УДК 530.191:539.211:539.25:514.86 DOI 10.36910/775.24153966.2025.81.10

І.В. Красіков, І.Є. Красікова, О.О. Васільєв

Інститут проблем матеріалознавства ім. І.М. Францевича НАНУ

РЕАЛІЗАЦІЯ ШВИДКОГО ОБЧИСЛЕННЯ МУЛЬТИФРАКТАЛЬНОГО СПЕКТРА ТРИВИМІРНОГО ЗОБРАЖЕННЯ

В роботі з алгоритмічної точки зору розглянуто метод обчислення мультифрактальних характеристик методом box counting для тривимірних зображень. Показано, що застосування алгоритмів префіксних сум, швидкого піднесення до степеня та багатопотокових обчислень забезпечує можливість ефективного аналізу зображень великого обсягу (десятки гігавокселів) за прийнятний час (від десятків секунд до кількох хвилин) за рахунок підвищеного використання оперативної пам'яті. Запропоновано методику оптимального вибору розміру зображення з метою максимізації кількості точок на графіках залежностей узагальнених статистичних сум від розмірів боксів, що сприяє підвищенню точності обчислень. Надано алгоритмічні рішення для реалізації ефективної багатопотокової обробки даних.

Ключові слова: *мультифрактальна розмірність, алгоритм, тривимірні структури, програмне забезпечення, ефективність*

I.V. Krasikov, I.E. Krasikova, O.O. Vasiliev

IMPLEMENTATION OF FAST CALCULATION OF THE MULTIFRACTAL SPECTRUM OF A THREE-DIMENSIONAL IMAGE

This paper examines the algorithmic aspects of calculating multifractal characteristics using the box-counting method for three-dimensional images. It is demonstrated that the application of prefix sums, fast exponentiation, and multithreaded computing enables efficient analysis of large images (tens of gigavoxels) within a reasonable time frame (from tens of seconds to several minutes) at the cost of increased RAM consumption. A methodology for selecting the optimal image size is proposed to maximize the number of points on the graphs of generalized statistical sum dependencies on box size, thereby improving calculation accuracy. Algorithmic solutions for efficient multithreaded data processing are also presented.

Keywords: *multifractal dimension, algorithm, three-dimensional structures, software, efficiency*

1. Problem formulation.

The application of multifractal analysis in materials science is gradually expanding from two-dimensional problems (e.g., the analysis of polished sections and cross-sections) to three-dimensional ones, driven by advancements in both experimental and computational technologies. In the three-dimensional case, the analysis encompasses the entire structure of an object without attempting to reduce its characteristics to those of its two-dimensional sections. While a relationship between two-dimensional and three-dimensional structural characteristics undoubtedly exists — particularly in the context of multifractal descriptions — its nature remains insufficiently explored. At the same time, modern research methods enable the acquisition of three-dimensional structural information, necessitating the development of efficient methods for calculating multifractal characteristics in three-dimensional structures.

From an algorithmic perspective, the problem of calculating multifractal characteristics for three-dimensional objects is a direct generalization of its two-dimensional counterpart and reduces to computing the spectrum of multifractal dimensions for a three-dimensional matrix of binary values (0/1) using the box-counting method. The primary challenge in transitioning to three-dimensional analysis lies in the increased computational complexity and memory requirements, which scale at least proportionally to the matrix size N. Consequently, the direct application of algorithms that exhibit acceptable efficiency in the two-dimensional case becomes impractical and necessitates substantial algorithmic optimization.

Notably, the choice of algorithms in scientific research on materials science is often underexplored. Researchers frequently provide only references to third-party software (if used) or omit descriptions of the implemented methods entirely. This lack of methodological transparency complicates the reproducibility of results and the comparability of different approaches, particularly given that different algorithmic strategies can lead to significant variations in multifractal analysis outcomes (e.g., [1], which compares results obtained via the box-counting and caliper methods).

This paper presents a methodology for the efficient computation of the multifractal dimension spectrum D_q using the box-counting method. Various factors affecting the accuracy of multifractal characteristic calculations for three-dimensional images are examined. While the primary focus is on three-dimensional structures, the proposed approach is equally applicable to two-dimensional cases without modification. The box-counting method was chosen for its versatility and widespread use, particularly in the field of materials science.

2. Materials and methods.

The computational times, averaged over 10 computational experiments, were obtained on an Alfa Server workstation equipped with two Intel Xeon Platinum 8280 processors (56 cores/112 threads), 512 GB of DDR4 RAM, and running Ubuntu 20.04.6 LTS. In all experiments, a set of seven values of the moment order q (see below) was used to illustrate a typical scenario.

The box-counting method is one of the most widely used techniques for assessing the multifractal characteristics of complex structures in materials science. A detailed description of this method can be found, for example, in [2,3]. The method involves covering the studied object with a grid of boxes of size ε and analyzing the dependence of the number of occupied boxes on ε. For multifractal analysis, a generalized statistical sum is used:

$$Z(q,\varepsilon) = \sum_i p_i^q(\varepsilon) \tag{1}$$

where $p_i(\varepsilon)$ is the probability of object points falling into the i-th box. The multifractal dimension is then defined as:

$$D_q = \lim_{\varepsilon \to 0} \frac{-1}{q-1} \frac{\ln Z(q,\varepsilon)}{\ln(\varepsilon)} \tag{2}$$

The case of $q=-1$ is considered separately and does not impact the computational efficiency.

Taking into account the limited range of fractality, the dimensionality in a real structure is calculated as:

$$D_q = \frac{-1}{q-1} \frac{d \ln Z(q,\varepsilon)}{d \ln(\varepsilon)} \tag{3}$$

where the computation is performed within the fractality range, defined as the region of ε values for which the dependence of $\ln(Z(q,\varepsilon))$ on $\ln(\varepsilon)$ is linear.

The task of computing the D_q spectrum involves constructing a matrix of generalized statistical sums $Z(q,\varepsilon)$ and further processing this data. Determining the fractality range (the set of ε values for which the graph of $Z(q,\varepsilon)$ versus ε in double logarithmic coordinates is a straight line) and calculating the D_q spectrum from the obtained datasets are not addressed in this paper, as they do not impact computational efficiency and constitute a separate algorithmic problem deserving independent study.

In the following, we limit ourselves to cubic images represented as an $N \times N \times N$ matrix. The box counting method involves dividing the image into boxes of size ε, which requires N to be divisible by ε. The number of valid values of ε directly influences computation time but is also critical for the accuracy of the D_q spectrum calculation.

3. Results and discussion.

Selection of image and box sizes.

For accurate analysis of three-dimensional images represented as bit matrices of size $N \times N \times N$, it is crucial to maximize the precision of multifractal dimension (D_q) calculations. This requires the largest possible set of box size values (ε). Increasing the number of possible ε values enhances the accuracy of approximating the dependence of the generalized statistical sum on the box size but also increases computational costs. At the same time, to ensure complete coverage of the object by the grid of boxes, ε values must be divisors of N.

An analysis of the number of divisors of integers in the range $1 \le N \le 3000$ (Fig. 1) demonstrates that selecting values such as N=1260,1440,…, which correspond to the maximum number of divisors in this range (see sequence A067128 in OEIS [4]), is most advantageous. For experimental data where N does not correspond to one of these optimal values—especially when N has a small number of divisors — it is preferable to discard "extra" parts of the data at the image edges and select the closest N with a sufficient number of divisors. This approach has minimal impact on the accuracy of D_q calculations, given the negligible influence of edge effects on multifractal dimension computations [5].

When studying artificially generated models of three-dimensional structures rather than experimentally obtained data, it is possible to choose an appropriate model size from the outset, taking into account available memory and allowable computation time.

Matrix scanning and box occupancy calculation.

The computation of generalized statistical sums for each pair (q,ε) requires determining the number of unit elements in each box of size ε.

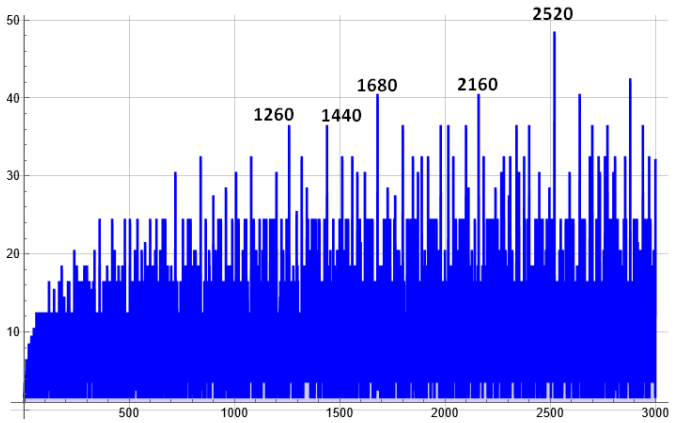

Fig.1. **Number of divisors for integers from 1 to 3000**

When using a direct scanning approach (brute force method), the entire matrix is processed repeatedly for each divisor ε of *N*. The computation time increases significantly with growing *N* (see Table 1).

Table 1

Computation time for generalized statistical sums using the brute force method for different values of *N*.

N	1260	1440	1680	2520
Calculation time, min	19.6	29.2	72.0	197.2

Pseudo-code for the method:

```
for Divisors N ε do
  for i = 0 to N/ε
  for j = 0 to N/ε
  for k = 0 to N/ε
Scan the box (i*ε,j*ε,k*ε)-(i*ε+ ε,j*ε+ ε,k*ε+ ε)
Calculation of n_i and p_i
  for Value q_i do
Calculate (p_i)^q and sum with Z(q,ε)
```

Note that when *N* is an exact power of some number (typically 2, less commonly 3), transitioning from a smaller divisor to a larger one results in each larger box being composed of a certain number (2^d or 3^d, where d is the dimension of the space) of smaller boxes already considered in the previous step. This enables a more efficient computation of the number of unit elements in a larger box by summing the corresponding values from the smaller boxes. This approach is used, for instance, in [6]. However, the number of divisors in such cases is relatively small — for example, for $N=2048$, there are only 12 divisors, whereas for nearby values of *N*, such as 1680 or 2160, the number of divisors reaches 40. Therefore, despite being a common practice, relying on "power" values of *N* is generally inadvisable, as demonstrated in [7], where it is shown that such an approach can lead to significant errors in the computation of fractal dimensions.

Conversely, for values of *N* with a large number of divisors, the larger boxes do not fully consist of smaller boxes. As a result, the number of unit elements within them must, at least partially, be recomputed at each step. The following method provides an efficient solution to this problem.

Prefix sum method.

Computational efficiency can be significantly improved by constructing a prefix sum matrix `M[i][j][k]`, which stores the cumulative sum of the unit elements in the submatrix (0,0,0)−(i,j,k). Using this structure, the number of unit elements in any given box can then be determined via the inclusion-exclusion principle:

```
M[l][m][n] - M[i-1][m][n] - M[l][j-1][n] - M[l][m][k-1]
  + M[i-1][j-1][n]+ M[i-1][m][k-1]+ M[l][j-1][k-1]
  - M[i-1][j-1][k-1]
```

The main drawback of this approach is the significant increase in memory consumption due to the use of an auxiliary matrix, whose size greatly exceeds that of the original bit matrix. This is because, if fully populated with ones, the values in the auxiliary matrix can reach N^3. The use of 4-byte unsigned

integers imposes a limit of N=1625. Considering the previously discussed data on the number of divisors, the optimal values are N=1440 or N=1260. In these cases, the total memory required to store both the bit and auxiliary matrices is approximately 11.5 GB and 7.7 GB, respectively.

If 8-byte unsigned integers are used, the maximum feasible value of N increases to approximately 2.6 million voxels. However, the computational complexity — requiring the processing of 1.8×10^{19} matrix elements — far exceeds the capabilities of modern computing systems, even with multi-threaded processing. Additionally, the memory required for storing the matrices would reach 130 exabytes, making such an approach infeasible.

Even for a relatively modest value of N=2520, the total memory required for storing both matrices reaches 121 GB. Thus, considering the hardware limitations of general-purpose computing systems, the optimal range of N values at the current level of technology is 1260–1440 voxels. If a substantial amount of RAM is available, this range can be extended to 1680–2520 voxels.

Constructing the prefix sum matrix in a single scan significantly improves computational efficiency. However, this step is poorly suited for parallelization due to dependencies between data elements: each element `M[i][j][k]` is computed based on the previously calculated values `M[i-1][j][k]`, `M[i][j-1][k]`, and `M[i][j][k-1]`. The corresponding pseudocode is presented below:

```
for i = 1 to N do
for j = 1 to N do
for k = 1 to N do
M[i][j][k] = Src[i-1][j-1][k-1]
+ M[i-1][j][k] + M[i][j-1][k] + M[i][j][k-1]
- M[i-1][j-1][k] - M[i-1][j][k-1] - M[i][j-1][k-1]
+ M[i-1][j-1][k-1];
```

The order of the nested loops must be maintained as specified due to processor architecture constraints. Altering the traversal order results in a several-fold increase in computation time due to disruptions in data cache locality.

Parallelizing the computation of prefix sums is an extremely challenging task. The dependence of each matrix element on previously computed values imposes a strict calculation order, making it impossible to evenly distribute the workload across threads. Implementing a parallel algorithm requires complex synchronization mechanisms, which introduce significant overhead, effectively nullifying the potential performance gains. Attempts to develop an efficient parallel algorithm for prefix sum computation have not yielded substantial improvements; even in the best case, the achieved speedup was negligible compared to the considerable increase in code complexity.

Given these limitations, a single-pass scan of the bit matrix was chosen as the preferred approach. This method enables the subsequent retrieval of π values for all boxes, regardless of their size, in $O(1)$ time. The time required to construct the prefix sum matrix for various values of N is presented in Table 2.

Table 2

Computation time for the prefix sum matrix at different values of N

N	1260	1440	1680	2520
Calculation time T, s	10,0	15,1	38,3	136,6
$T/N^3\times10^9$	5,0	5,1	8,1	8,5

A sharp increase in the T/N^3 ratio is associated with the transition to a larger matrix element size (from 4 bytes to 8 bytes), as well as with cache locality issues arising from large memory sizes.

Once the auxiliary prefix sum matrix has been constructed, the next step is to compute the generalized statistical sums matrix.

Computing generalized statistical sums.

At this stage of the algorithm, a matrix of generalized statistical sums must be constructed. For each value of q, this matrix consists of a vector of `double` values, one for each ε.

Using fast exponentiation.

Experience gained during the development of Fraculator [9], a program for computing the multifractal characteristics of two-dimensional images, indicates that employing the fast exponentiation algorithm [10] instead of the standard `pow` function for integer values of q significantly improves performance [11]. As shown in Table 3, simply replacing the `pow` function call with fast exponentiation results in a performance increase of approximately 1.7-fold for typical input matrix sizes.

Table 3

Computation time for the matrix of generalized statistical sums for different values of *N* using various optimization methods

	Calculation time, s		
	Serial		Parallelized
N	Standard `pow` function	Fast exponentiation	
1260	488	283	13.9
1440	730	423	20.3
1680	1705	1012	48.7
2520	5077	4246	162.0

Parallelization of Computation.

Further acceleration of calculations is achieved through multithreading. However, simply dividing the calculation of generalized statistical sums between threads for different ε values is inefficient and provides little or no real speedup. This is because while ε=1 requires processing N^3 boxes, ε=2 requires only $N^3/8$ and so on. The total computation time for all ε values is estimated as follows:

$$\sum_{i=1}^{\infty}\frac{1}{n^3}\approx 1.202 \tag{4}$$

where the time for calculating the generalized statistical sum for ε=1 is assumed to be 1. Even if all sums except for ε=1 are calculated instantly across all threads, this only speeds up the calculation by 17%. Therefore, standard parallelization algorithms, including those using OpenMP, are not applicable.

The solution involves finer-grained parallelization, i.e., parallelizing computations for the same value of ε. Since the computation for ε=2 is much smaller than for ε=1, multithreading is used only for ε=1, while other ε values are processed in a single thread as separate tasks.

Given a limited number of processors/cores, it is important to distribute the workload evenly. To achieve this, a simplified model is used: out of *M* threads, *K* threads handle ε>1, while the remaining threads work on ε=1. The total computation time is minimized by considering the expression (5) derived from Equation (4), when it reaches its minimum:

$$\frac{1}{M-K}+\frac{0.2}{K} \tag{5}$$

The highest computing speed is achieved when the optimal distribution of threads is found (K=0.31*M*).

In the program, the calculation for ε=1 was divided into 64 parts, which were dynamically assigned to threads. The number of threads was determined by the condition in Equation (5). For other ε values, each sum was computed as a separate task. To avoid the overhead of mutexes, atomic task indices were used with `std::atomic<int>`.

Figure 2 shows a general block diagram of the D_q spectrum calculation.

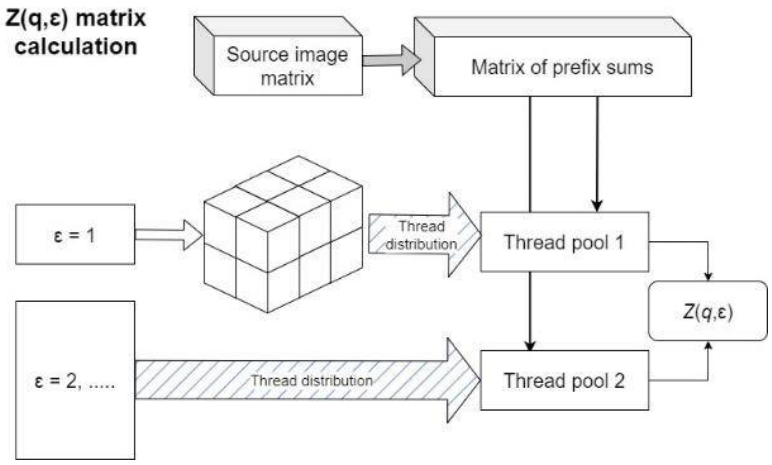

Fig. 2. **Scheme for calculating the matrix of generalized statistical sums**

As can be seen from Table 3, parallelization allowed us to speed up the calculations by approximately 20 times (for the given computer architecture and number of cores). Other optimizations at the architectural level (memory alignment, unsynchronized writing data, etc.) had an unstable effect within the measurement error.

4. Conclusions.

1.A method is proposed for selecting the size of the source images by choosing values with the maximum number of divisors that are closest to, but not exceeding, the available sizes. This approach helps to maximize the number of points and improve the accuracy in determining multifractal characteristics from the graphs of the dependence of generalized statistical sums on the box size.

2.To accelerate computation, the use of the prefix sum method, fast exponentiation, and multithreading is proposed. Together, these techniques resulted in a computational speedup of approximately 50 times compared to the straightforward brute-force method.

3.Efficient parallelization requires fine-grained parallelization of calculations, particularly splitting the task of calculating the sum for individual boxes into subtasks.

4.The applied improvements enabled a tenfold speedup in calculating the multifractal characteristic spectrum, ensuring that it can be completed within minutes even for gigavoxel images.

References.

1.Maryenko N. I. Fractal dimension of external linear contour of human cerebellum (magnetic resonance imaging study) / N. I. Maryenko, O. Yu. Stepanenko // *Reports of Morphology*. - 2021. - No. 27 (2). - P. 16-22.

2.Bouda, Martin, Joshua S. Caplan, and James E. Saiers. "Box-Counting Dimension Revisited: Presenting an Efficient Method of Minimising Quantification Error and an Assessment of the Self-Similarity of Structural Root Systems." *Frontiers in Plant Science* 7 (February 18, 2016). https://doi.org/10.3389/fpls.2016.00149.

3.Da Silva, D., F. Boudon, C. Godin, O. Puech, C. Smith, and H. Sinoquet. "A Critical Appraisal of the Box Counting Method to Assess the Fractal Dimension of Tree Crowns." In *Advances in Visual Computing*, edited by George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Paolo Remagnino, Ara Nefian, Gopi Meenakshisundaram, et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. https://doi.org/10.1007/11919476_75.

4.A067128. *Ramanujan's largely composite numbers, defined to be numbers m such that d(m) >= d(k) for k = 1 to m-1*. Electronic resource https://oeis.org/A067128, accessed 11.03.2025.

5.I.E. Krasikova, V.V. Kartuzov, I.V. Krasikov. Characteristics of the computer implementation of the algorithm for calculating the fractal dimension of two-dimensional images. In *Mathematical models and computational experiment in materials science*. 15. Proceedings of IPM NASU. - Kyiv, 20013. - P. 67-73.

6.Ruiz De Miras, J., M.A. Posadas, A.J. Ibáñez-Molina, M.F. Soriano, and S. Iglesias-Parro. "Fast Computation of Fractal Dimension for 2D, 3D and 4D Data." *Journal of Computational Science* 66 (January 2023): 101908. https://doi.org/10.1016/j.jocs.2022.101908.

7.I.E. Krasikova, I.V. Krasikov, V.V. Kartuzov. Determination of fractal characteristics of materials structure by multifractal image analysis. Computational experiment on model objects. In *Mathematical models and computational experiment in materials science*. 9th edition. Proceedings of IPM NASU. - Kyiv, 2007. - P. 79-84.

8.Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. 3rd ed. Cambridge, MA: MIT Press, 2009.

9.Krasikov I.V., Krasikova I.E., Kartuzov V.V., Krasikov A.I. *Computer programme "Fraculator 2". Certificate of copyright registration for work No. 115438*. Ukraine, 25.10.2022.

10.Knuth, Donald E. *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. 2nd ed. Reading, MA: Addison-Wesley, 1997.

11.Optimisation of floating-point number raising to an integer degree in computing programmes. In *Collection of abstracts of the Fourteenth All-Ukrainian, Twenty-first Regional Scientific Conference of Young Researchers "Actual Problems of Mathematics and Informatics*." - Zaporizhzhia, 2023. - P. 148-150.

Рецензент: Гордієнко Юрій Григорович, професор кафедри обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», доктор фізико-математичних наук