2}\mathbf{u} = \mathbf{w}. \tag{9}$$

The application of the BURA method for solving such problems has been relatively well studied, during the last years.

In [2] the authors used similar approach where $\alpha = \sum_{i=1}^{l} \alpha_i$. In our study, we limit our analysis to the case $l = 2$, for the following reasons. First, using $l > 2$ requires solving larger number of linear systems, which increases the computational complexity, and reduces the efficiency of the solver. Second, for small values of α, a larger k must be used, to achieve good accuracy, which further increases the computational cost.

IV. Numerical experiments

The presented numerical experiments are designed to determine the optimal combination of parameters, balancing the numerical error and the computational time. Specifically, we consider the one dimensional problem (1) with the right hand side

$$f(x) = sin(\pi x) - sin(2\pi x) + sin(3\pi x),$$

where the computational domain Ω is the unit interval $[0, 1]$ and $\alpha = 1.25, 1.5, 1.75$. For this problem, it is possible to compute the exact solution of the problem (1). Therefore, the error in the numerical solution can be also precisely calculated.

The problem is discretized on a uniform mesh ω_h, $h^{-1} = n = 8000, 16000, 32000, 64000$, by applying the standard three point stencil. Next, the BURA method is applied with:

- $\alpha_1 = 1$;
- $\alpha_1 = \alpha_2 = \frac{\alpha}{2}$;
- $\alpha_2 = 1$.

Specifically, the additive BURA method (see [4]) is used, where

$$r_{\alpha,k}(t) = c_0 + \sum_{i=1}^{k} \frac{c_i}{t - d_i}. \tag{10}$$

In this case, the equation (7) can be rewritten as

$$\mathbf{u}_k = \left[c_0 \lambda_{1,h}^{-1} I + \sum_{i=1}^{k} c_i \left(A - \lambda_{1,h} d_i I\right)^{-1}\right]\mathbf{f}, \tag{11}$$

The BRASIL [7], [8] software, version 1.2.1, was used for the computation of the coefficients of the rational approximation $r_{\alpha,k}(t)$ in the interval $[0, 1]$.

The l_2 norm of the error was computed as:

$$\|\mathbf{u} - \mathbf{u}_{exact}\|_{l_2} = \sqrt{\frac{\sum_{i=1}^{n+1}(u_i - u_{exact}(x_i))^2}{n+1}}.$$

The code has been implemented in Python. (See the Appendix for the main part of the code.) The equation (11) requires a solver for the tridiagonal systems of equations. In the implemented solver, the LAPACK [9] subroutine DGTSV was used for solving these tridiagonal systems. Across the solver, double precision real numbers were used.

The code has been tested on a server which has Intel(R) Xeon(R) Silver 4309Y CPU with 16 cores at 2.8 GHz and 32 GB of RAM. However, it should be noted that no attempt was made to parallelize the solution process. This direction of research may be pursued in the future, when substantially larger problems are going to be solved.

V. Numerical results

The l_2 norm of the error for $n = 8000, 32000$ is presented in Tables I and II, the l_2 norm of the error for $n = 64000$ is presented in Fig. 1. Let us note that the error of the numerical solution, in the experiments, consists of a composition of several types of errors. First, the discretization leads to an error. Here, when the Laplace equation is formed, it can be estimated as $O(h^2)$. Second, the rational approximation has an error $E_{\alpha,k}$ and the sharp estimate of $E_{\alpha,k}$ is presented in (6). Thus, the error of the rational approximation depends on αk. For $\alpha \in (0, 1)$, a more accurate rational approximation for α close to 1, and for large k, is obtained. Third, there is also an error in the numerical solution of the systems of linear algebraic equations, in (11). It should be noted that increasing the size of these systems leads to a larger error in the solution. Finally, at every step of the numerical solution of the problem there are rounding errors. Unfortunately, there is no easy way to decouple these errors, and capture them independently.

From Tables I and II it follows that for the same α and the same k a better accuracy is obtained when $\alpha_1 = \alpha_2$. Next, as expected, for α close to 1, the accuracy is higher. However, higher accuracy comes at an additional cost. When $\alpha_1 = \alpha_2$, the BURA method is applied twice to solve (8–9). When $\alpha_1 = 1$ (or $\alpha_2 = 1$) the BURA method is applied only for solving equations (9) (or (8)). Thus, in total, for $\alpha_1 = \alpha_2$, $2k$ linear systems are solved, while for $\alpha_1 = 1$ (or $\alpha_2 = 1$) only $k + 1$ linear systems have to be solved. The practical realization of this comparison can be seen in Table III, where the average execution time (from multiple runs) for $n = 8000$ is presented in milliseconds.

The next observation that can be derived from Tables I and II is that the difference in the accuracy for $\alpha_1 = 1$ and $\alpha_2 = 1$ is less than one percent and thus can be neglected.

From the results represented in Fig. 1 it is possible to compare the error obtained for $\alpha_1 = \alpha_2$, $k = 10$ and for $\alpha_1 = 1$, $k = 19$. In both cases, in total, 20 linear systems

TABLE I
l_2 NORM OF THE ERROR FOR $n = 8000$.

k	(α_1, α_2)								
	(1, 0.25)	(0.625, 0.625)	(0.25, 1)	(1, 0.5)	(0.75, 0.75)	(0.5, 1)	(1, 0.75)	(0.875, 0.875)	(0.75, 1)
12	2.64e-6	1.86e-08	2.64e-6	3.02e-08	1.67e-09	3.02e-08	3.34e-10	2.08e-10	3.34e-10
16	4.91e-7	1.33e-09	4.91e-7	2.47e-09	4.24e-10	2.47e-09	2.86e-10	3.07e-10	2.86e-10
20	1.14e-7	7.50e-10	1.14e-7	3.50e-10	5.01e-10	3.50e-10	3.09e-10	3.05e-10	3.08e-10
24	2.99e-8	8.82e-10	2.99e-8	4.63e-10	5.10e-10	4.63e-10	3.11e-10	3.11e-10	3.11e-10

TABLE II
l_2 NORM OF THE ERROR FOR $n = 32000$.

k	(α_1, α_2)								
	(1, 0.25)	(0.625, 0.625)	(0.25, 1)	(1, 0.5)	(0.75, 0.75)	(0.5, 1)	(1, 0.75)	(0.875, 0.875)	(0.75, 1)
12	2.64e-6	1.97e-08	2.64e-6	3.07e-08	1.62e-09	3.07e-08	4.30e-10	3.10e-10	4.30e-10
16	4.91e-7	1.59e-09	4.91e-7	2.91e-09	1.11e-10	2.91e-09	4.08e-11	1.67e-10	4.08e-11
20	1.14e-7	1.42e-10	1.14e-7	2.85e-10	2.32e-10	2.85e-10	7.51e-11	3.21e-11	7.51e-11
24	3.06e-8	1.87e-10	3.06e-8	4.90e-11	1.94e-11	4.92e-11	8.93e-12	1.91e-10	8.91e-12

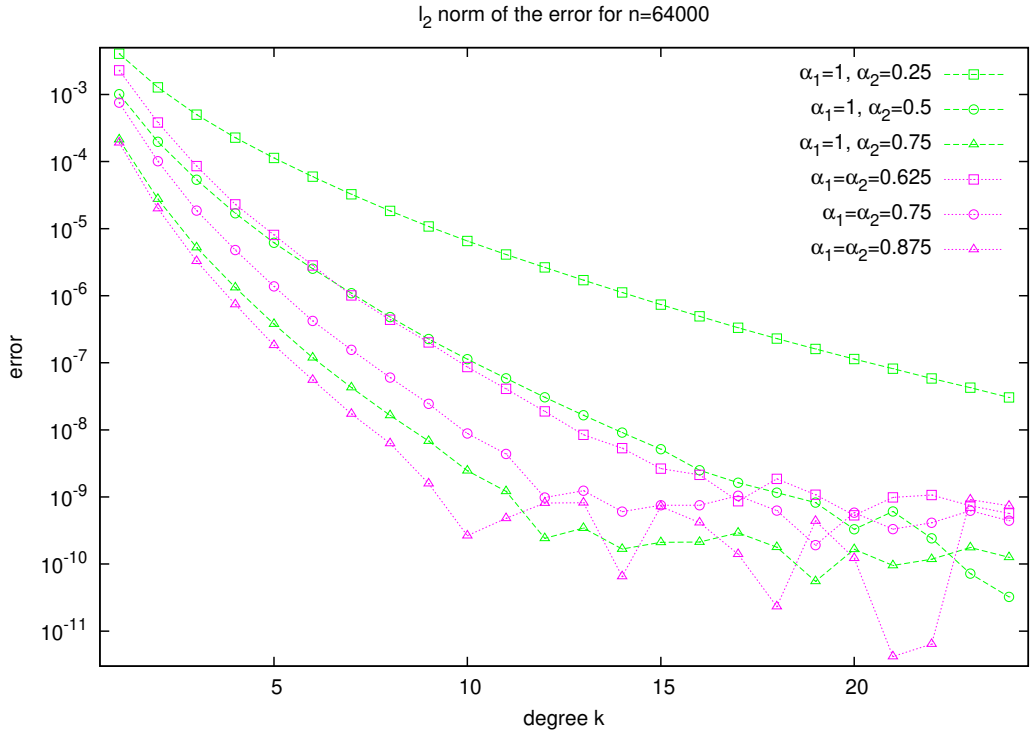

Fig. 1. l_2 norm of the error for $n = 64000$.

have been solved, and the execution time is almost the same. For $\alpha = 1.25$ the obtained errors are $\approx 9.10^{-8}$ and 2.10^{-7}, for $\alpha = 1.5$, the errors are $\approx 9.10^{-9}$ and 8.10^{-9}, for $\alpha = 1.75$ the errors are $\approx 3.10^{-10}$ and 6.10^{-11}. Therefore, it can be concluded that, depending on the value of α, it is possible to choose the optimal order k and the parameter α_1.

The next observation concerns the results depicted in Fig. 1. There, one can see the strange behavior of the error for $\alpha_1 = \alpha_2 = 0.875$ for large k. As noted above, the error consists (among others) of the discretization error and the rational approximation error. For $n = 64000$, the discretization error is $\approx 2.5 \times 10^{-10}$ and this error could not be improved for $k > 10$.

Finally, Fig. 2 shows the average execution time (from multiple runs) for $n = 8000, 16000, 32000, 64000$, in seconds. As noted above, for $\alpha_1 = 1$ the algorithm solves $k + 1$ linear systems of size $n + 1$, while for $\alpha_1 = \alpha_2$ — $2k$ systems of the same size. As expected, the execution time is proportional to $k + 1$ or to k and the linearity is clearly visible in the plot.

VI. CONCLUDING REMARKS

In this contribution, the numerical solution of the spectral fractional elliptic equation with power $\alpha \in (1, 2)$ was studied.

TABLE III
THE EXECUTION TIME FOR $n = 8000$ IN MILLISECONDS.

k	(α_1, α_2)								
	(1, 0.25)	(0.625, 0.625)	(0.25, 1)	(1, 0.5)	(0.75, 0.75)	(0.5, 1)	(1, 0.75)	(0.875, 0.875)	(0.75, 1)
12	2.82	5.09	2.79	2.82	5.09	2.79	2.83	5.09	2.79
16	3.63	6.72	3.62	3.64	6.72	3.62	3.65	6.71	3.61
20	4.44	8.35	4.42	4.44	8.37	4.43	4.46	8.35	4.36
24	5.27	9.96	5.23	5.26	9.98	5.23	5.33	9.96	5.24

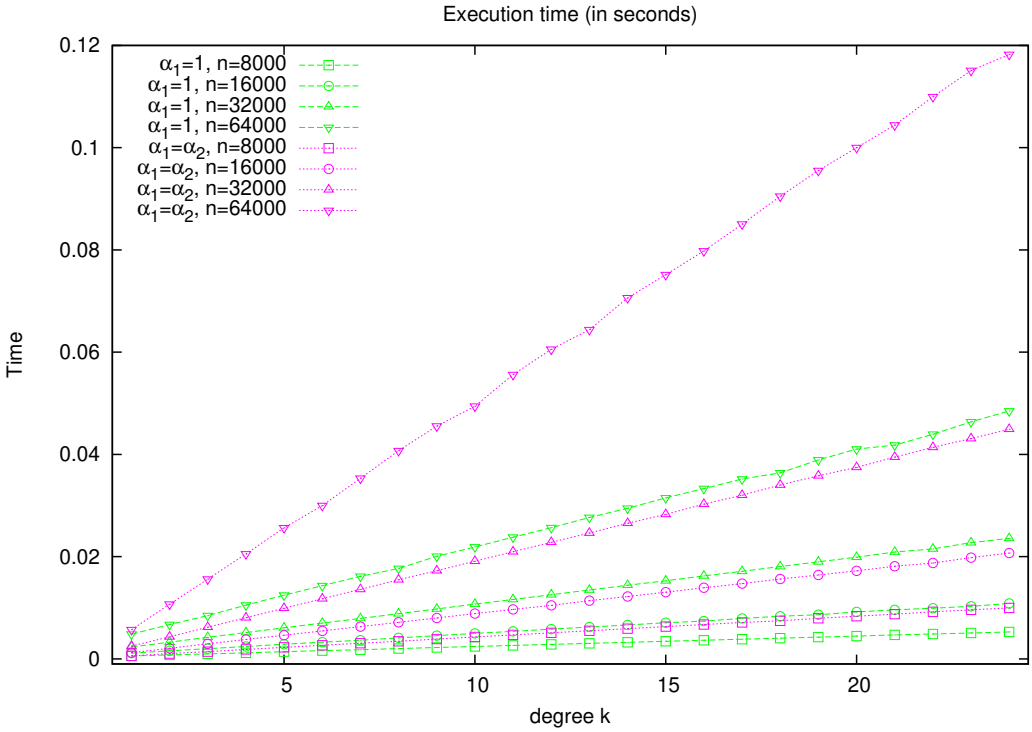

Fig. 2. Execution time in seconds for $n = 8000, 16000, 32000, 64000$.

Specifically, for the problem in question, the Best Uniform Rational Approximation method has been applied. Next a comprehensive set of experiments has been completed. The obtained results match the theoretical expectations. In particular, the fact that the solution error is composed of multiple components has been captured for large values of k and n. This may constitute the actual limit of applicability of the proposed approach (in its standard form).

ACKNOWLEDGMENT

Work presented here is part of the collaborative grant between the Polish Academy of Sciences and the Bulgarian Academy of Sciences, IC-PL/01/2022-2023, "Practical aspects of scientific computing". The work was partially supported by the Bulgarian National Science Fund under grant No. KP-06-N72/2.

REFERENCES

[1] C. Hofreither, "A unified view of some numerical methods for fractional diffusion," *Comp Math Appl.*, vol. 80, no. 2, pp. 332–350, 2020. doi: 10.1016/j.camwa.2019.07.025

[2] D. Slavchev, S. Harizanov, and N. Kosturski, "Analysis on computational issues when approximating fractional powers of sparse SPD matrices," *International Journal of Applied Mathematics*, vol. 37, no. 6, pp. 699–711, 2024. doi: 10.12732/ijam.v37i6.7

[3] S. Harizanov, R. Lazarov, S. Margenov, P. Marinov, and Y. Vutov, "Optimal solvers for linear systems with fractional powers of sparse SPD matrices," *Numer Linear Algebra Appl.*, vol. 25, no. 5, p. e2167, 2018. doi: 10.1002/nla.2167

[4] S. Harizanov, N. Kosturski, I. Lirkov, S. Margenov, and Y. Vutov, "Reduced multiplicative (BURA-MR) and additive (BURA-AR) best uniform rational approximation methods and algorithms for fractional elliptic equations," *Fractal and Fractional*, vol. 5, no. 3, p. 61, 2021. doi: 10.3390/fractalfract5030061

[5] H. Stahl, "Best uniform rational approximation of x^{α} on [0, 1]," *Acta Math*, vol. 190, no. 2, pp. 241–306, 2003.

[6] S. Harizanov, R. Lazarov, S. Margenov, P. Marinov, and J. Pasciak, "Analysis of numerical methods for spectral fractional elliptic equations based on the best uniform rational approximation," *Journal of Computational Physics*, vol. 408, p. 109285, 2020. doi: 10.1016/j.jcp.2020.109285

[7] "Software BRASIL," accessed on September 1, 2022. [Online]. Available: https://baryrat.readthedocs.io/en/latest/#baryrat.brasil

[8] C. Hofreither, "An algorithm for best rational approximation based on barycentric rational interpolation," *Numerical Algorithms*, vol. 88, pp. 365–388, 2021. doi: 10.1007/s11075-020-01042-0

[9] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. Philadelphia: SIAM, 1999.

APPENDIX

```
from baryrat import brasil
from mpmath import mp
from numpy import linalg
from scipy.linalg import lapack
import argparse
import numpy as np
# ...
lambda1 = (2*(args.n)*np.sin(np.pi/2/(args.n)))**2
# solver
def thomas(dl, d, du, b):
    _, _, _, x, _ = lapack.dgtsv(dl, d, du, b)
    return x

# BURA
class bura:
    def __init__(self, alpha, k):
        if alpha == 0 or alpha == 1:
            self._alpha = alpha
            return
        rat = brasil(
            lambda x: x**alpha,
            (0, 1),
            k,
            tol=np.finfo(np.double).eps,
            npi=-30,
            maxiter=10000,
        )

        poles = np.real_if_close(np.array(rat.poles(True), dtype=np.complex128))
        zeros = np.real_if_close(np.array(rat.zeros(True), dtype=np.complex128))

        d = 1 / np.flip(np.sort(poles))
        z = 1 / zeros
        b = np.zeros(k + 1)
        b[-1] = rat(1)
        m = 1 / np.subtract.outer(np.append(z, 1), d)
        m = np.hstack((m, np.ones((m.shape[0], 1))))
        c = np.real(np.array(mp.lu_solve(np.real(m), b), dtype=np.complex128))
        self._alpha, self._k, self._c, self._d = alpha, k, c, d

    def solve(self, dl, d, du, b, lambda1=lambda1):
        if self._alpha == 0:
            return b
        if self._alpha == 1:
            return thomas(dl, d, du, b)
        n = len(b)
        r = self._c[-1] * b
        for i in range(self._k):
            r += (
                lambda1
                * self._c[i]
                * thomas(dl, d - lambda1 * self._d[i] * np.ones(n), du, b)
            )
        return lambda1**-self._alpha * r
```

```
bura1, bura2 = bura(args.alpha1, args.k1), bura(args.alpha2, args.k2)
num = bura1.solve(lower_diag, main_diag, upper_diag, b)
num = bura2.solve(lower_diag, main_diag, upper_diag, num)

exact = exact_solution(dofs, alpha)
err = exact - num
max_err = np.max(np.abs(err))
print("Max error:", max_err)
print("L2 error:", linalg.norm(err)/((args.n+1)**0.5))
```