ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. М. В. Ломоносова

Факультет фундаментальной механики и математики

Сравнительный анализ численных методов восстановления поверхностных функций

Студент 431 группы _____ А. К. Новиков

Преподаватель _____ А. А. Корнев

Москва 2020

Содержание

1. Введение

В данной работе приведён сравнительный анализ численных методов восстановления поверхностных функций. В качестве исследуемых методов выступают:

1) Метод дискретного преобразования Фурье

2) Метод кубических интерполяционных сплайнов

3) Метод наилучшего квадратичного приближения

Цель данной работы: сравнить определённые выше методы по точности восстановления, сложности реализации и вычисления, а также с точки зрения удобства использования и области применения.

Для этого будем использовать некоторый набор известных функций. Функции предполагаются непрерывными на всей области определения. Будем генерировать по известной функции конечный набор значений в некоторых узлах разбиения двумерной области определения, над которой необходимо восстановить двумерную поверхность, заданную этой функцией. Назовём эти узлы известными, а остальные узлы разбиения - неизвестными. Далее применим каждый из исследуемых алгоритмов восстановления, чтобы получить предполагаемые алгоритмом значения функции в неизвестных узлах. После, так как функцию мы заранее задали, вычислим истинные значения функции в неизвестных узлах и сравним с соответствующими значениями, построенными алгоритмом, в этих узлах.

Проведём описанную выше процедуру при различных количествах узлов разбиения и для различных функций, каждая из которых будет иметь свои недостатки с точки зрения трудности восстановления, запишем все полученные результаты в виде таблицы и определим по полученным данным несколько основных характеристик для каждого из исследуемых методов, а далее на основе вычисленных характеристик сравним исследуемые методы между собой, и опредеим примущества и недостатки каждого их них.

2. Постановка задачи

Пусть имеется некоторая односвязная двумерная область $\Omega \subset \mathbb{R}^2$. На данной области задана сетка узлов $(x_i, y_i) \in \Omega$, $i, j \in I \subset \mathbb{N}$, т.ч. $\|I\| < \infty$, в каждом из которых находится значение некоторой неизвестной функции $f(x, y) \in C[\Omega]$. Часть значений потеряна. Необходимо восстановить значения этой функции в потерянных узлах при помощи некоторого численного метода.

Для оценки точности работы алгоритма, реализующего один из исследуемых численных методов, предлагается использовать заранее известную функцию $f(x, y)$. Сначала полностью заполняются истинные значения данной функции во всех узлах разбиения области Ω, затем случайно выбираются некоторые узлы, значения в которых алгоритм будет считать потерянными. Затем применяем алгоритм, подавая в качестве входных данных значения функции, в узлах, выбранных, как известные. В качестве выходных данных получим предполагаемые значения функции $f(x, y)$ в тех узлах, которые были выбраны, как неизвестные.

Обозначим множество неизвестных значений $\Gamma = \{(x_i, y_j) |$ точка (x_i, y_j) была выбрана, как неизветсная$\}$, а его мощность, то есть количество неизвестых значений $\|\Gamma\| < \infty$, множество всех значений обозначим Λ, а его мощность $\|\Lambda\| < \infty$. Значения, предложенные алгоритмом обозначим $\widetilde{f}(x_i, y_j)$, а площадь области Ω обозначим $S(\Omega) < \infty$.

Вычислим скорость сходимости алгоритма. Данная величина будет характеризоваться константой p, определённой следующим образом:

$$\begin{gathered}\frac{1}{\|\Gamma\|}\sum_{(x_i,x_j)\in\Gamma}|f(x_i,y_i)-\widetilde{f}(x_i,y_j)|\approx\left(\frac{S(\Omega)}{\|\Lambda\|}\right)^p=h^p\\ p=\log_h\left(\frac{1}{\|\Gamma\|}\sum_{(x_i,x_j)\in\Gamma}|f(x_i,y_i)-\widetilde{f}(x_i,y_j)|\right)\end{gathered}\tag{1}$$

Также вычислим сложность реализованного алгоритма, подсчитав количество операций, необходимых для его выполнения.

При помощи значений (5) скорости сходимости алгоритма, а также величины сложности алгоритма можно будет судить о качестве применённого метода, а также сравнивать этот метод с другими.

3. Функции

Пусть область Ω представляет из себя прямоугольник на плоскости:

$$\Omega=\{(x,y)\in\mathbb{R}^2|a_x\leqslant x\leqslant b_x,a_y\leqslant y\leqslant b_y|a_x,b_x,a_y,b_y\in\mathbb{R}\}\tag{2}$$

Возьмём в качестве функций, значения которых необходимо будет восстанавливать, следующие функции над обыкновенным квадратом, то есть при $a_x=-1$, $b_x=1$, $a_y=-1$ и $b_y=1$:

$$f_1(x,y)=(x^2-1)(y^2-1)e^{-xy}\tag{3}$$

Изобразим поверхность, которую задаёт функция (**??**):

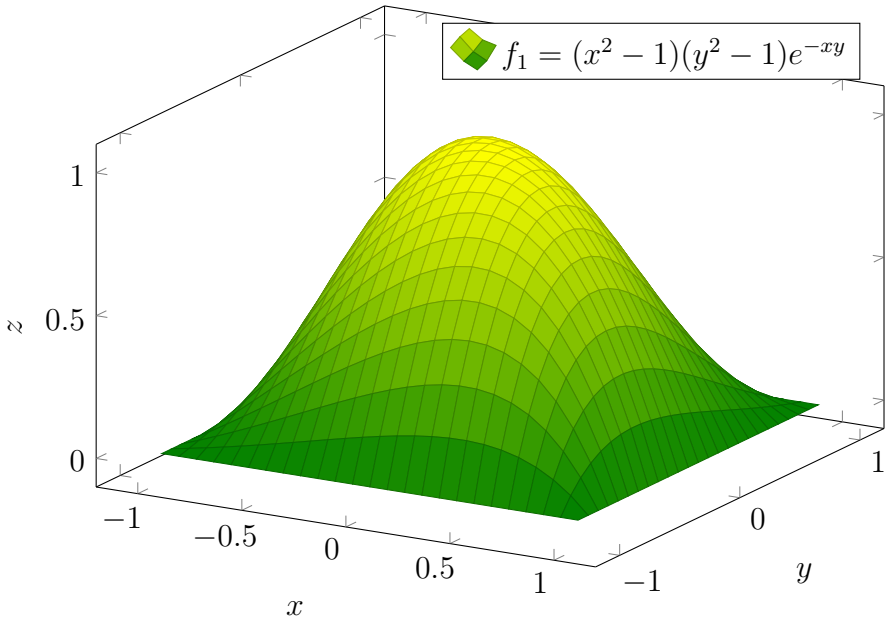

Рис. 1.

$$f_2(x,y)=\sin\pi x+\sin 2\pi x+\sin 50\pi x\tag{4}$$

Поверхность, которую задаёт функция (**??**) при $a_x = -1$, $b_x = 1$, $a_y = -1$ и $b_y = 1$ представляет из себя расширение кривой $\varphi(x) \equiv f_2(x, y)$ по оси абсцисс, так как функция (**??**) не зависит от y. В силу невозможности корректно отобразить данную поверхность, изобразим её сечение плоскостью $y = 0$:

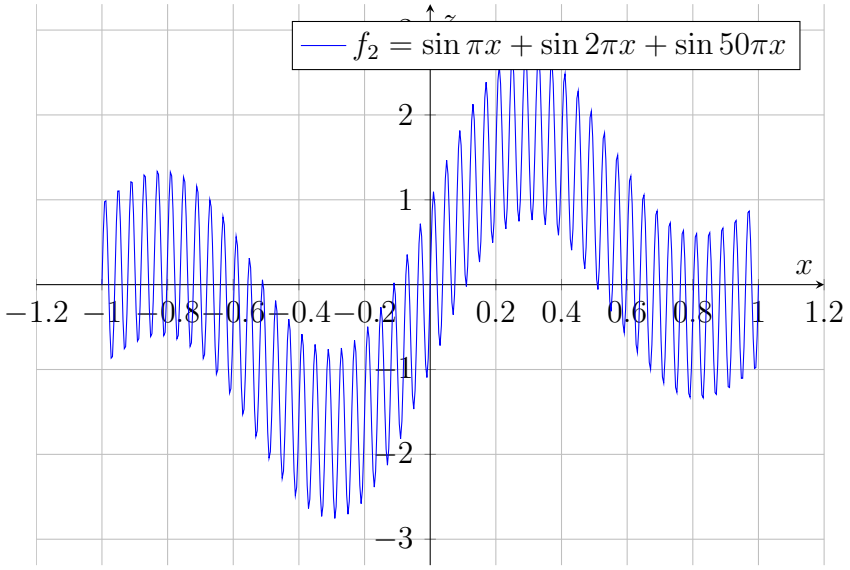

Рис. 2.

$$f_3(x, y) = (x^2 - 1)(y^2 - 1)(\text{arctg}(y) \cdot e^x) \tag{5}$$

Изобразим поверхность, которую задаёт функция (**??**):

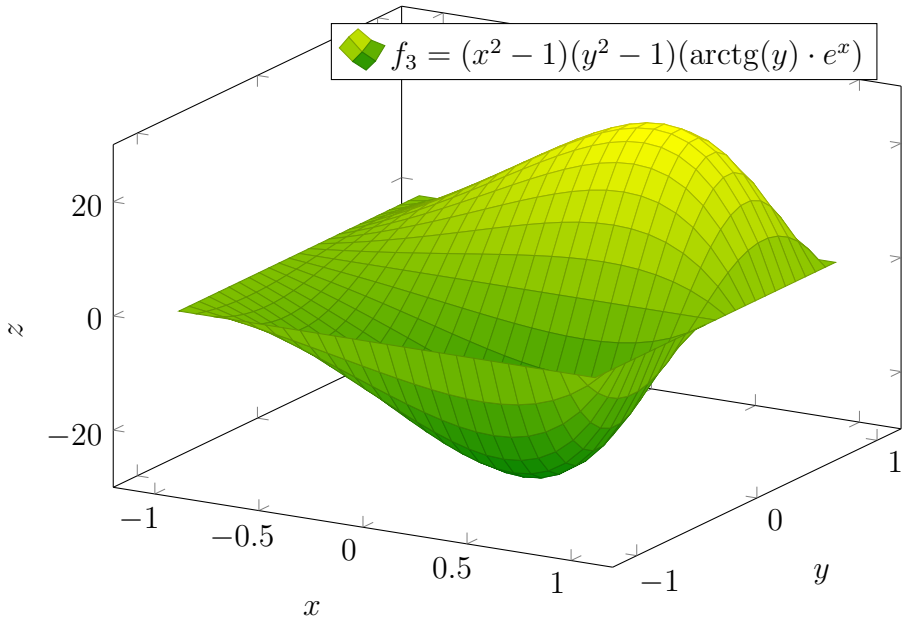

Рис. 3.

$$f_4(x, y) = (x^2 - 1)(y^2 - 1)(x^2 - 0.01)(y^2 - 0.01)e^{xy} \tag{6}$$

Изобразим поверхность, которую задаёт функция (**??**):

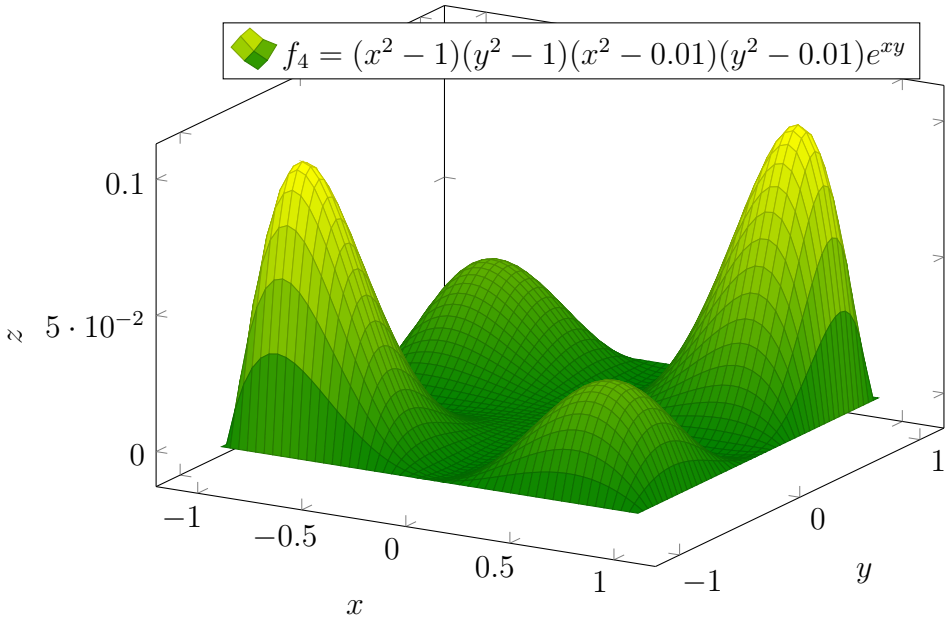

Рис. 4.

$$f_5(x, y) = (x^2 - 1)(\cos(x) - \ln(y^2 + 3)) \tag{7}$$

Изобразим поверхность, которую задаёт функция (**??**):

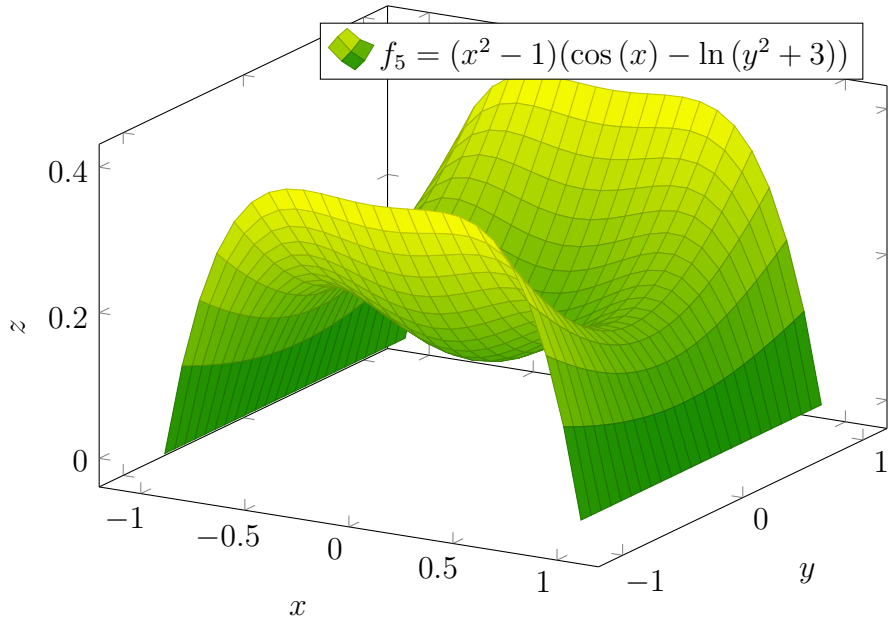

Рис. 5.

4. Дискретное преобразование Фурье

4.1. Описание метода

Дискретное преобразование Фурье можно применять, если на отрезке $[0,1]$ задана 1-периодическая функция $g(x)$, то есть $g(0) = g(1) = 0$. В качестве базиса можно взять систему

$$\{1, \cos 2\pi x, \sin 2\pi x, \cdots, \cos 2\pi m x, \sin 2\pi m x\} \tag{8}$$

и построить ряд Фурье для $g(x)$.

Если функция $g(x)$ является $(b-a)$-периодической, то есть $g(a) = g(b) = 0$, то рассмотрим $g_j = g(a + jh)$, $h = \frac{b-a}{n}$, $0 \leqslant j \in \mathbb{Z} \leqslant n, n \in \mathbb{N}$. На пространстве дискретных функций g_j, $g_0 = g_n = 0$ определим скалярное произведение

$$(g, p) = \frac{1}{n} \sum_{j=1}^{n-1} g_j p_j \tag{9}$$

Тогда система

$$\{\varphi_j^{(m)} = C \sin \frac{\pi m j h}{b-a},\ m = 1, \cdots, n-1,\ j = 0, \cdots, n\} \tag{10}$$

ортогональна. Более того, система (8) является ещё и ортонормальной при некотором значении константы C.

Покажем сначала ортогональность:

$$(\varphi^{(m)}, \varphi^{(k)}) = \frac{1}{n} \sum_{j=1}^{n-1} (\varphi_j^{(m)} \cdot \varphi_j^{(k)}) = \frac{1}{n} \sum_{j=1}^{n-1} \left(C \sin \frac{\pi m j h}{b-a} \cdot C \sin \frac{\pi k j h}{b-a} \right) =$$

$$= \frac{C^2}{n} \sum_{j=1}^{n-1} \left(\sin \frac{\pi m j h}{b-a} \sin \frac{\pi k j h}{b-a} \right) = \frac{C^2}{2n} \sum_{j=1}^{n-1} \left(\cos \frac{\pi (m-k) j h}{b-a} - \cos \frac{\pi (m+k) j h}{b-a} \right) =$$

$$= \frac{C^2}{2n} \sum_{j=1}^{n-1} Re \left[exp\Big(\frac{i\pi (m-k) j h}{b-a}\Big) - exp\Big(\frac{i\pi (m+k) j h}{b-a}\Big) \right] =$$

$$= \frac{C^2}{2n} \sum_{j=1}^{n-1} Re \left[exp\Big(\frac{i\pi m j h}{b-a}\Big) \left(exp\Big(\frac{-i\pi k j h}{b-a}\Big) - exp\Big(\frac{i\pi k j h}{b-a}\Big) \right) \right] =$$

$$= \frac{C^2}{2n} \sum_{j=1}^{n-1} \left(\cos \frac{\pi m j h}{b-a} \cdot Re \left[exp\Big(\frac{-i\pi k j h}{b-a}\Big) - exp\Big(\frac{i\pi k j h}{b-a}\Big) \right] \right) =$$

$$= \frac{C^2}{2n} \sum_{j=1}^{n-1} \left(\cos \frac{\pi m j h}{b-a} \cdot Re \left[\cos \frac{\pi k j h}{b-a} - i \sin \frac{\pi k j h}{b-a} - \cos \frac{\pi k j h}{b-a} - i \sin \frac{\pi k j h}{b-a} \right] \right) =$$

$$= \frac{C^2}{2n} \sum_{j=1}^{n-1} \left(\cos \frac{\pi m j h}{b-a} \cdot Re \left[-2i \cdot \sin \frac{\pi k j h}{b-a} \right] \right) = \frac{C^2}{2n} \sum_{j=1}^{n-1} \left(\cos \frac{\pi m j h}{b-a} \cdot 0 \right) =$$

$$= 0,\ \forall\ 1 \leqslant k \neq m \leqslant n-1 \tag{11}$$

Таким образом, из (9) следует ортогональность системы (8). Для доказательства ортонормальности системы (8) необходимо найти константу C так, чтобы:

$$(\varphi^{(m)}, \varphi^{(m)}) = 1, \ \forall\ 1 \leqslant m \leqslant n-1 \tag{12}$$

$$(\varphi^{(m)}, \varphi^{(m)}) = \frac{1}{n}\sum_{j=1}^{n-1}(\varphi_j^{(m)} \cdot \varphi_j^{(m)}) = \frac{1}{n}\sum_{j=1}^{n-1} C^2 \sin^2\frac{\pi m j h}{b-a} = \frac{C^2}{2n}\sum_{j=1}^{n-1}\left(1 - \cos\frac{2\pi m j h}{b-a}\right) =$$

$$= \frac{C^2}{2n}\sum_{j=1}^{n-1}\left(1 - \cos\frac{2\pi m j}{n}\right) = \frac{C^2(n-1)}{2n} - \frac{C^2}{2n}\sum_{j=1}^{n-1}\cos\frac{2\pi m j}{n} = \frac{C^2(n-1)}{2n} -$$

$$-\frac{C^2}{2n} Re\left[\sum_{j=1}^{n-1} exp\left(\frac{2\pi m i j}{n}\right)\right] = \frac{C^2(n-1)}{2n} - \frac{C^2}{2n} Re\left[\frac{exp\left(\frac{2\pi m i}{n}\right)\left(exp\left(\frac{2\pi m i (n-1)}{n}\right) - 1\right)}{exp\left(\frac{2\pi m i}{n}\right) - 1}\right] =$$

$$= \frac{C^2(n-1)}{2n} - \frac{C^2}{2n} Re\left[\frac{exp\left(\frac{2\pi m i n}{n}\right) - exp\left(\frac{2\pi m i}{n}\right)}{exp\left(\frac{2\pi m i}{n}\right) - 1}\right] = \frac{C^2(n-1)}{2n} - \frac{C^2}{2n} Re\left[\frac{1 - exp\left(\frac{2\pi m i}{n}\right)}{exp\left(\frac{2\pi m i}{n}\right) - 1}\right] =$$

$$= \frac{C^2(n-1)}{2n} + \frac{C^2}{2n} = \frac{C^2}{2} = 1 \Leftrightarrow C = \sqrt{2} \tag{13}$$

Таким образом, из (11) следует, что для ортонормальности системы (8) необходимо и достаточно положить $C = \sqrt{2}$. Тогда ортонормальный базис Фурье имеет вид:

$$\{\varphi_j^{(m)} = \sqrt{2}\sin\frac{\pi m j h}{b-a} = \sqrt{2}\sin\frac{\pi m j}{n},\ m = 1, \cdots, n-1,\ j = 0, \cdots, n\} \tag{14}$$

Разложение функции $g(x)$ по ортонормальному базису (12) имеет вид:

$$g_j = \sum_{m=1}^{n-1} c_m \varphi_j^{(m)},\ c_m = (g_j, \varphi_j^{(m)}) \text{ и } g(x) \approx \sum_{m=1}^{n-1} c_m \varphi^{(m)}(x) \tag{15}$$

Где $\varphi^{(m)}(x) = \sqrt{2}\sin\frac{\pi m(x-a)}{b-a}$ - значение базисной гармоники из системы (12) в некоторой промежуточной точке x.

4.2. Программная реализация метода

Дискретное преобразование Фурье подразумевает равномерное разбиения отрезка на узлы, поэтому для реализации данного метода необходимо.

А что необходимо? Не ясно.

5. Кубические интерполяционные сплайны

5.1. Описание метода

Пусть имеется функция $p(x)$, определённая на отрезке $[a, b]$, который разбит на подотрезки $[x_{i-1}, x_i]$, $i = 1, \cdots, n$, причём, $x_0 = a$, $x_n = b$. Кубический интерполяционный сплайн на каждом из подотрезков $[x_{i-1}, x_i]$ является полиномом третьей степени

$S_3(x) = a_i x_i^3 + b_i x_i^2 + c_i x + d_i$, который проходит через точки $(x_i, p(x_i))$, $i = 0, \cdots, n$ (два условия на каждом отрезке: $2n$ условий) и имеет непрерывную первую и вторую производные в точках x_i, $i = 1, \cdots, n-1$ (два условия для каждой внутренней точки: $2(n-1) = 2n-2$ условий). Итого имеем $4n-2$ уравнений, а неизвестных коэффициентов $4n$. Поэтому для однозначного решения, необходимо дополнительно задать ещё два уравнения.

Пусть $M_i = S_3''(x_i)$, $i = 0, \cdots, n$. Тогда M_i, в силу условия непрерывности второй производной во внутренних точках разбиения, удовлетворяют системе линейных уравнений $C\overrightarrow{M} = \overrightarrow{d}$, где прямоугольная матрица $C \in \mathbb{R}_{n+1}^{n-1}$ и вектор-столбцы $\overrightarrow{d} \in \mathbb{R}^{n-1}$, $\overrightarrow{M} \in \mathbb{R}^{n+1}$ имеют вид:

$$C = \begin{pmatrix} \frac{h_1}{6} & \frac{h_1+h_2}{3} & \frac{h_2}{6} & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & \frac{h_2}{6} & \frac{h_2+h_3}{3} & \frac{h_3}{6} & \dots & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & \frac{h_{n-2}}{6} & \frac{h_{n-2}+h_{n-1}}{3} & \frac{h_{n-1}}{6} & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & \frac{h_{n-1}}{6} & \frac{h_{n-1}+h_n}{3} & \frac{h_n}{6} \end{pmatrix} \tag{16}$$

$$\overrightarrow{d} = \left(\frac{p_2 - p_1}{h_2} - \frac{p_1 - p_0}{h_1}, \cdots, \frac{p_n - p_{n-1}}{h_n} - \frac{p_{n-1} - p_{n-2}}{h_{n-1}}\right)^T \tag{17}$$

$$\overrightarrow{M} = (M_0, M_1, \cdots, M_{n-1}, M_n)^T \tag{18}$$

Где $h_i = x_i - x_{i-1}$, $i = 1, \cdots, n$.

В качестве доопределяющих уравнений возьмём:

$$S_3''(x_0) = M_0 = S_3''(x_n) = M_n = 0 - \text{естественный сплайн} \tag{19}$$

Таким образом система линейных уравнений $C\overrightarrow{M} = \overrightarrow{d}$ с учётом доопределённых уравнений (17) имеет вид:

$$\begin{pmatrix} \frac{h_1+h_2}{3} & \frac{h_2}{6} & 0 & \dots & 0 & 0 & 0 \\ \frac{h_2}{6} & \frac{h_2+h_3}{3} & \frac{h_3}{6} & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \frac{h_{n-2}}{6} & \frac{h_{n-2}+h_{n-1}}{3} & \frac{h_{n-1}}{6} \\ 0 & 0 & 0 & \dots & 0 & \frac{h_{n-1}}{6} & \frac{h_{n-1}+h_n}{3} \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ \dots \\ M_{n-2} \\ M_{n-1} \end{pmatrix} = \begin{pmatrix} \frac{p_2-p_1}{h_2} - \frac{p_1-p_0}{h_1} \\ \frac{p_3-p_2}{h_3} - \frac{p_2-p_1}{h_2} \\ \dots \\ \frac{p_{n-1}-p_{n-2}}{h_{n-1}} - \frac{p_{n-2}-p_{n-1}}{h_{n-2}} \\ \frac{p_n-p_{n-1}}{h_n} - \frac{p_{n-1}-p_{n-2}}{h_{n-1}} \end{pmatrix} \tag{20}$$

Теперь мы можем однозначно решить систему линейных уравнений (18).

Получив все значения M_i, $i = 0, \cdots, n$ в силу непрерывности и линейности вторых производных во внутренних точках, представим кубический сплайн на каждом подотрезке $[x_{i-1}, x_i]$ в виде:

$$S_3''(x) = M_{i-1}\frac{x_i - x}{h_i} + M_i\frac{x - x_{i-1}}{h_i} \tag{21}$$

Для получения аналитического представления кубического интерполяционного сплайна на подотрезке $[x_{i-1}, x_i]$ необходимо дважды проинтегрировать полученное выражение второй производной (19) с учётом $S_3(x_i) = p_i$, $S_3(x_{i-q}) = p_{i-1}$:

$$\begin{aligned} S_3(x) = M_{i-1}\frac{(x_i - x)^3}{6h_i} + M_i\frac{(x - x_{i-1})^3}{6h_i} + \\ + \left(f_{i-1} - \frac{M_{i-1}h_i^2}{6}\right)\frac{x_i - x}{h_i} + \left(f_i - \frac{M_i h_i^2}{6}\right)\frac{x - x_{i-1}}{h_i} \end{aligned} \tag{22}$$

5.2. Программная реализация метода

В качестве сетки узлов возьмём набор точек разбиения области Ω на n равных частей по горизонтали и m равных частей по вертикали, как представлено на следующем рисунке:

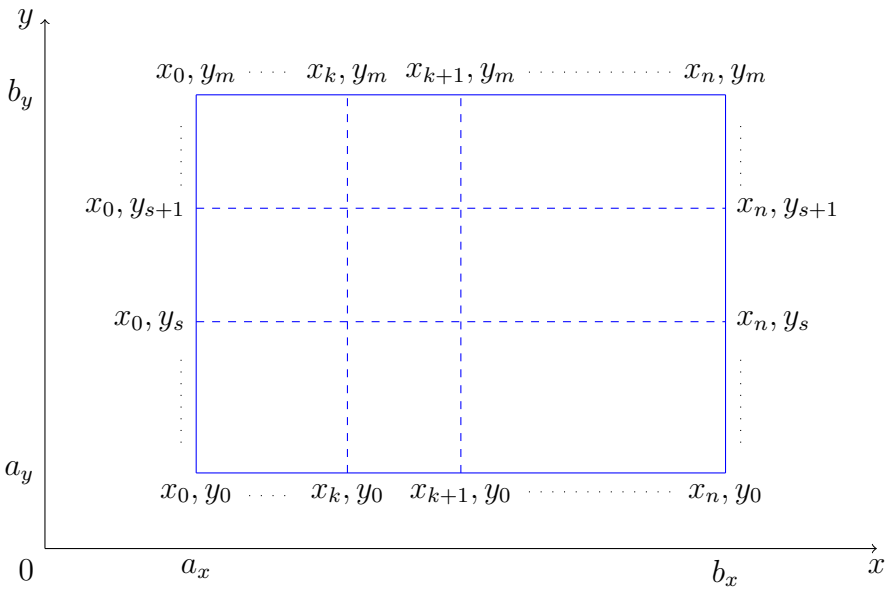

Рис. 6.

Пусть имеются значения некоторой неизвестной функции $f(x, y)$ в узлах (x_k, y_s), где $0 \leqslant k \in \mathbb{Z} \leqslant n$, $0 \leqslant s \in \mathbb{Z} \leqslant m$, $n, m \in \mathbb{N}$ двумерной области $\Omega \subset \mathbb{R}^2$, представляющей из себя прямоугольник, изображенный на Рис. 1, причём часть значения потеряна.

Для того, чтобы восстановить потерянные значения, нам необходимо выбрать направление движения и применить метод кубических интерполяционных сплайнов для восстановления значений функции на отрезке. Удобней всего выбрать два направления: **по оси абсцисс** при фиксированном значении координаты y слева направо в сторону увеличения координаты x и **по оси ординат** при фиксированном значении координаты x снизу вверх в сторону увеличения координаты y. Затем, применив алгоритм по обоим направлениям и получив в каждом неизветсном узле по два значения, возьмём среднее арифметическое этих двух значений в каждом узле и выдадим его в качестве прогнозируемого.

Входными данными для алгоритма являются четыре числа: $a_x, b_x, a_y, b_y \in \mathbb{R}$, описывающие прямоугольную область $\Omega = \{(x, y) \in \mathbb{R}^2 | a_x \leqslant x \leqslant b_x,\ a_y \leqslant y \leqslant b_y\}$ и две прямоугольные $(m + 1) \times (n + 1)$ матрицы: матрица $MatrixTrue \in \mathbb{R}_{n+1}^{m+1}$ значений неизвестной функции $f(x, y)$ в некоторых точках, в которой на $(i,\ j)$-ом месте стоит значение $f(a_x + j\frac{b_x - a_x}{n}, a_y + i\frac{b_y - a_y}{m})$, а также матрица $MatrixFilter \in \mathbb{R}_{n+1}^{m+1}$ индикаторов, состоящая из нулей и единиц, где единица на $(i,\ j)$-ом месте означает, что имеется значение функции $f(x, y)$ в точке $(a_x + j\frac{b_x - a_x}{n}, a_y + i\frac{b_y - a_y}{m})$, а нуль - что это значение потеряно. Предполагается, что в матрице значений функции на $(i,\ j)$-ом месте стоит любое число, если в матрице индикаторов на $(i,\ j)$-ом месте находится нуль. Данное введение необходимо для упрощения представления входных данных, а также для более удобной реализации алгоритма.

Для того, чтобы реализовать алгоритм в горизонтальном направлении, то есть по оси абсцисс, необходимо разбить имеющуюся область Ω на горизонтальные промежутки, и применить метод кубических интерполяционных сплайнов для восстановления значений функции на отрезке по каждому получившемуся в результате разбиения горизонтальному

отрезку $\{(x_k, y_s) \in \mathbb{R}^2 | y_s \in [a_y, b_y],\ s = 0, \cdots, m$ - фиксированнная точка, а $x_k \in [a_x, b_x]$, пробегает все значения при $k = 0, \cdots, n\}$, а затем скомпановать получившиеся восстановленные значения по каждому отрезку в единую матрицу $MatrixRecoveryHor \in \mathbb{R}^{m+1}_{n+1}$ восстановленных значений.

Для того, чтобы реализовать алгоритм в вертикальном направлении, то есть по оси ординат, необходимо разбить имеющуюся область Ω на вертикальные промежутки, и применить метод кубических интерполяционных сплайнов для восстановления значений функции на отрезке по каждому получившемуся в результате разбиения вертикальному отрезку $\{(x_k, y_s) \in \mathbb{R}^2 | x_k \in [a_x, b_x],\ k = 0, \cdots, n$ - фиксированнная точка, а $y_s \in [a_y, b_y]$, пробегает все значения при $s = 0, \cdots, m\}$, а затем также скомпановать получившиеся восстановленные значения по каждому отрезку в единую матрицу $MatrixRecoveryVer \in \mathbb{R}^{m+1}_{n+1}$ восстановленных значений.

Затем, необходимо взять среднее арифметическое значений данных двух матриц в каждом неизвестном узле и сформировать по ним матрицу $MatrixRecovery$. Данная матрица будет выступать в качестве выходных данных программы. Если в некотором узле $(i,\ j)$ значение было известно алгоритму, то в выходной матрице восстановленных значений в этом узле будет находится в точности известное значение, а если в узле $(i,\ j)$ значение подавалось на вход алгоритму, как потерянное, то в этом узле матрицы восстановленных значений будет находится значение, отличающееся от истинного на некоторую погрешность. Так как, мы подали на вход алгоритму матрицу, созданную при помощи заранее известной функции, мы можем сравнить истинные и восстановленные значения и сгенерировать матрицу $MatrixDelta \in \mathbb{R}^{m+1}_{n+1}$, состоящую из разниц истинных и восстановленных значений.

Вычислим матрицы $MatrixDelta$ на пяти функциях, введённых в разделе **Функции**.

Номер	Функция	Ссылка
1	$f_1(x, y) = (x^2 - 1)(y^2 - 1)e^{-xy}$	(**??**)
2	$f_2(x, y) = \sin \pi x + \sin 2\pi x + \sin 50\pi x$	(**??**)
3	$f_3(x, y) = (x^2 - 1)(y^2 - 1)(\text{arctg}(y) \cdot e^x)$	(**??**)
4	$f_4(x, y) = (x^2 - 1)(y^2 - 1)(x^2 - 0.01)(y^2 - 0.01)e^{xy}$	(**??**)
5	$f_5(x, y) = (x^2 - 1)(\cos(x) - \ln(y^2 + 3))$	(**??**)

По данным матрицам, применив алгоритм при различных количествах узлах разбиения, можно будет вычислить скорость (5) сходимости восстановленных значений к истинным, характеризующую точность алгоритма.

Реализовав данный алгоритм в виде программного кода и выполнив его, получаем следующую таблицу значений скорости сходимости (5), где n - количество отрезков разбиения области Ω по оси абсцисс, а m - количество отрезков разбиения области Ω по оси ординат:

n = m	f_1	f_2	f_3	f_4	f_5
100	1.3640	1.0726	1.2765	1.3359	1.4100
200	1.3505	0.0253	1.3412	1.3655	1.4492
300	1.3961	0.0856	1.3833	1.3637	1.4160
400	1.4040	0.1445	1.3709	1.4079	1.4497
500	1.3982	0.1972	1.3883	1.4038	1.4426

Таблица 1. Скорость (5) сходимости восстановленной матрицы к истинной

По результатам Таблицы 1. можно сделать вывод, что при очень маленьких значениях количества отрезков разбиения (100 и 200) точность восстановления не высокая в силу слишком больших длин отрезков разбиения, в то же время, при больших значениях количества отрезков разбиения (700 и больше) мы теряем точность в силу машинной погрешности, так как процессору приходится работать с очень маленькими числами. Чтобы уменьшить зависимость точности от количества точек разбиения по оси абсцисс, возьмём при каждом значении m среднее арифметическое по всем значениям n. Средние значения приведены в последней строке Таблицы 1.

Как итог, можно выбрать **оптимальное значение** количества отрезков разбиения по оси ординат, это **300**, а скорость сходимости восстановленной матрицы к истинной равна **3.04**.

Таким образом, мы построили алгоритм восстановления потерянных значений функции в вертикальном направлении по оси ординат.

Также, вычислим количество операций, необходимых для реализации данного алгоритма.