Решение уравнения Пуассона методом быстрого преобразования Фурье

Сотников А.Д, Северилов П.А.

Аннотация

В проекте описывается реализация решения двумерного уравнения Пуассона методом быстрого преобразования Фурье на языке программирования Python 3.

Ключевые слова: *уравнение Пуассона, быстрое преобразование Фурье (FFT), уравнения в частных производных, граничные условия.*

1 Преобразование Фурье

1.1 DTF

Вспомним, что мы хотим вычислить полином

$$A(x) = \sum_{j=0}^{n-1} a_j x^j$$

с границей степени n в точках $\omega_n^0, \omega_n^1, \omega_n^2, \ldots, \omega_n^{n-1}$, которые представляют собой n комплексных корней n–степени из единицы. Пусть полином A задан в форме коэффициентов: $a = (a_0, a_1, \ldots, a_{n-1})$. Определим результат y_k, где $k = 0,1,\ldots,n-1$, с помощью формулы

$$y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj}.$$

Вектор $y = (y_0, y_1, \ldots, y_{n-1})$ представляет собой ***дискретное преобразование Фурье.***

1.2 FFT

С помощью ***быстрого преобразования Фурье (FFT)***, вычисления можно проводить за $\Theta(n \lg n)$, где n – точная степень 2. В методе FFT применяется стратегия декомпозиции, в которой отдельно используются коэффициенты полинома $A(x)$ с четными и нечетными индексами:

$$A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_{n-2} x^{n/2-1},$$

$$A^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \cdots + a_{n-1} x^{n/2-1}.$$

Такая декомпозиция является основой следующего рекурсивного алгоритма FFT:

Algorithm 1 Recursive-FFT(a)

```
1: n = a.length        // n является степенью 2
2: if n == 1 then
3:     return a
4: end if
5: ω_n = e^{2πi/n}
6: ω = 1
7: a^[0] = ((a_0, a_2, ..., a_{n-2}))
8: a^[1] = ((a_1, a_3, ..., a_{n-1}))
9: y^[0] = Recursive-FFT(a^[0])
10: y^[1] = Recursive-FFT(a^[1])
11: for k = 0 to n/2 − 1 do
12:     y_k = y_k^[0] + ω y_k^[1]
13:     y_{k+(n/2)} = y_k^[0] − ω y_k^[1]
14:     ω = ω ω_n
15: end for
16: return y        // Считаем, что y–вектор-столбец
```

Видно, что в строчках 11-15 алгоритм дважды вычисляет выражение $\omega_n^k y_k^{[1]}$. Можно изменить цикл так, чтобы вычислять это значение только один раз, сохраняя его во временной переменной t. Такой прием называется ***преобразованием бабочки***. Иллюстрацию работы алгоритма можно посмотреть на рис.1.

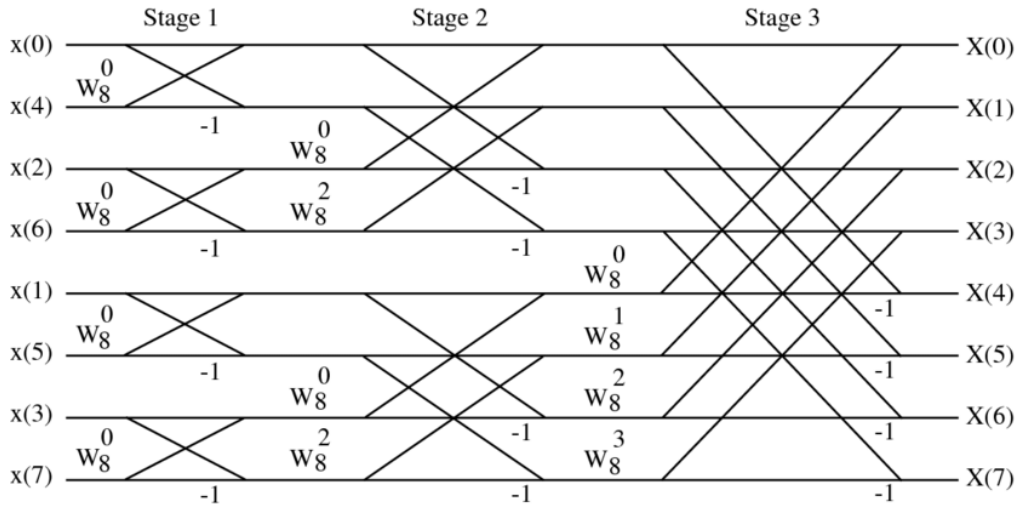

Рис. 1: Схема FFT с преобразованием бабочки.

С использованием данной операции рекурсивный метод можно ускорить и сделать итеративным следующим образом:

Algorithm 2 Iterative-FFT(a)

```
1: n = a.length        // n является степенью 2
2: for k = 0 to n − 1 do
3:     A[rev(k)] = a_k
4: end for
5: for s = 1 to lg n do
6:     m = 2^s
7:     ω_m = e^{2πi/m}
8:     for k = 0 to n − 1 by m do
9:         ω = 1
10:        for j = 0 to m/2 − 1 do
11:            t = ωA[k + j + n/2]
12:            u = A[k + j]
13:            A[k + j] = u + t
14:            A[k + j + m/2] = u − t
15:            ω=ωω_m
16:        end for
17:    end for
18: end for
19: return A
```

Полученный алгоритм можно ускорить, распараллелив вычисления на $\lg n$ этапов, на каждом из которых выполняется $n/2$ операций бабочки.

2 Решение уравнения Пуассона с помощью FFT

Быстрое преобразование Фурье может быть использовано для решения эллиптических дифференциальных уравнений в частных производных в измерениях разной размерности. Основная идея очень проста. Рассмотрим одномерное уравнение

$$\frac{d^2\varphi}{dx^2} = \rho(x).$$

Выразим f и ρ через их преобразование Фурье:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int g(k) e^{ikx} dk$$

$$\rho(x) = \frac{1}{\sqrt{2\pi}} \int \sigma(k) e^{ikx} dk.$$

Решение задается обратным преобразованием

$$f(x) = -\frac{1}{\sqrt{2\pi}} \int \frac{\sigma(k)}{k^2}\ e^{ikx} dk.$$

Нужно наложить граничные условие, а также указать, что нужно делать, если $k = 0$ под знаком интеграла.

2.1 Граничные условия и типы преобразований

Граничные условия определяют соответствующий тип преобразования Фурье для решения задачи. Некоторые задачи, которые решал Фурье, требуют синус-преобразования, другие – экспоненциального.

Рассмотрим область $0 \le x \le L$ в одномерном случае. Выберем решетку из N одинаково расположенных точек $x_n = nL/N,\ n = \overline{0,N-1}$.

Коэффициенты комплексного преобразования Фурье функции $f(x)$ имеют вид

$$g_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \omega^{kn} f_n, \quad \omega = e^{2\pi i/N}.$$

Обратное преобразование

$$f_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{-nk} g_k$$

будет периодическим по x_n с периодом L. Поэтому, комплексное преобразование Фурье подходит для задач, которые удовлетворяют периодическим граничным условиям.

Если задача включает в себя граничные условия Дирихле, например, $f(0) = f(L) = 0$, то целесообразнее использовать синус-преобразование Фурье

$$f_n = \sqrt{\frac{2}{N}} \sum_{k=1}^{N-1} sin\Big(\frac{\pi nk}{N}\Big) g_k.$$

Если задача включает в себя граничные условия Неймана, целесообразно использовать косинус-преобразование Фурье. На $N+1$ точках решение имеет вид

$$f_n = \frac{1}{\sqrt{2N}}[g_0 + (-1)^n g_N] + \sqrt{\frac{2}{N}} \sum_{k=1}^{N-1} cos\Big(\frac{\pi nk}{N}\Big) g_k.$$

Обратим внимание на то, что косинус- и синус-преобразования – это не действительная и мнимая части сложного экспоненциального преобразования. Это связано с тем, что функции $sin(x), cos(x)$ и e^x по отдельности образуют полные множества, хотя и с разными граничными условиями. Базовый фазовый коэффициент для синус- и косинус-преобразований равен π/N по сравнению с $2\pi/N$ для экспоненциального, т.к. для них требуется вдвое больше точек на сетке.

2.2 Уравнение Пуассона в 2D

Рассмотрим дискретную форму ***уравнения Пуассона***

$$\Big(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\Big)\varphi(x,y) \simeq \frac{1}{h^2}[\varphi_{j+1,k} + \varphi_{j-1,k} + \varphi_{j,k+1} - 4\varphi_{j,k}] = -\rho_{j,k}$$

на сетке $N \times N$ в области $0 \leq x,y \leq 1$ с заданной плотностью заряда. Пусть точечный заряд находится в центре рассматриваемой области.

Для простоты, наложим периодические граничные условия, чтобы можно было использовать экспоненциальное FFT для решения.

Дискретное преобразование Фурье является линейной операцией, поэтому мы можем применять его отдельно в направлениях x и y, и не имеет значения, в каком порядке выполняются преобразования.

Коэффициенты FFT в 2D:

$$\tilde{\varphi}_{m,n} = \frac{1}{N}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}\omega^{mj+nk}\varphi_{j,k},$$

$$\tilde{\rho}_{m,n} = \frac{1}{N}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}\omega^{mj+nk}\rho_{j,k}.$$

Обратные преобразования:

$$\varphi_{j,k} = \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}\omega^{-jm-kn}\tilde{\varphi}_{j,k},$$

$$\rho_{j,k} = \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}\omega^{-jm-kn}\tilde{\rho}_{j,k}.$$

Подставляя эти выражения в дискретное уравнение и приравнивая коэффициенты ω^{-mj-nk}, получим

$$\frac{1}{h^2}[\omega^m + \omega^{-m} + \omega^n + \omega^{-n} - 4]\tilde{\varphi}_{m,n} = -\tilde{\rho}_{m,n},$$

что легко решается:

$$\tilde{\varphi}_{m,n} = \frac{h^2\tilde{\rho}_{m,n}}{4 - \omega^m - \omega^{-m} - \omega^n - \omega^{-n}}.$$

Обратное преобразование Фурье дает искомый потенциал.

3 Реализация

Решение задачи реализовано на языке Python 3 с помощью библиотеки *NumPy* и было протестировано на следующих трех уравнениях:

$$\Delta u = 8\pi^2 cos(4\pi y)(cos(4\pi x) - sin(4\pi x)) - \\ - 16\pi^2(sin(4\pi x)cos^2(2\pi y) + sin^2(2\pi x)cos(4\pi y)) \quad (1)$$

$$\Delta u = 6xy(1-y) - 2x^3 \quad (2)$$

$$\Delta u = -2(2y^3 - 3y^2 + 1) + 6(1-x^2)(2y-1) \quad (3)$$

Соответствующие аналитические решения, на которых проверялась точность обозреваемого в данной работе метода:

$$u = sin^2(2\pi x)cos(4\pi y) + sin(4\pi x)cos^2(2\pi y)$$

$$u = y(1-y)x^3$$

$$u = (1-x^2)(2y^3 - 3y^2 + 1)$$

Во всех уравнениях граничные условия одинаковы и имеют вид $0 \leq x,y \leq 1$.

4 Результаты

Рассматриваемая ошибка – среднее квадратичное (MSE) между значениями сетки на точном решении и полученном с помощью FFT.

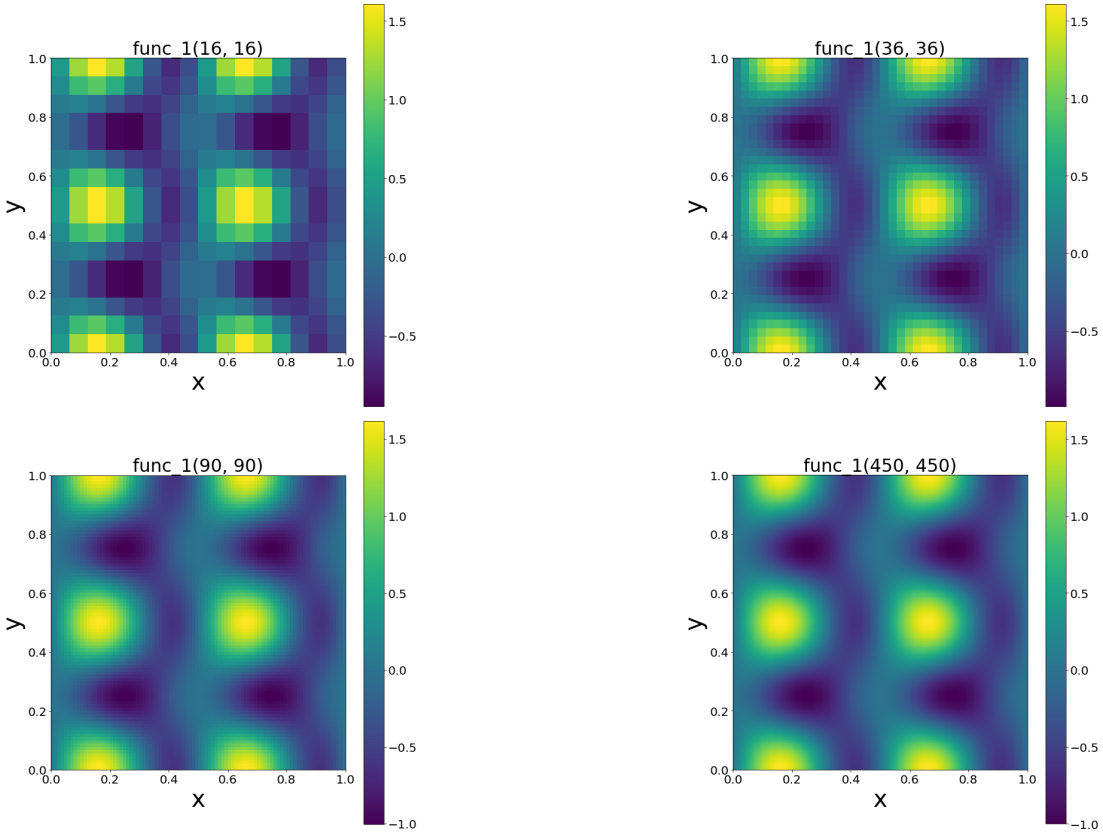

Рис. 2: График зависимости решения уравнения (1) от числа узлов в создаваемой сетке.

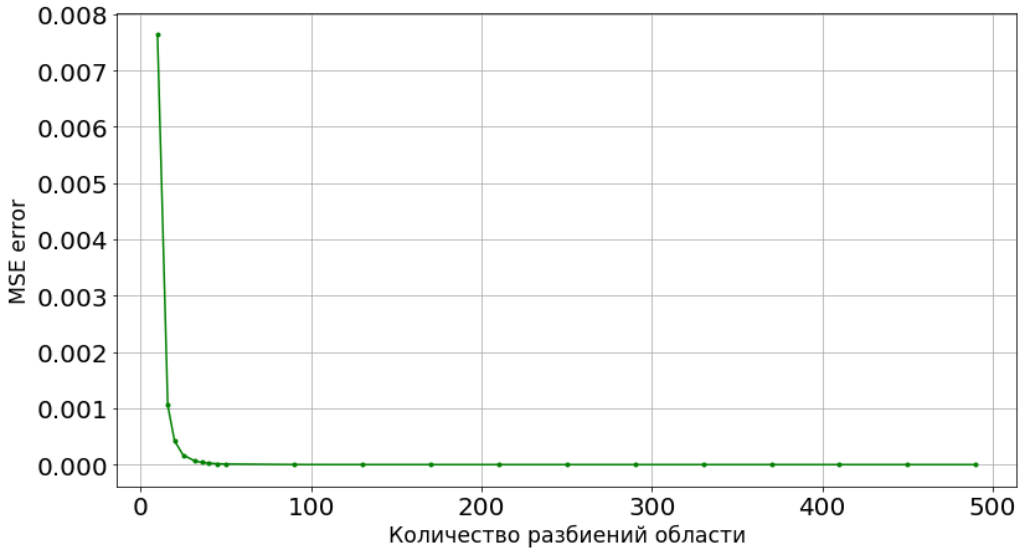

Рис. 3: График зависимости ошибки от количества разбиений области для уравнения (1).

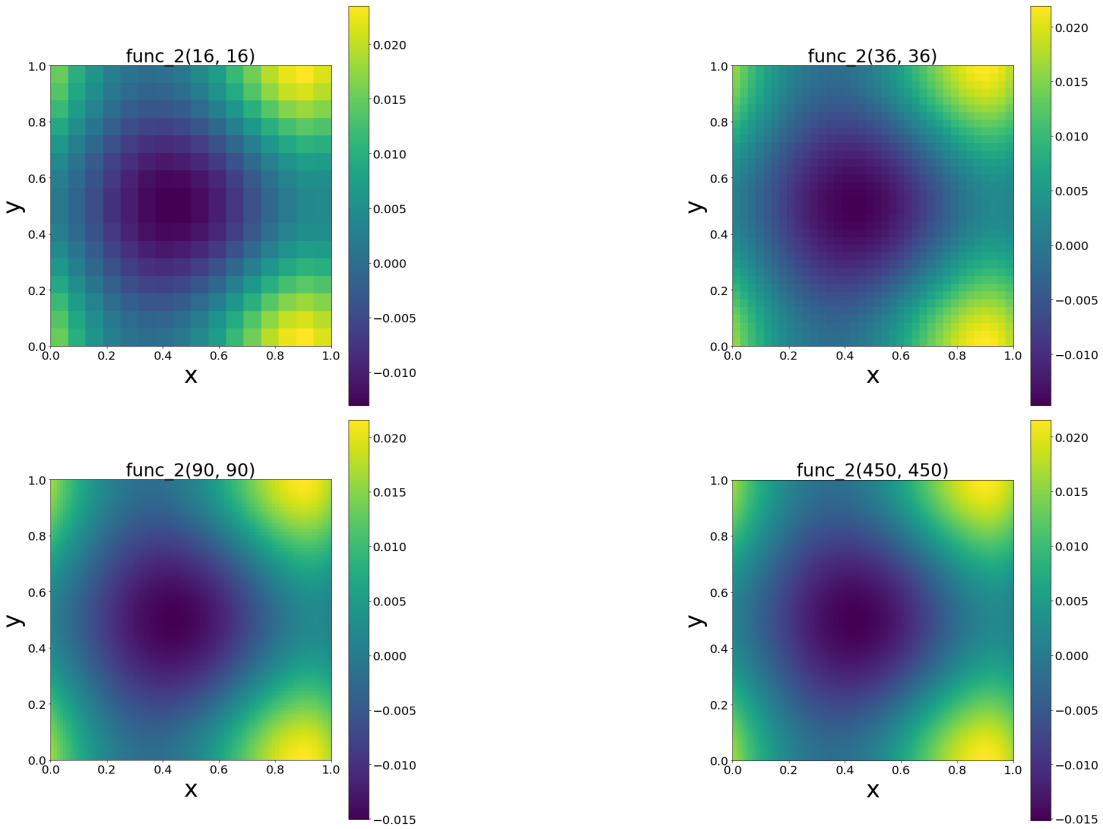

Рис. 4: График зависимости решения уравнения (2) от числа узлов в создаваемой сетке.

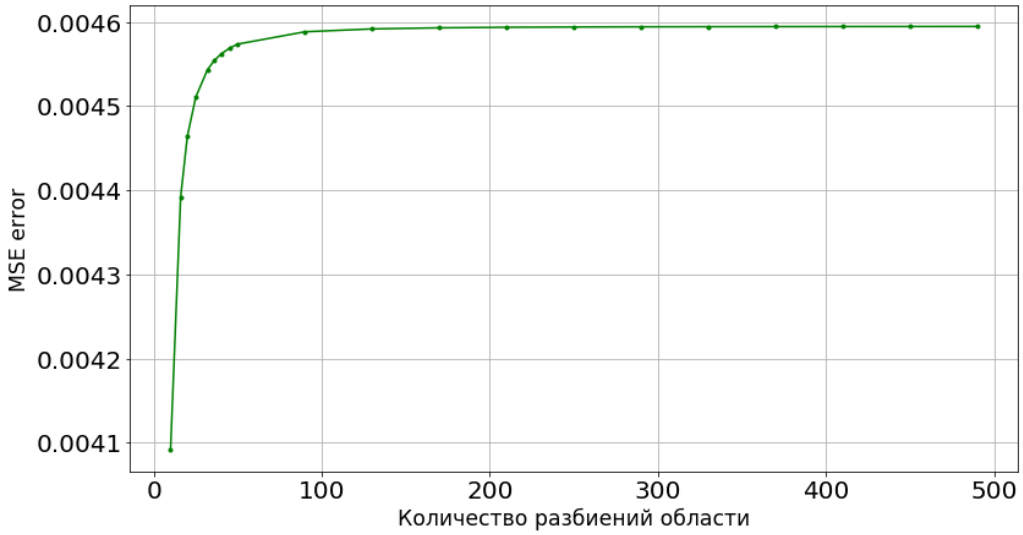

Рис. 5: График зависимости ошибки от количества разбиений области для уравнения (2).

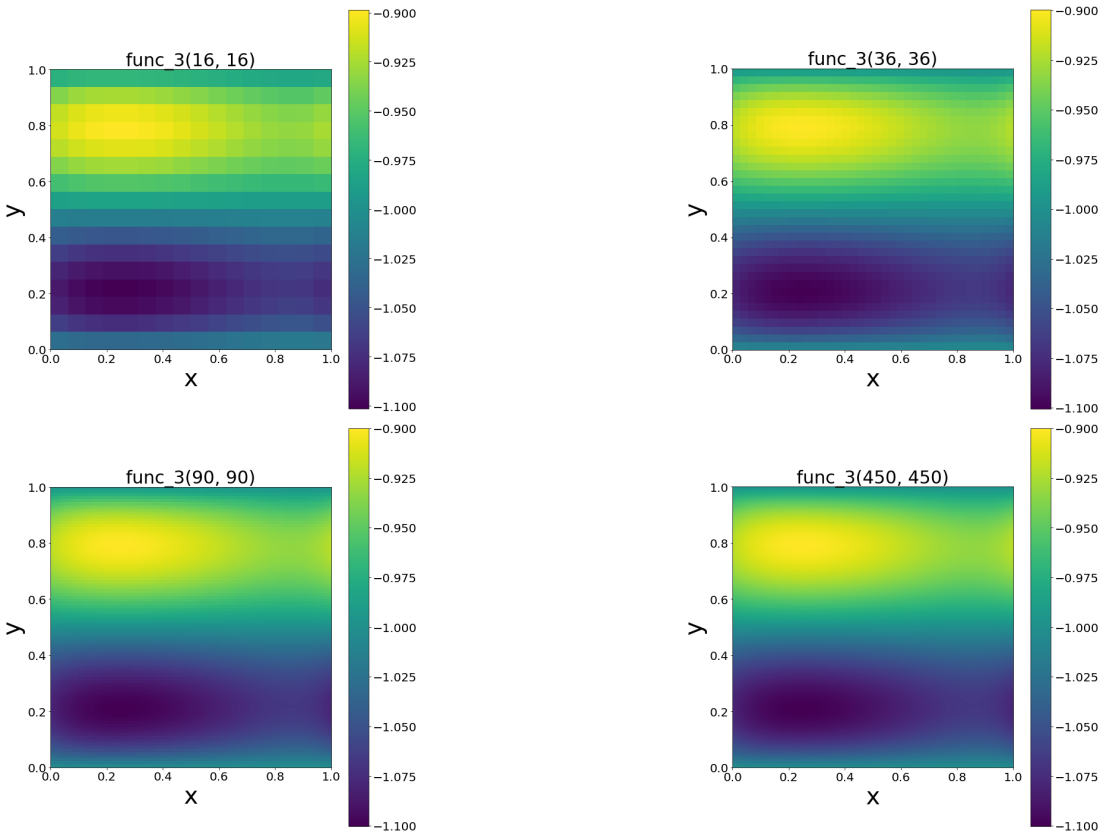

Рис. 6: График зависимости решения уравнения (3) от числа узлов в создаваемой сетке.

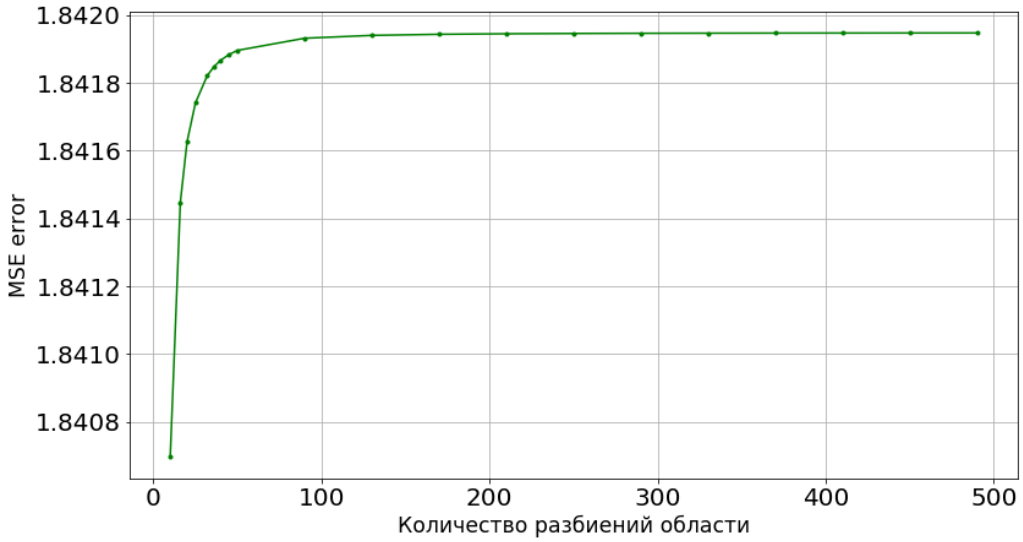

Рис. 7: График зависимости ошибки от количества разбиений области для уравнения (3).

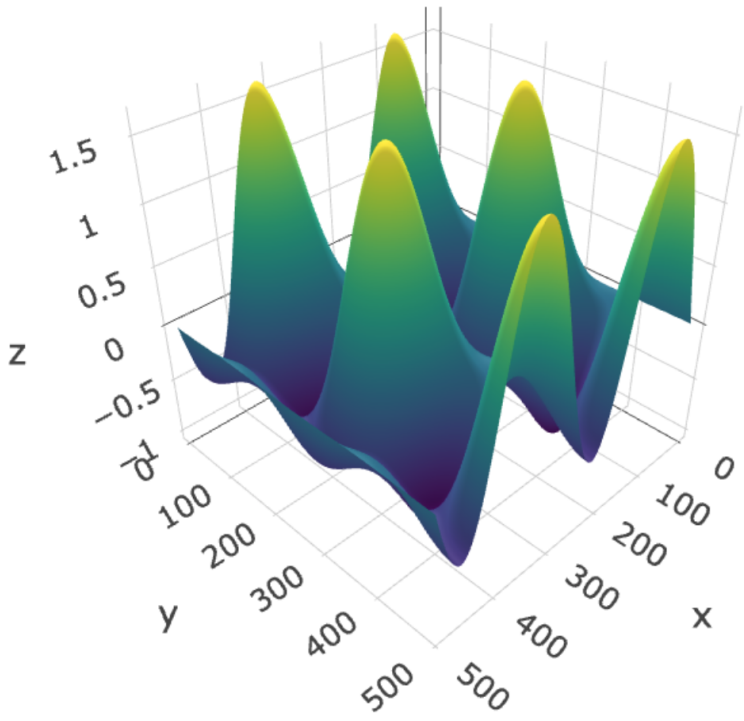

Рис. 8: Иллюстрация решения уравнения (1) в *3D*.

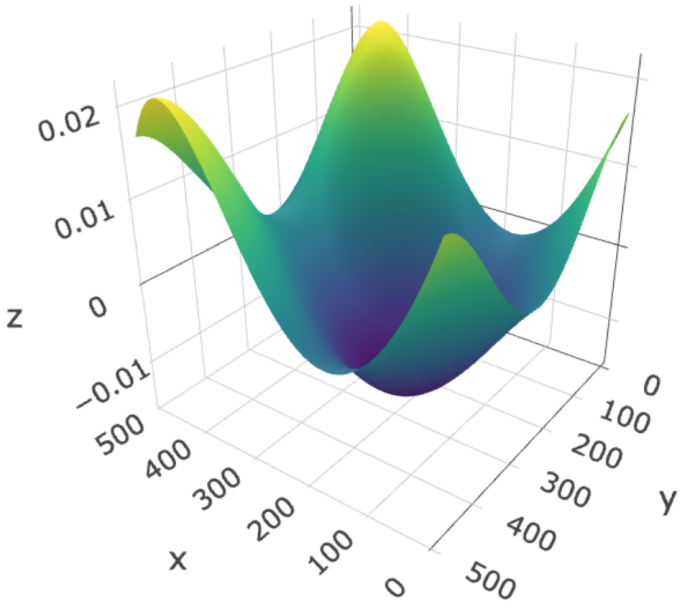

Рис. 9: Иллюстрация решения уравнения (2) в *3D*.

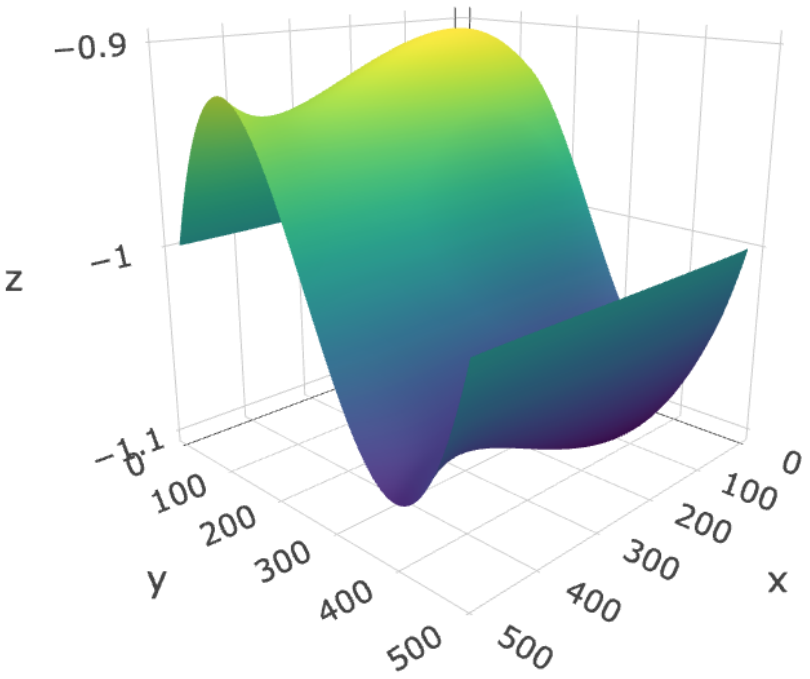

Рис. 10: Иллюстрация решения уравнения (3) в *3D*.

5 Заключение

В данной работе нами был рассмотрен способ решения двумерного уравнения Пуассона методом быстрого преобразования Фурье. Анализ ошибки говорит о том, что данный метод крайне точно решает поставленную задачу. При увеличении количества разбиений на сетке ошибка мало менялась.

Список литературы

[1] *Thomas Cormen, Charles E.Leiserson, Ronald L.Rivest, Clifford Stein N.* Introduction to Algorithms // MIT, 3rd edition, 2013, Pp.952-962.

[2] *Kasper Ornstein Mecklenburg N.*Creating Visual Effects by Solving Partial Differential Equations in Real-Time // Bachelor's thesis, 2014

[3] *Dr. Richard J. Gonsalves N.* Solving Poisson's Equation using the FFT // SUNY Buffalo State College, 2009