Recent Advances in Fast Algorithms for PDEs

Prabir Daripa,
Department of Mathematics,
Texas A&M University,
College Station, TX–77843,
Email: daripa@math.tamu.edu

Abstract

Fast and accurate algorithms provide solutions of problems at a reasonable cost with high accuracy; hence allows reliable computations of large scale problems. Such algorithms, if parallelizable, offer further opportunities to solve even larger scale problems at a low cost. The applications of such algorithms are abound. A series of recently developed such fast and parallel algorithms for elliptic PDE's are surveyed and the basic mathematical philosophy behind such algorithms is described. Application possibilities are described and some results from implementations of these algrorithms are presented.

1 Introduction

Computers are now constantly being used as an experimental tool to explore science, to make new discoveries and to solve many practical problems which affect our day to day lives. In spite of this computer revolution, there are many problems of national interest which are either too slow computationally or outright intractable due to lack of efficient computational methodologies. To overcome these computational difficulties, it is imperative that new and more efficient ways to solve practical problems through improvements in hardware and algorithms should be developed.

In past few decades, significant advances have taken place in the area of fast algorithms and numerical methods for solving partial differential equations by various methods. There are many applied problems of scientific and industrial interests which require solutions of elliptic and parabolic equations. Analytically intractable problems of this type are solved using numerical methods. One of the efficient ways to solve parabolic equations is to use a finite difference implicit or semi-implicit scheme for linear terms and an explicit one for the convection term. This yields, at each time step, a linear elliptic problem. Therefore, the developement of efficient solvers for elliptic problems is very important. One of the methods for solving such problems is integral equation methods. The integral equations usually involve line, surface or volume integrals depending on the dimensionality and the nature of the problem. Numerical solutions of these integral equations by most iterative methods such as Generalized Conjugate Residual type algorithms require evaluations of these integrals at each iteration. Numerical evaluation of these integrals by quadrature results

matrix-vector multiplication which have $0(n^2)$ operation count for each surface integral where n is the number of nodes in the domain of integration. Such operation counts are prohibitively large for computation of many large scale problems in realistic time.

In recent years, algorithms for rapid evaluation of these integrals and matrix-vector multiplications have appeared These algorithms reduce the complexity to $0(n)$ from $0(n^2)$ but have large constants of proportionality k (which measures the number of operations required per point). Therefore, the number of nodes in the domain of the integral, n, has to be very large to offset the cost of such high values of k for these algorithms to be computationally superior to algorithms which may have very small values of k but slightly large values of complexity. Furthermore, if these algorithms (with very small values of k but slightly large values of complexity) have better accuracy, then such algorithms could be preferable to the ones with better compexity but less accuracy even for very large values of N. The subject of this paper is author's recent and ongoing research in the development, implementation and applications of such algorithms. These algorithms are analysis-based, have complexity $0(\log n)$ per node, have small values of k and have almost spectral accuracy. Moreover, these are easy to implementa and are applicable in complex geometry,

2 A Brief Overview of Mathematical Preliminaries

The basic mathematical formulation behind these fast algorithms is based on representation of solutions of elliptic partial differential equations using Green's function method. In particular, free space Green's functions (the fundamental or principal solutions) for various elliptic operators are used in this formulation. In this formulation, most expensive part of the compuatation is associated with the evaluation of convolution integrals with high accuracy. Our fast algorithm is aimed at computing such integrals very accurately at a very low cost within a unit ball. For arbitrary domains, this algorithm is currently used in conjunction with domain embedding method which eliminates the need for numerical evaluation of these convolution integrals for various complex domains. Direct fast algorithms without any embedding are in progress.

In this short article, we briefly mention some of the very rudimentary ideas behind the development of some of our fast algorithms. It is worth mentioning here that the analysis required behind development of such algorithms may be more involved. For the analysis involved and some applications of such algorithms, see [1]-[6]. Consider the following very well known Dirichlet problem for the Poisson equation.

$$-\Delta u = f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \quad u = g(\mathbf{y}), \quad \mathbf{y} \in \Gamma = \partial\Omega, \tag{2.1}$$

where Ω is a plane domain bounded by a smooth curve Γ and Δ is the two-dimensional Laplace operator. The solution of this equation can be written as

$$u(\mathbf{x}) = v(\mathbf{x}) + F(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{2.2}$$

where

$$F(\mathbf{x}) = -\frac{1}{2\pi}\int_{\Omega} \log|\mathbf{x}-\zeta|\ f(\zeta)d\zeta, \qquad \mathbf{x} \in \Omega, \tag{2.3}$$

and $v(\mathbf{x})$ is the solution of the following problem:

$$\nabla^2 v(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega, \quad v(\mathbf{y}) = g(\mathbf{y}) - F(\mathbf{y}), \quad \mathbf{y} \in \Gamma. \tag{2.4}$$

The solution of this Dirichlet problem (2.4) for the Laplace's equation is standard and can be computed very efficiently in complicated domains using boundary element method. For example, solution of equation (2.4) can be represented in terms of distribution of dipoles $\mu(\mathbf{y})$ on Γ

$$v(\mathbf{x}) = -\frac{1}{\pi} \int_\Gamma \frac{\partial}{\partial \nu} \log |\mathbf{x} - \mathbf{y}| \ \mu(\mathbf{y}) d\mathbf{y}, \qquad \mathbf{x} \in \Omega, \quad \mathbf{y} \in \Gamma, \tag{2.5}$$

and the dipole strength can be computed from an integral equation, namely

$$v(\mathbf{y}) = -\mu(\mathbf{y}) - \frac{1}{\pi} \int_\Gamma \frac{\partial}{\partial \nu} \log |\mathbf{y} - \zeta| \ \mu(\zeta) d\zeta, \qquad \mathbf{y} \in \Gamma. \tag{2.6}$$

The partial derivatives in (2.5) and (2.6) are in the direction normal to the curve Γ.

As seen from (2.2), computation of $u(\mathbf{x})$ contains two components: computations of $v(\mathbf{x})$ and $F(\mathbf{x})$ for $\mathbf{x} \in \mathbf{\Omega}$. The computation of $v(\mathbf{x}), \mathbf{x} \in \mathbf{\Omega}$ can be done with little computations since it involves only the boundary integrals (see eqns (2.5) and (2.6)) with effective reduction of degrees of freedom by one. In addition, this method is known to give very accurate results. These considerations have led to very efficient and successful applications of this method in cases where the problem is linear and the function $f(\mathbf{x})$ in (2.1) is identically zero.

When $f(\mathbf{x}) \neq \mathbf{0}$ and depends on the solution u which is often the case in many practical problems, equation (2.2) is a Fredholm integral equation of second kind. This can be solved by Picard iteration which would require explicit evaluation of the singular integral $F(\mathbf{x})$ defined by (2.3). In the discrete representation of $\Omega \in R^2$ by N^2 points, the numerical method would require evaluation of N^2 integrals like (2.3), one for each point at each level of Picard iteration. Since straight forward computation of each integral by numerical quadrature requires N^2 operations, the algorithmic complexity of the evaluations of $u(\mathbf{x})$ for all N^2 points is $O(N^4)$. In R^3, this cost is proportional to $O(N^6)$ (the kernel in 3D is $\frac{1}{|\mathbf{x}-\zeta|}$ in (2.3)). This becomes prohibitively expensive, in particular for time dependent nonlinear problems where the problem (2.1) would have to be solved repeatedly for convergence at each time level. This method is computationally so intensive for nonlinear time dependent problems in higher dimensions, that it is widely thought as less versatile and therefore is less widely used than the finite difference or finite element(FEM) methods, in particular for nonlinear problems.

There are instances when Cauchy principal value or/and hypersingular integrals are to be evaluated. For example, higher derivatives of $u(\mathbf{x})$ can be represented in terms of such integrals as can be seen from (2.2). Actually hypersingular integral equations arise in many areas including scattering theory, fluid mechanics, fracture mechanics and elasticity. Because the singularity in the integrand is no more weak, great care is required in how these integrals are evaluated. For example, approximation of the Cauchy principal value integrals by simple numerical quadrature without explicitly including the contribution due to the singularity itself (which may require careful nontrivial analysis) may not be sufficient [4].

Therefore, issues of efficiency as well as accuracy considerations are very important in solving these singular integral equations. We have, in recent years, developed analysis-based fast algorithms that are high order accurate, easy to implement and requires few number of operations per point ([1]-[6]). Analysis based on the same basic ideas can be applied to rapidly evaluate various other integrals in any dimensions: be it singular, Cauchy principal value or hypersingular integrals. These ideas also extend to solving other pdes in multi-dimensional complex domains with similar efficiency and accuracy and with considerable ease of implementation.

We give a very short account of the basic philosophy behind fast and accurate evaluation of singular integrals of the type (2.3), (2.4) and (2.6) when the domain is a unit disk. The idea is very simple but requires some amount of ingenuity and analysis due to singular (including Cauchy principal value) nature of these integrals. Our analysis exploits special Fourier transform properties of these convolution-type integrals and incorporates the exact contribution due to the singularity to the integral, if any. The exact contribution is explicitly evaluated through analysis. High accuracy of the resulting algorithms is due to these exact analyses.

The low computational complexity of the algorithm has to do with certain patterns that result due to the above analyses; see [1]-[6] for details. Due to analyses, the algorithm in simplistic form amounts to the following when the domain is a unit disk. The unit disk is discretized using $M \times N$ lattice points with M equidistant points in the radial direction and N equidistant points in the circular direction. The fast algorithm employs two groups of Fourier Transforms (FFT) for each of the functions: $f(\mathbf{x})$ and $F(\mathbf{x})$ for each fixed radius r_j; and some recursive relations obtained through analyses that facilitates calculation of the Fourier coefficients $F_k(r_j)$ (of the values of the singular integrals, $F(\mathbf{x})$) from the Fourier coefficients $f_k(r_j)$ (of $f(\mathbf{x})$) at a cost which is much smaller than the cost of evaluating the Fourier Transforms using FFTs. Each use of FFT requires $N \log N$ operations and since there are $2M$ such FFT's of length N, the total number of operation count for evaluating all of $M \times N$ area integrals (one integral for each point) is asymptotically $O(MN \log N)$ or $(\log N)$ per integral. This operation count is much smaller than $O(M^2N^2)$ which will be required if these integrals were to be evaluated directly. Hence, the overall theoretical computational complexity remains at $O(\log N)$ per point. In practice, however, the complexity appears to be even lower ($O(1)$) due to very low values of the constant hidden behind such asymptotic estimates. Computations can be performed even more efficiently on a parallel machine.

3 Parallel Implementation

The performance of a parallel system is largely determined by the degree of concurrency of its processors. The identification of intrinsic parallelism in the method leads to our choice for data partitioning [7]. As mentioned earlier, the fast algorithm employs two groups of Fourier transforms which can be evaluated independently for each fixed radius r_l. Consequently their computations can be performed in parallel. Since each FFT usually engages lengthy computations, the computational granularity of each processor will be large and therefore very well suited for MIMD architectures.

Negative effects resulting from communication delays in a MIMD computer can be minimized by an efficient implementation. Mechanisms to reduce communication delays on message-passing architectures include: evenly distributed load balancing between processors, overlapping of communication and computations, reduced message lengths, and reduced frequency in exchanging messages. Often the above mechanisms are conflicting and, in practice, a tradeoff will define an efficient implementation. For more details on this issue and for an efficient parallel implementation, please see [5]-[6].

4 Applications

Due to page limitations, we show some results of applications. Further work on application to problems in fluid dyanmics is in progress. See also [1],[3], and [6].

4.1 Poisson Equation

Here we show some results when the following problem is solved using our fast algorithm. This is a ultimate test for validating superb performance of our algorithm because this problem presents discontinuities on the boundary conditions. The formulation is best described in polar coordinates

$$\begin{cases} \Delta u = f, & \text{in} \quad B = B(0;1), \\ \\ u = g, & \text{on} \quad \partial B, \end{cases}$$

where

$$f(re^{i\alpha}) = -4r^3\left(\cos^2\alpha\cdot\sin\alpha+\sin^3\alpha\right)\ \sin\left(1-r^2\right) - 8r\sin\alpha\cos\left(1-r^2\right),$$

and

$$g(e^{i\alpha}) = \begin{cases} 0, & \alpha\in(0,\pi), \\ 1, & \alpha\in(\pi,2\pi), \\ \frac{1}{2} & \alpha\in\{\pi,2\pi\}. \end{cases}$$

In this case we have the solution u given by

$$u(re^{i\alpha}) = \frac{1}{2} + \sin\left(r(1-r^2)\sin(\alpha)\right) - \frac{2}{\pi}\sum_{k=1}^{\infty} r^{2k-1}\frac{\sin(2k-1)\alpha}{2k-1}, \tag{4.1}$$

and the actual input data is expressed in Cartesian coordinates as

$$f(x,y) = -4(x^2y+y^3)\sin(1-x^2-y^2) - 8y\cos(1-x^2-y^2).$$

Figure 1 presents the actual solution of this problem obtained by expanding the summation in (4.1) up to the machine precision on each point of the $M\times N$ discretization of the domain $\overline{B(0;1)}$. The rapid variations in the points $(1,0)$ and $(-1,0)$ produce considerable errors when the Dirichlet problem is solved using 64 Fourier coefficients and 256 circles as shown in Figure 2. Nevertheless, the use of a larger number of Fourier coefficients for representing the solution preserves the locality

of the errors caused by rapid variations of the solution: Figure 3 contains the errors when increasing the number of coefficients to 128; and Figure 4 presents errors for 256 Fourier coefficients. Although the magnitude of the maximum error remains constant, the solution obtained by the algorithm converges globally. As an example, Figure 6 of [6] contains the errors when only observing the grid points in $\overline{B(0;1)}$ laying on the segment from $(0,-1)$ to $(0,1)$. In this case we say that the radial position is equal to -1 for the point $(0,-1)$, and it is 1 for the point $(0,1)$. The linear plot of the errors presented in Figure 6(a) of [6] shows convergence as the number of Fourier coefficients increases from 64 to 128, and to 256. The log-scaling in Figure 6(b) of [6] shows the rate of convergence. Global convergence can also be assessed by evaluating the global error without considering the points close to $(-1,0)$ and $(1,0)$. See [6] for detials on the relative errors in the domain $B(0;1)-(B_{0.01}(1,0)\cup B_{0.01}(-1,0))$. As the number of Fourier coefficients increases, convergence is observed.

5 Conclusions

Recently, progress has been made in the accurate and efficient evaluation of the singular integral operators based on some recursive relations in Fourier space, FFT and domain embedding method. Proof of the principle has been demonstrated here. We have also discussed the parallel algorithm in brief. By reformulating the inherently sequential recurrences present in the original algorithm, we were able to obtain a reduced amount of communication, and even message lengths depending only on the number of Fourier coefficients being evaluated. The implementation is very scalable in a parallel distributed environment and is virtually independent of the computer architecture. It only utilizes a linear neighbor-to-neighbor communication path which makes the algorithm very suitable for any architecture where a topology of the type ring or array of processors can be embedded. Some numerical results were presented to corroborate theoretical estimates. More detials can be found in [1]-[6]. Further work is in progress in this direction.

Acknowledgments

This material is based in part upon work supported by the Texas Advanced Research Program under Grant No. TARP-97010366-030.

References

[1] P. Daripa, *On applications of a complex variable method in compressible flows*, J. Comput. Phys., 88 (1990), pp. 337–361.

[2] ——, *A fast algorithm to solve nonhomogeneous Cauchy-Riemann equations in the complex plane*, SIAM J. Sci. Stat. Comput., 6 (1992), pp. 1418–1432.

[3] ——, *A fast algorithm to solve the Beltrami equation with applications to quasiconformal mappings*, J. Comput. Phys., 106 (1993), pp. 355–365.

[4] P. Daripa and D. Mashat, *Singular integral transforms and fast numerical algorithms*, Num. Algor., 18 (1998), pp. 133–157.

[5] L. Borges and P. Daripa, *A parallel version of a fast algorithm for singular integral transforms*. To appear in Num. Algor.

[6] L. Borges and P. Daripa, *A Fast Parallel Algorithm for the Poisson Equation on a Disk*. To appear.

[7] K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, McGraw-Hill, New York, NY, 1993.

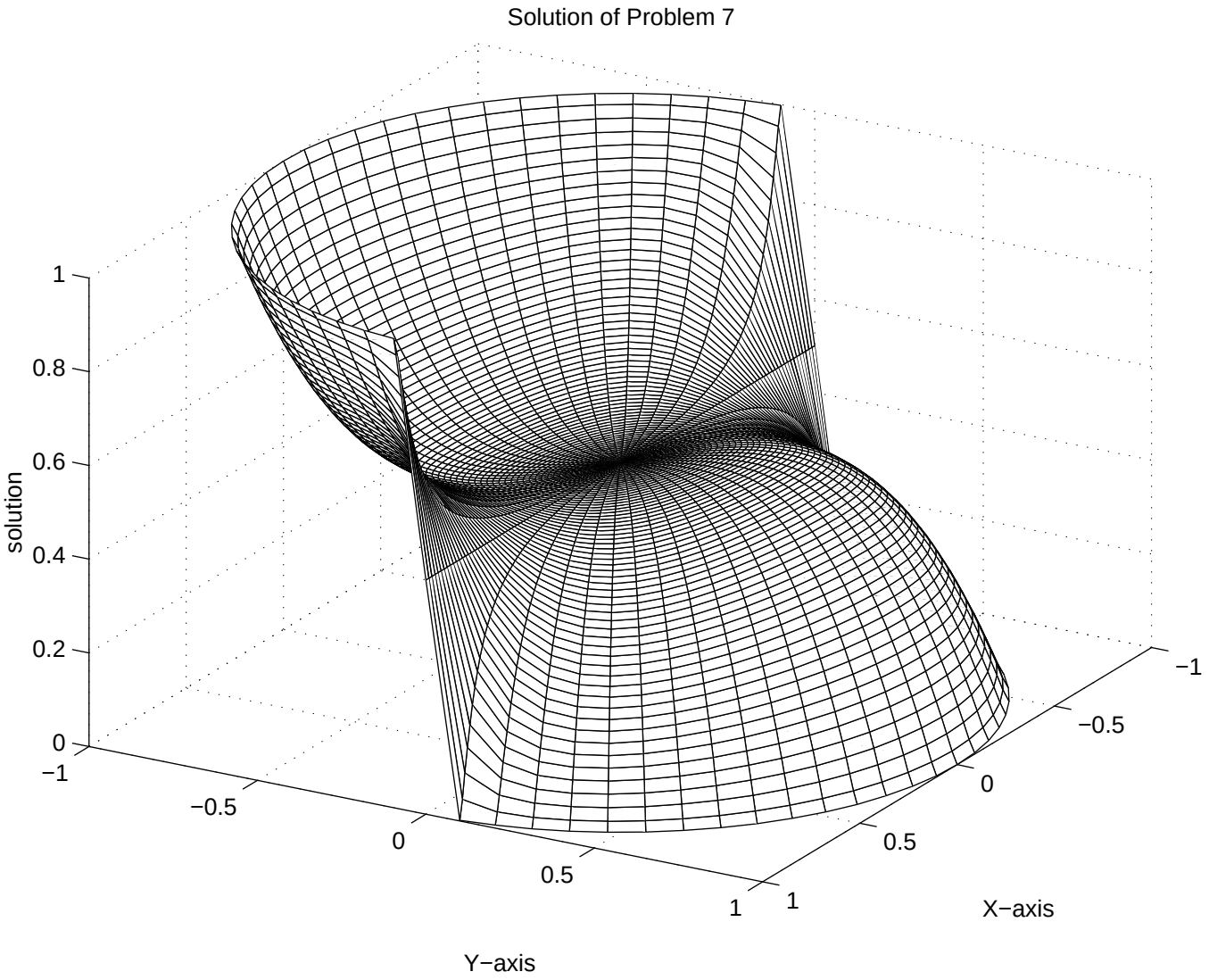

Figure 1: Analytical Solution.

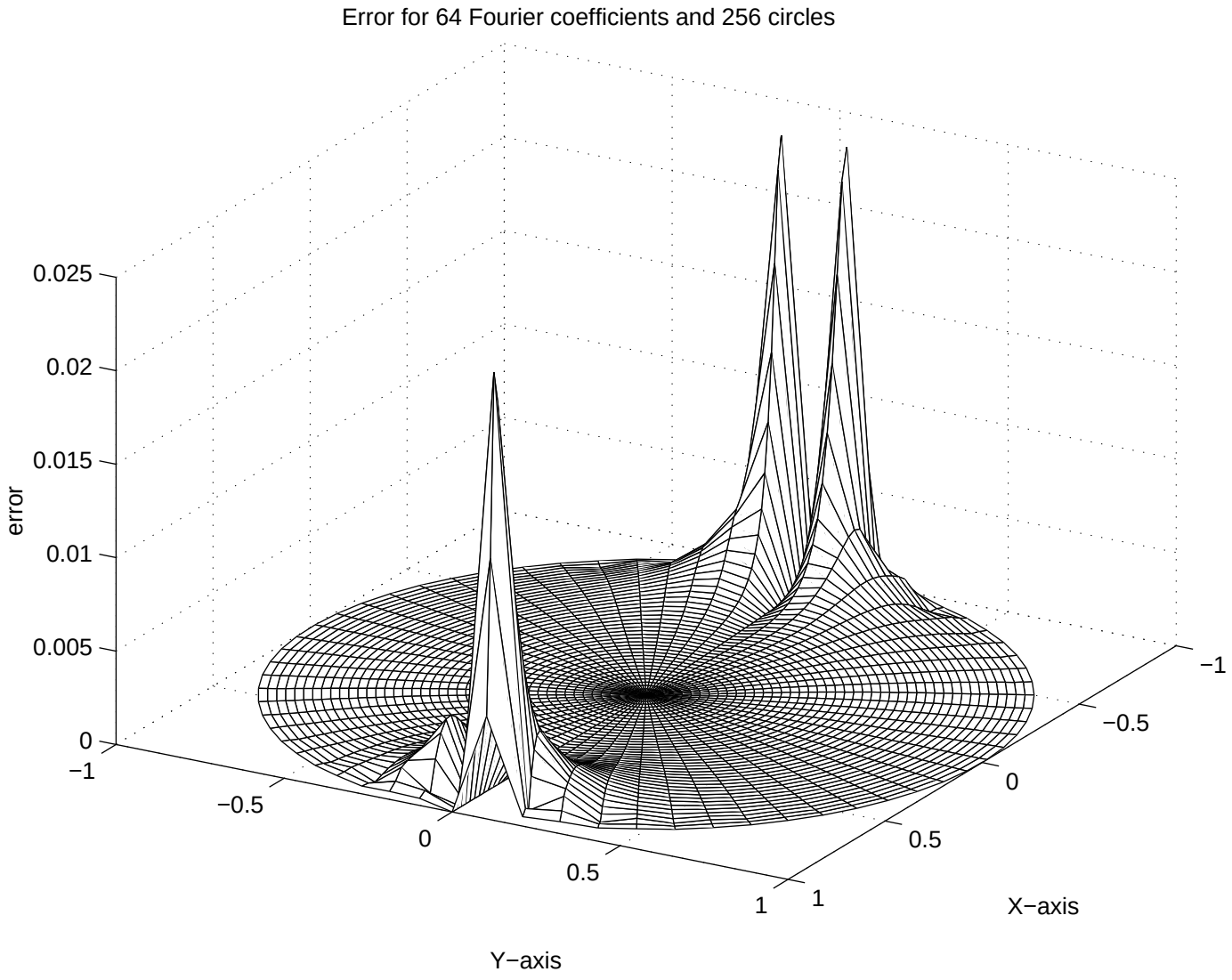

Figure 2: errors for 64 Fourier coefficients and 256 circles.

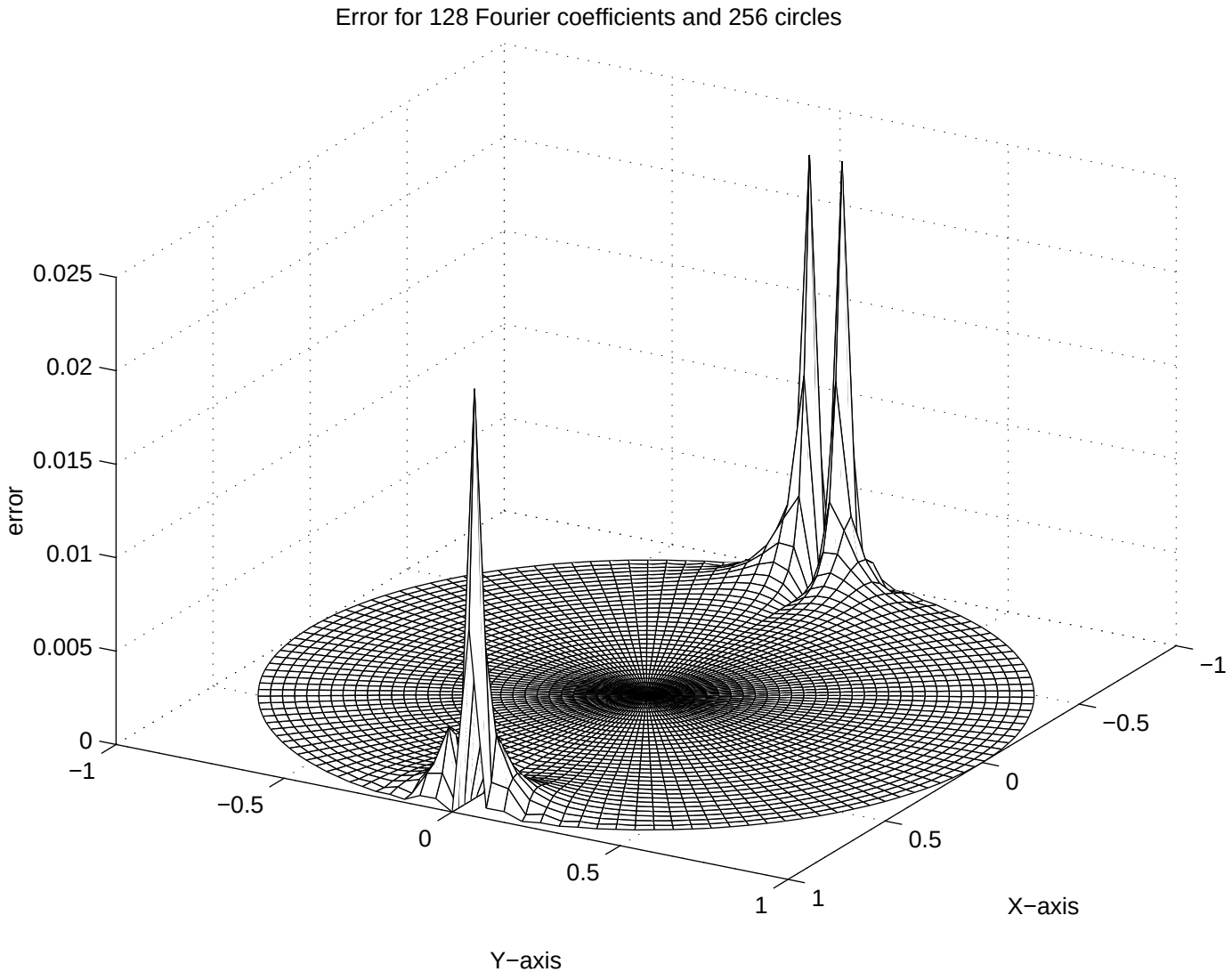

Figure 3: errors for 128 Fourier coefficients and 256 circles.

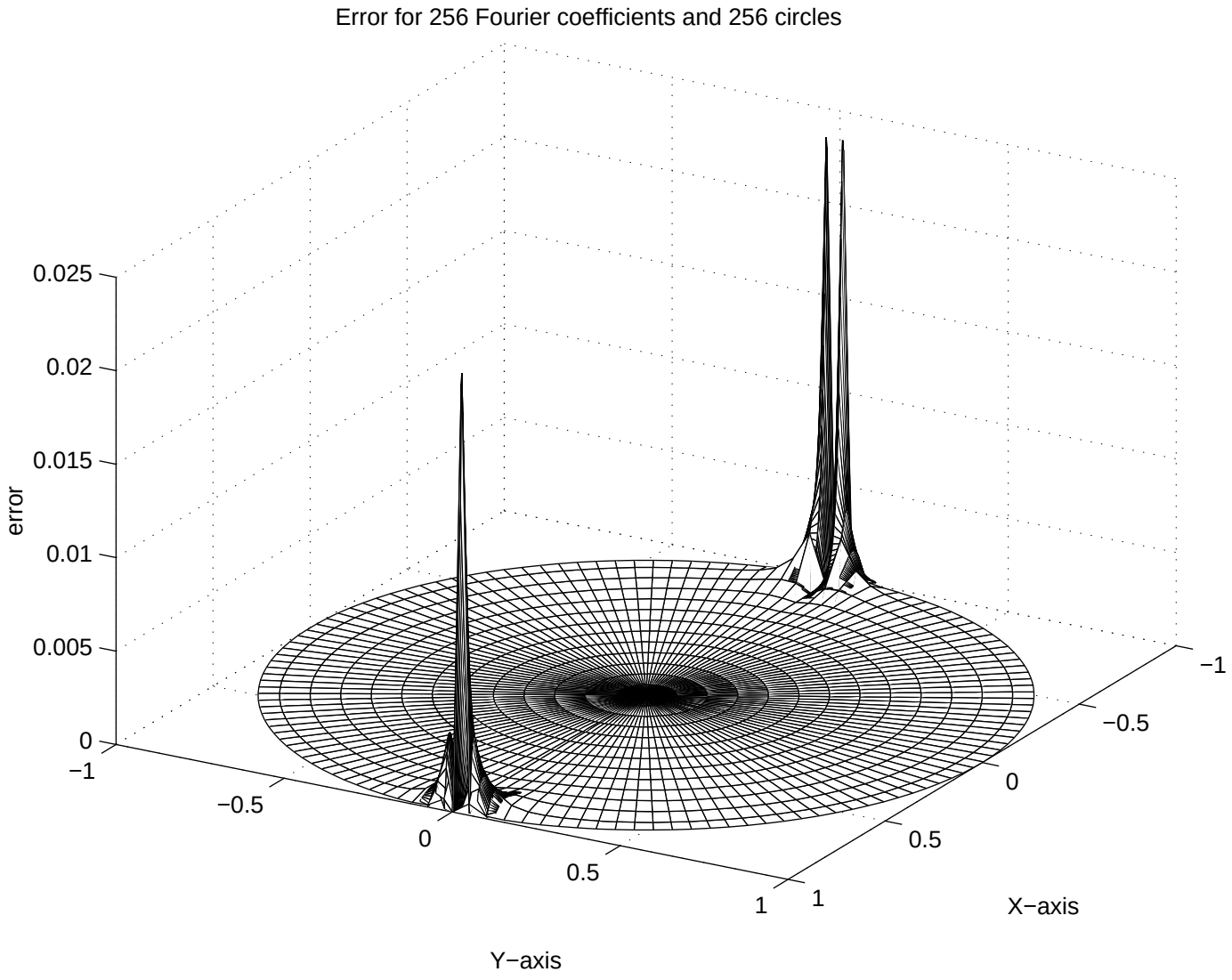

Figure 4: errors for 256 Fourier coefficients and 256 circles.