Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное

образовательное учреждение высшего образования

Санкт-Петербургский университет аэрокосмического приборостроения

КАФЕДРА № 2

БАЛЛЫ ЗА ЗАЩИТУ Отлично 23.12.24

ПРЕПОДАВАТЕЛЬ

доц., канд. техн. наук		В. А. Галанина
должность, уч. степень, звание	*подпись, дата*	*инициалы, фамилия*

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

«АППРОКСИМАЦИЯ ФУНКЦИИ МНК»

по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ Группа	М321		Слотин Н.С.
		подпись, дата	инициалы, фамилия

Санкт – Петербург

2024

Содержание

1. ПОСТАНОВКА ЗАДАЧИ

1.1. Цель работы

Данная курсовая работа является завершает дисциплину «Основы программирования» и требует от студента в процессе ее выполнения решения следующих задач:

1. Решение типовых инженерных задач обработки данных с использованием методов матричной алгебры.
2. Решение систем линейных алгебраических уравнений.

Исходными данными являются функциональная зависимость между X и Y, заданная таблично в n точках, и набор базисных функций. Требуется, используя МНК, найти аппроксимирующую функцию $\varphi\,(x)$ из заданного класса функций и оценить степень ее отклонения от исходной функции.

1.2. Условие задачи

В курсовой работе задан исходный набор базисных аппроксимирующих функций $\varphi 1(x)$, $\varphi 2(x)$, $\varphi 3(x)$ и тем самым определенно число коэффициентов C1, C2, C3 подлежащих определению (m = 3). таким образом, аппроксимирующая функция имеет вид

$$\varphi(x) = C1 * \varphi 1(x) + C2 * \varphi 2(x) + C3 * \varphi 3(x)$$

Исходный набор базисных аппроксимирующих функций используется для откладки разрабатываемых алгоритмов, получения результатов аппроксимации и построения графиков, отображающих исходную и аппроксимирующую функции.

Таблица 1 – исходные данные

№ в-та	n	Значения x_i и $y_i, i = 1, \dots, n$		Базисные функции			Метод решения СЛАУ
				$\phi_1(x)$	$\phi_2(x)$	$\phi_3(x)$	
11.	5	x_i	0,0 0,79 1,57 2,36 3,14	$\sin(x)$	$\cos(x)$	1	Обратной матрицы
		y_i	2,0 6,0 7,0 3,0 0,0				

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

2.1. Методика выбора аппроксимирующей функции

Аппроксимирующую функцию

$$\varphi(x) = \varphi(x, C_1, C_2, \dots, C_m)$$

выбирают из некоторого семейства функций, для которого задан вид функции, но неизвестны и подлежат определению ее параметры C_1, C_2, ..., C_m.

Определение аппроксимирующей функции φ(x) подразделяется на два основных этапа:

- подбор подходящего вида функции φ(x);
- нахождение параметров функции φ(x) в соответствии с критерием МНК.

Подбор вида функции φ(x) представляет собой сложную задачу, решаемую методом проб и последовательных приближений. Исходные данные, представленные в графической форме, сопоставляются с семействами графиков ряда типовых функций, используемых обычно для аппроксимации. После того как выбран вид аппроксимирующей функции φ(x) и, следовательно, определена функциональная зависимость, необходимо найти в соответствии с требованиями МНК значения параметров C_1, C_2, ..., C_m. Как уже указывалось, параметры должны быть определены таким образом, чтобы значение критерия было наименьшим. С учётом критерий аппроксимации J, определяемым выражением представляется функцией искомых параметров

$$J = J(C_1, C_2, \dots, C_m).$$

Последующие действия сводятся к отысканию минимума функции J, зависящий от переменных Ck. Определение значений $Ck = C*_k$, $k = 1,2, \dots, m$, соответствующих этому минимальному J, и является целью решаемой задачей.

Для реализации подхода, заключающегося в том, чтобы использовать известные условия минимум функции нескольких переменных, воспользуемся необходимыми условиями минимума функции нескольких переменных, в соответствии с которыми в

точке минимума должны быть равны нулю частные производные этой функции по всем ее переменам:

$$\frac{\partial J}{\partial C_k} = 0 \ (k = 1,2, \dots , m).$$

Полученные m равенств следует рассматривать как систему уравнений относительно искомых значений C_1, C_2, ..., C_m. При Произвольном виде функциональной зависимости (1.2) уравнения (1.4) оказывается нелинейными относительно величин C_k*, и их решение требует применение приближенных численных методов.

Используемые равенства дают лишь необходимые, но достаточные условия минимума функции. Поэтому требуется уточнить, обеспечивают ли найденные значения C_k* именно минимум функции $J(C_1, C_2, ..., C_m)$. Однако, поскольку величина J неотрицательна (как сумма квадратов) и нижняя граница равна 0, то, существующее решение системы единственно, оно отвечает именно минимуму J. Таким образом, равенство является как необходимым, так и достаточным условием минимума функции.

Уравнения, используемые в МНК, называется нормальными поэтому описываемые способ решения задачи будем называть методом нормальных уравнений.

Структура этих уравнений получается более простой в том случае, когда аппроксимирующая функция $\varphi(x)$ выбирается линейной функции искомых параметров C_k, и выражение имеет вид

$$\varphi(x)=\sum_{k=1}^{m} C_k \varphi_k(x),$$

где C_k - определяемые параметры; $\varphi_1(x)$, $\varphi_2(x)$, … , $\varphi_m(x)$ - Система линейно независимых функций, называемых базисными функциями.

Примеры функции, которые можно использовать в качестве базисных, приведены в Таблице 2.

Таблица 2

Вид функции $\varphi_k(x)$ $(k = 1, \ldots, m)$	**Название функции**
$\varphi_1(x) = x$	Линейная
$\varphi_2(x) = x^2$	Параболическая
$\varphi_3(x) = \frac{1}{x}$	Обратно-пропорциональная
$\varphi_4(x) = \log_a x$	Логарифмическая
$\varphi_5(x) = \sin(x)$ и т.д.	Тригонометрические (и обратные к ним)
...	

В этом случае, подставляя функцию в выражение и выполняя условие минимума проведем дифференцирование, получив систему уравнения относительно искомых C_k.

Подставим выражение функции для m базисных функций

$$\varphi(x) = C_1 * \varphi_1(x) + C_2 * \varphi_2(x) + \cdots + C_m * \varphi_m(x)$$

в Формулу критерия аппроксимации.

Получим

$$J = min \sum_{i=1}^{n} [y_i - C_1\varphi_1(x) - C_2\varphi_2(x) - \ldots - C_m\varphi_m(x)]^2$$

Применим операцию дифференцирования к параметру C_1:

$$2\sum_{i=1}^{n} [y_i - C_1\varphi_1(x) - C_2\varphi_2(x) - \ldots - C_m\varphi_m(x)](-\varphi_1(x_i) = 0$$

и, выполняя необходимые алгебраические преобразования, получим уравнение

$$C_1 \sum_{i=1}^{n} \varphi_1(x)\varphi_1(x) + \ldots + C_m \sum_{i=1}^{n} \varphi_m(x)\varphi_1(x) = \sum_{i=1}^{n} y_i\varphi_1(x)$$

Аналогичные уравнения можно получить, применяя описанные выше действия по отношению к переменным $C_2, \dots, Cm$. Эти уравнения образуют систему нормальных уравнений:

$$a_{11}C_1 + a_{12}C_2 + \dots + a_{1m}C_m = b_1$$
$$a_{21}C_1 + a_{22}C_2 + \dots + a_{2m}C_m = b_2$$
$$\dots\dots\dots\dots\dots\dots\dots\dots$$
$$a_{m1}C_1 + a_{m2}C_2 + \dots + a_{mm}C_m = b_m$$

где коэффициенты akl и величины b_k ($k, l = 1,2, \dots, m$) определяется выражениями

$$a_{kl} = \sum_{i=1}^{n} \varphi_k(x_i)\varphi_l(x_i)$$

$$b_k = \sum_{i=1}^{n} y_i\varphi_k(x_i)$$

Уравнения представляет собой систему линейных алгебраических уравнений. Систему m линейных уравнений вида можно записать посредством матричных обозначений следующем виде:

$A \times C = B$, где

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mm} \end{bmatrix}, C = \begin{bmatrix} C_1 \\ C_2 \\ \dots \\ C_m \end{bmatrix}, B = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_m \end{bmatrix}.$$

Квадратная матрица A называется матрицей системы, вектор C – вектором – столбцом неизвестных системы, а вектор B – вектором - столбцом свободных членов.

В матричном представлении исходная система линейных уравнений примет вид:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mm} \end{bmatrix} \times \begin{bmatrix} C_1 \\ C_2 \\ \dots \\ C_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_m \end{bmatrix}.$$

Решение системы линейных уравнений сводится к отысканию значений элементов вектора – столбца C, называемых корнями системы. Для получения

единственного решения системы, входящие в неё m уравнений должны быть линейно независимыми. Необходимым и достаточным условием этого является неравенство нулю определителя данной системы, то есть detA ≠ 0.

Алгоритмы решения систем линейных уравнений подразделяются на прямые (конечные) и итерационные (бесконечные). Теоретически для получения точного решения итерационные методы требует бесконечного числа арифметических операций. Практически это число приходится брать конечным, поэтому решение имеет некоторую ошибку (конечную точность). Что же касается прямых методов, то они даже при конечном числе операций могут в принципе дать точное решение, если пренебречь ошибками округлений при вычислениях на современных ПК

2.2. Решение системы уравнений методом обратных матриц

Один из методов решения системы линейных алгебраических уравнений, записываемой в матричной форме A × C = B, связан с использованием обратной матрицы A^{-1}. В этом случае решение системы уравнений получается в виде

$$C = A^{-1} \times B,$$

где A^{-1} – матрица, определяемая следующим образом.

Пусть A – квадратная матрица размером m×m с ненулевым определителем det A ≠ 0. Обратная матрица A^{-1} вычисляется как:

$$A^{-1} = \frac{1}{\det(A)} \cdot \mathrm{adj}(A),$$

где adj(A) — присоединённая матрица.

Определитель матрицы A:

$$\det(A) = \sum_{j=1}^{n} a_{1j} \cdot C_{1j},$$

где a_{1j} — элементы первой строки, C_{1j} — алгебраические дополнения.

Алгебраические дополнения:

$$C_{ij} = (-1)^{i+j} \cdot \det\left(M_{ij}\right)$$

где M_{ij} — минор, полученный удалением i-й строки и j-го столбца из матрицы A.

Присоединённая матрица ($adj(A)$):

Присоединённая матрица получается транспонированием матрицы алгебраических дополнений:

$$\operatorname{adj}(A) = \begin{bmatrix} C_{11} & C_{21} & \dots & C_{n1} \\ C_{12} & C_{22} & \dots & C_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ C_{1n} & C_{2n} & \dots & C_{nn} \end{bmatrix}.$$

Обратная матрица:

После нахождения $adj(A)$ и $det(A)$ обратная матрица рассчитывается как:

$$A^{-1} = \frac{1}{\det(A)} \cdot \operatorname{adj}(A)$$

Определитель квадратной матрицы A размерности n×n можно выразить рекуррентно через её миноры. Для n×n-матрицы:

$$det(A) = \sum_{j=1}^{n} (-1)^{1+j} \cdot a_{1j} \cdot \det\left(M_{1j}\right),$$

где M_{1j}— минор, полученный исключением первой строки и jjj-го столбца.

Базовые случаи:

- Для матрицы 1×1:

$$\det(A) = a_{11}$$

- Для матрицы 2×22:

$$\det(A) = a_{11}a_{22} - a_{12}a_{21}$$

Вычислив матрицу R (матрицу A^{-1}), находим матрицу коэффициентов C.

3. РУЧНОЙ СЧЕТ

Исходные данные представлены в табл. 3.

Таблица 3

№ в-та	n	Значения x_i и $y_i, i = 1, \dots, n$		Базисные функции			Метод решения СЛАУ
				$\phi_1(x)$	$\phi_2(x)$	$\phi_3(x)$	
11.	5	x_i	0,0 0,79 1,57 2,36 3,14	$\sin(x)$	$\cos(x)$	1	Обратной матрицы
		y_i	2,0 6,0 7,0 3,0 0,0				

Формулировка критерия аппроксимации и составление системы нормальных уравнений

1. Выражение для аппроксимирующей функции будет иметь следующий вид:

$$\phi(x) = C_1 \cdot \phi_1(x) + C_2\phi_2(x) + C_3\phi_3(x) = C_1 \cdot \sin(x) + C_2 \cdot cos(x) + C_3 \cdot 1$$

2. Запишем выражения для критерия аппроксимации:

$$\mathrm{J} = \sum_{\mathrm{i}=1}^{5} \big(y_i - \phi(x_i)\big)^2 = \sum_{i=1}^{5} [y_i - C_1 \cdot \phi_1(x_i) - C_2\phi_2(x_i) - C_3\phi_3(x_i)]^2 = \\ = \sum_{i=1}^{5} [y_i - C_1 * \sin(x_i) - C_2 * \cos(x_i) - C_3]^2 \quad (1)$$

3. В соответствии с условиями локального минимума функции $J(C_1, C_2, C_3)$ найдём частные производные $\frac{\partial J}{\partial C_1}, \frac{\partial J}{\partial C_2}, \frac{\partial J}{\partial C_3}$ и приравняем их к нулю:

$$\frac{\partial J}{\partial C_1} = \sum_{i=1}^{5} 2(y_i - C_1 - C_2 \cdot x_i - C_3 \cdot e^{-x_i}) \cdot (-\sin(x_i))$$

$$= -2\left(\sum_{i=1}^{5} y_i * \sin(x_i) - C_1 \sum_{i=1}^{5} \sin(x_i)^2 - C_2 \sum_{i=1}^{5} \sin(x_i) * \cos(x_i)\right.$$

$$\left. - C_3 \sum_{i=1}^{5} \sin(x_i)\right) = 0,$$

$$\frac{\partial J}{\partial C_2} = \sum_{i=1}^{5} 2(y_i - C_1 - C_2 \cdot x_i - C_3 \cdot e^{-x_i}) \cdot (-\cos(x_i)) =$$

$$= -2\left(\sum_{i=1}^{5} y_i * \cos(x_i) - C_1 \sum_{i=i}^{5} \sin(x_i) * \cos(x_i) - C_2 \sum_{i=1}^{5} \cos(x_i)^2 - C_3 \sum_{i=1}^{5} \cos(x_i)\right) = 0,$$

$$\frac{\partial J}{\partial C_3} = \sum_{i=1}^{5} 2(y_i - C_1 - C_2 \cdot x_i - C_3 \cdot e^{-x_i}) \cdot (-1) =$$

$$= -2\left(\sum_{i=1}^{5} y_i - C_1 \sum_{i=i}^{5} \sin(x_i) - C_2 \sum_{i=1}^{5} \cos(x_i) - C_3 \sum_{i=1}^{5} 1\right) = 0$$

Таким образом, получим три уравнения:

$$\begin{cases} -2\left(\sum_{i=1}^{5} y_i * \sin(x_i) - C_1 \sum_{i=1}^{5} \sin(x_i)^2 - C_2 \sum_{i=1}^{5} \sin(x_i) * \cos(x_i) - C_3 \sum_{i=1}^{5} \sin(x_i)\right) = 0 \\ -2\left(\sum_{i=1}^{5} y_i * \cos(x_i) - C_1 \sum_{i=i}^{5} \sin(x_i) * \cos(x_i) - C_2 \sum_{i=1}^{5} \cos(x_i)^2 - C_3 \sum_{i=1}^{5} \cos(x_i)\right) = 0 \\ -2\left(\sum_{i=1}^{5} y_i - C_1 \sum_{i=i}^{5} \sin(x_i) - C_2 \sum_{i=1}^{5} \cos(x_i) - C_3 \sum_{i=1}^{5} 1\right) = 0 \end{cases}$$

4. Приведём полученную систему уравнений к нормальному виду, перенеся свободные члены вправо и поделив обе части уравнений на 2.

$$\begin{cases} C_1 \sum_{i=i}^{5} \sin(x_i)^2 + C_2 \sum_{i=1}^{5} \sin(x_i) * \cos(x_i) + C_3 \sum_{i=1}^{5} \sin(x_i) = \sum_{i=1}^{5} y_i \cdot \sin(x_i) \\ C_1 \sum_{i=i}^{5} \sin(x_i) * \cos(x_i) + C_2 \sum_{i=1}^{5} \cos(x_i)^2 + C_3 \sum_{i=1}^{5} \cos(x_i) = \sum_{i=1}^{5} y_i * \cos(x_i) \\ C_1 \sum_{i=i}^{5} \sin(x_i) + C_2 \sum_{i=1}^{5} \cos(x_i) + 5C_3 = \sum_{i=1}^{5} y_i \end{cases} \quad (2)$$

5. Для удобства подставим промежуточные результаты вычислений в соответствии с (2) в табл. 2, запишем систему уравнений (2) в окончательном виде, используя значения из табл. 2:

$$\begin{cases} 2{,}001C_1 - 0{,}001C_2 + 2{,}416C_3 = 13{,}375 \\ -0{,}001C_1 + 2{,}999C_2 - 0{,}005C_3 = 4{,}099 \\ 2{,}416C_1 - 0{,}005C_2 + 5C_3 = 18 \end{cases} \quad (3)$$

Введём обозначения:

$$a_{11} = 2{,}001; a_{12} = -0{,}001; a_{13} = 2{,}416; b_1 = 13{,}375;$$
$$a_{21} = -0{,}001; a_{22} = 2{,}999; a_{23} = -0{,}005; b_2 = 4{,}099;$$
$$a_{31} = 2{,}416; a_{32} = -0{,}005; a_{33} = 5; b_3 = 18$$

Таблица 4

№	x	y	cos(x)	sin(x)	cos(x)^2	sin(x)^2	sin(x)*cos(x)	y*cos(x)	y*sin(x)
1	0,000	2,000	1,000	0,000	1,000	0,000	0,000	2,000	0,000
2	0,790	6,000	0,704	0,710	0,495	0,505	0,500	4,223	4,262
3	1,570	7,000	0,001	1,000	0,000	1,000	0,001	0,006	7,000
4	2,360	3,000	-0,710	0,704	0,504	0,496	-0,500	-2,129	2,113
5	3,140	0,000	-1,000	0,002	1,000	0,000	-0,002	0,000	0,000
∑	7,860	18,000	-0,005	2,416	2,999	2,001	-0,001	4,099	13,375

Тогда систему нормальных уравнений можно записать в матричном виде:

$$\begin{bmatrix} 2{,}001 & -0{,}001 & 2{,}416 \\ -0{,}001 & 2{,}999 & -0{,}005 \\ 2{,}416 & -0{,}005 & 5 \end{bmatrix} \cdot \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} 13{,}375 \\ 4{,}099 \\ 18 \end{bmatrix}$$

6. Полученную систему уравнений решаем методом обратной матрицы.

Находим определитель матрицы А:

$\Delta = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{21}a_{32}a_{13} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{23}a_{32} =$

$= 12{,}49227766$

Определитель матрицы А не равен нулю, соответственно обратная матрица существует. Составляем дополненную матрицу:

$$A_{11} = 1 \cdot \begin{vmatrix} 2{,}999 & -0{,}005 \\ -0{,}005 & 5 \end{vmatrix} = 14{,}996;$$

$$A_{21} = -1 \cdot \begin{vmatrix} -0{,}001 & -0{,}005 \\ 2{,}416 & 5 \end{vmatrix} = -0{,}008;$$

$$A_{31} = 1 \cdot \begin{vmatrix} -0{,}001 & 2{,}999 \\ 2{,}416 & -0{,}005 \end{vmatrix} = -7{,}247;$$

$$A_{12} = -1 \cdot \begin{vmatrix} -0{,}001 & 2{,}416 \\ -0{,}005 & 5 \end{vmatrix} = -0{,}008;$$

$$A_{22} = 1 \cdot \begin{vmatrix} 2{,}001 & 2{,}416 \\ 2{,}416 & 5 \end{vmatrix} = 4{,}165;$$

$$A_{32} = -1 \cdot \begin{vmatrix} 2{,}001 & -0{,}001 \\ 2{,}416 & -0{,}005 \end{vmatrix} = 0{,}008;$$

$$A_{13} = 1 \cdot \begin{vmatrix} -0{,}001 & 2{,}416 \\ 2{,}999 & -0{,}005 \end{vmatrix} = -7{,}247;$$

$$A_{23} = -1 \cdot \begin{vmatrix} 2{,}001 & 2{,}416 \\ -0{,}001 & -0{,}005 \end{vmatrix} = 0{,}008;$$

$$A_{33} = 1 \cdot \begin{vmatrix} 2{,}001 & -0{,}001 \\ -0{,}001 & 2{,}999 \end{vmatrix} = 6{,}001;$$

$$A^* = \begin{bmatrix} 14{,}996 & -0{,}008 & -7{,}247 \\ -0{,}008 & 4{,}165 & 0{,}008 \\ -7{,}247 & 0{,}008 & 6{,}001 \end{bmatrix}$$

Полученную матрицу А* транспонируем, но так как эта матрица симметричная, то $A^* = A^{*T}$:

$$A^* = A^{*T} = \begin{bmatrix} 14{,}996 & -0{,}008 & -7{,}247 \\ -0{,}008 & 4{,}165 & 0{,}008 \\ -7{,}247 & 0{,}008 & 6{,}001 \end{bmatrix}$$

Найдём обратную матрицу A^{-1}, умножив матрицу A^{*T} на $\frac{1}{\Delta}$:

$$A^{-1} = \begin{bmatrix} 14{,}996 & -0{,}008 & -7{,}247 \\ -0{,}008 & 4{,}165 & 0{,}008 \\ -7{,}247 & 0{,}008 & 6{,}001 \end{bmatrix} \cdot \frac{1}{12{,}49} = \begin{bmatrix} 1{,}2004 & -0{,}0007 & -0{,}5801 \\ -0{,}0007 & 0{,}3334 & 0{,}0007 \\ -0{,}5801 & 0{,}0007 & 0{,}4804 \end{bmatrix}$$

Находим вектор коэффициентов, умножив обратную матрицу A^{-1} на матрицу значений В:

$$A^{-1} \cdot B = \begin{bmatrix} 1{,}2004 & -0{,}0007 & -0{,}5801 \\ -0{,}0007 & 0{,}3334 & 0{,}0007 \\ -0{,}5801 & 0{,}0007 & 0{,}4804 \end{bmatrix} \cdot \begin{bmatrix} 13{,}375 \\ 4{,}099 \\ 18 \end{bmatrix} = \begin{bmatrix} 5{,}6109 \\ 1{,}3698 \\ 0{,}8898 \end{bmatrix}$$

7. В результате решения исходной системы линейных уравнений и нахождения значений C_1^*, C_2^* и C_3^* получаем запись искомой аппроксимирующей функции в следующем виде

$$\phi^*(x) = 5{,}6109 \cdot \sin(x) + 1{,}3698 \cdot cos(x) + 0{,}8898 \cdot 1 \qquad (4)$$

8. Оценка погрешности аппроксимации.

Рассчитаем по формуле (4) значения аппроксимирующей функции в заданных точках x_i $(i = 1, \dots, 5)$ и соответствующие отклонения δ_i (табл. 3).

В соответствии с табл. 3 построим графики исходной и аппроксимирующей функций (рис. 1).

Таблица 5

x_i	y_i	$\phi(x_i)$	$\delta_i = \lvert\phi(x_i) - y_i\rvert$
0,000	2,000	2,259618	0,26
0,790	6,000	5,839695	0,16
1,570	7,000	6,501837	0,50
2,360	3,000	3,869927	0,87
3,140	0,000	-0,47108	0,47

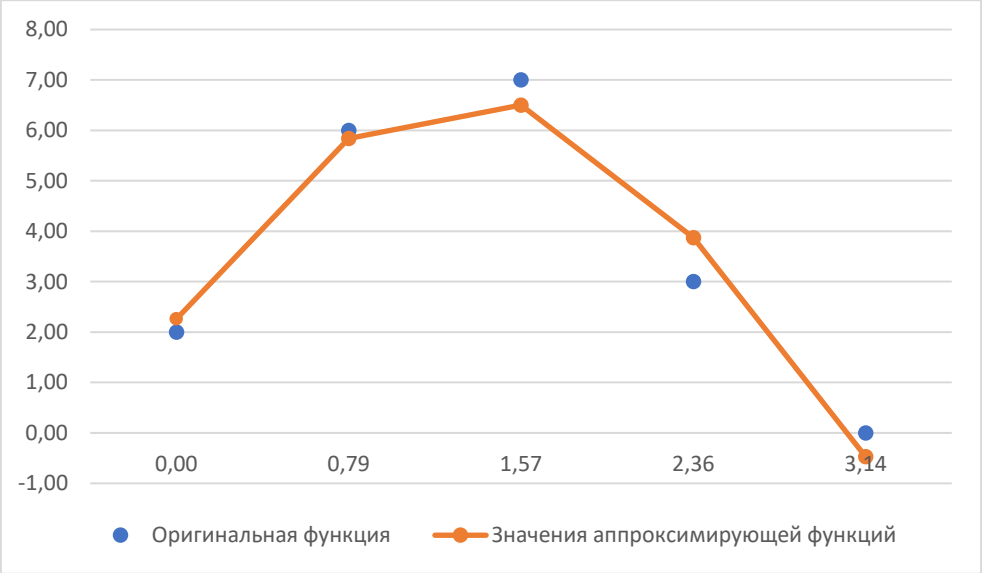


Рис. 1. Графики исходной y_i и аппроксимирующей $\phi(x_i)$ функций.

Вычислим значение критерия аппроксимации, подставив в (1) данные из табл. 3:

$$J_{min} = J(C_1^*, C_2^*, C_3^*) = (0{,}26)^2 + (0{,}16)^2 + (0{,}50)^2 +$$
$$+(0{,}87)^2 + (0{,}47)^2 = 1{,}32$$

Максимальное по модулю отклонение $|\delta_{max}| = 0{,}87$ при $x_i = 2{,}36$.

4. АЛГОРИТМ

4.1. Схема алгоритма

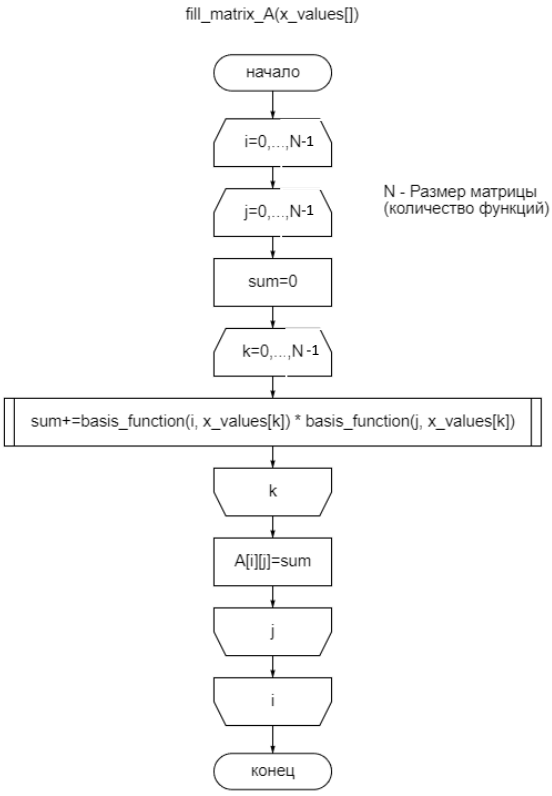


Рис. 2. Алгоритм функции для заполнения матрицы A

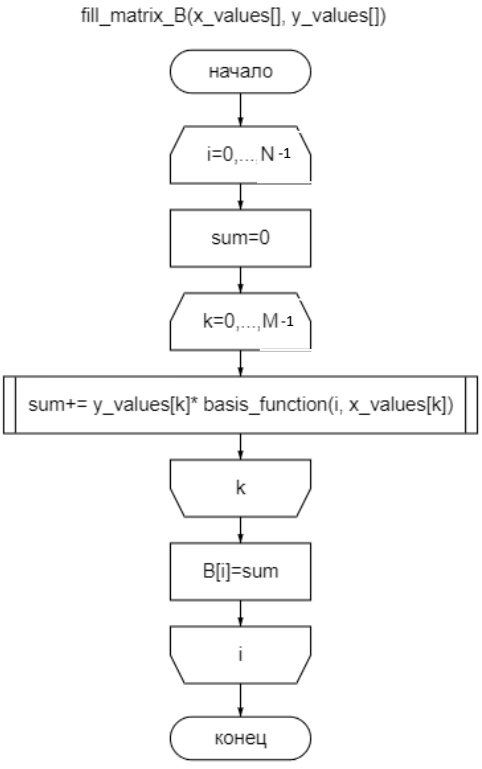


Рис. 3. Алгоритм функции для заполнения матрицы B

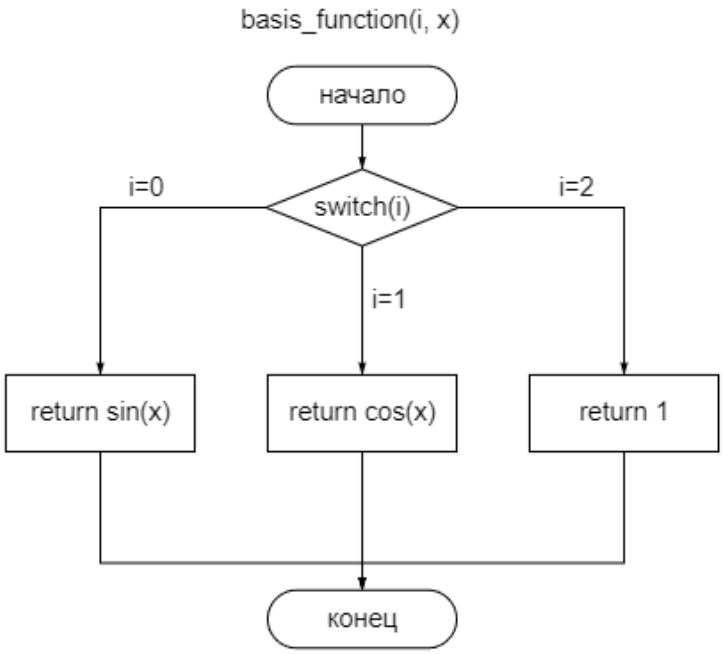


Рис. 4. Алгоритм выбора базисной функции

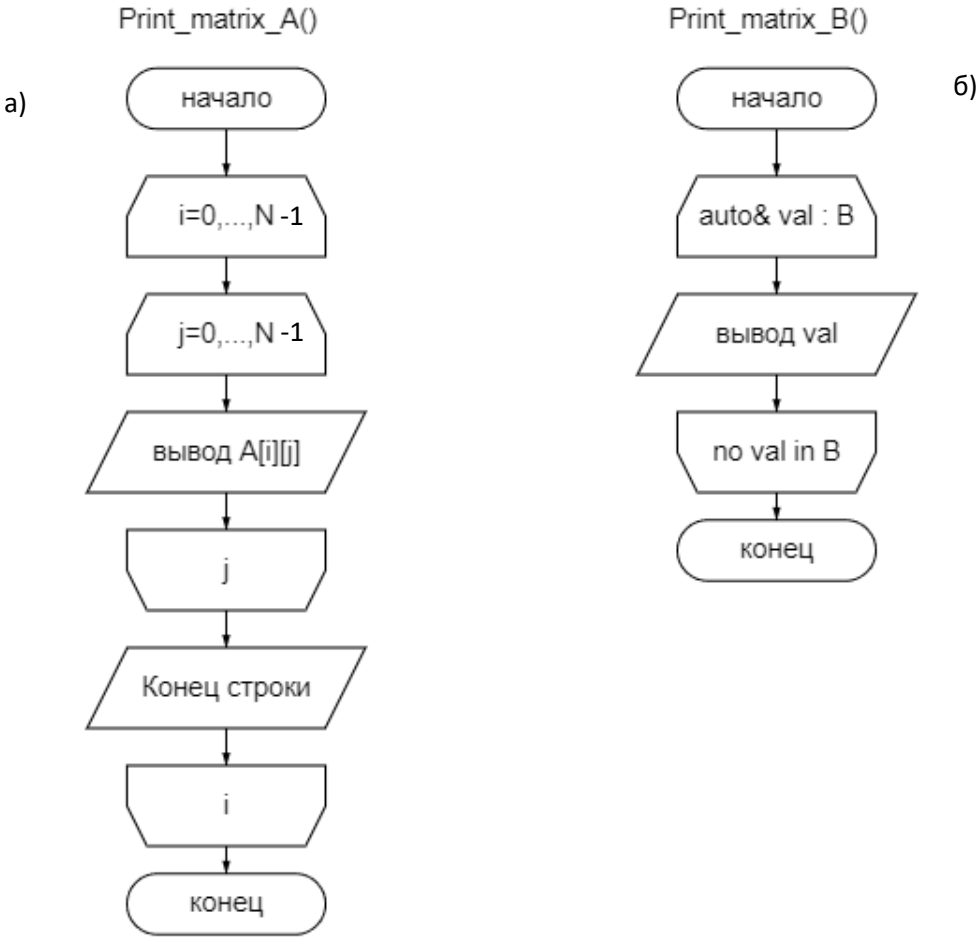


Рис. 5. Алгоритм функции вывода матрицы A (а) и матрицы B (б)

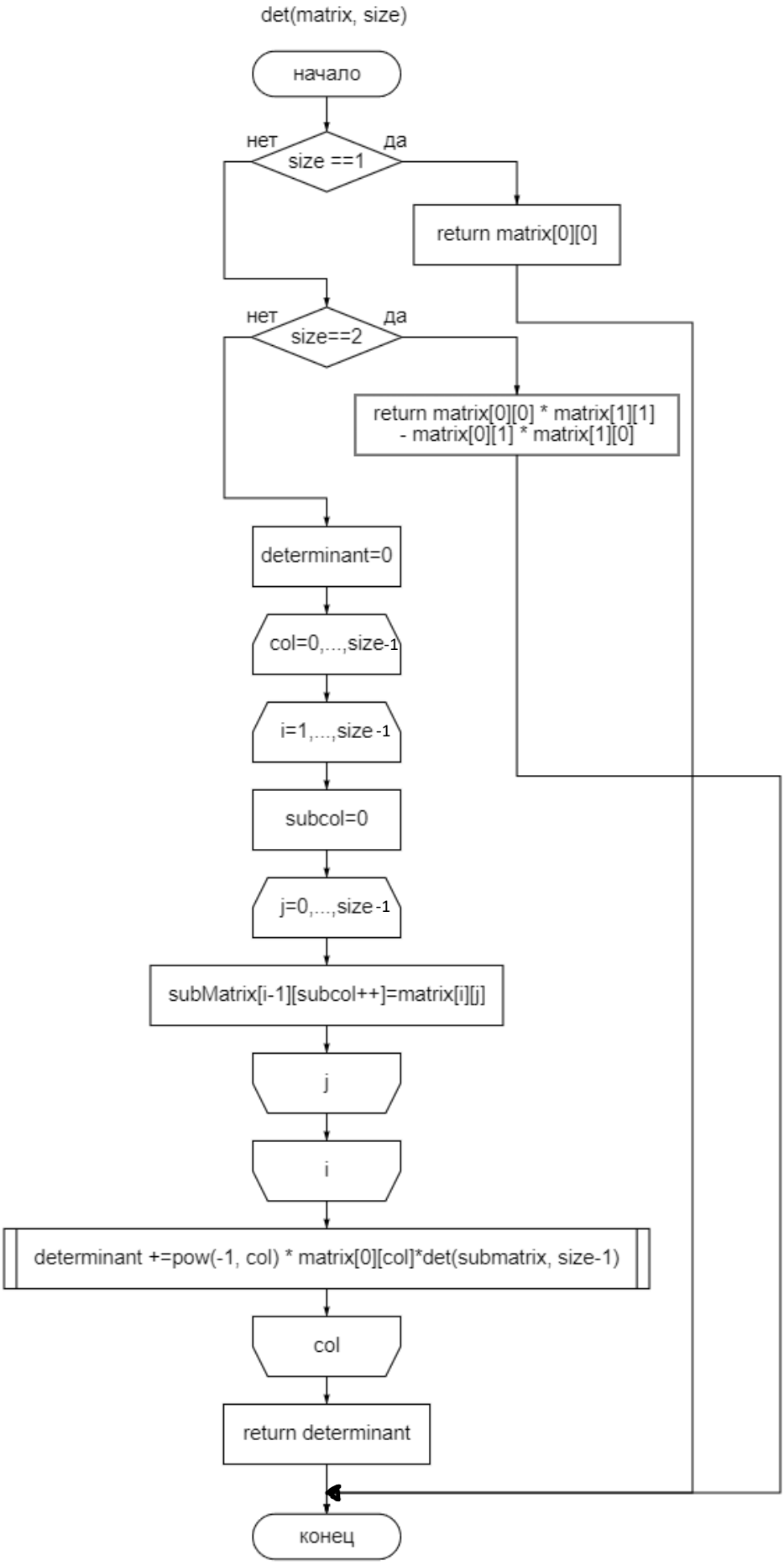


Рис. 6. Алгоритм функции для вычисления детерминанта (рекурсия)

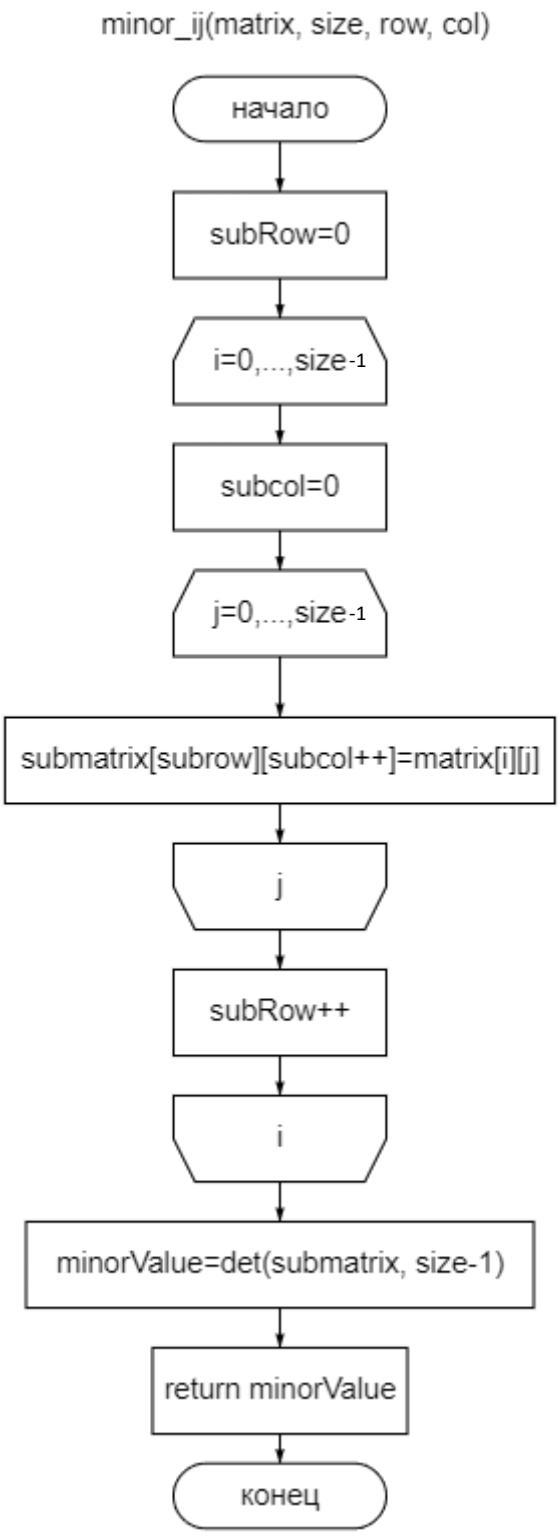


Рис. 7. Алгоритм вычисления арифметических дополнений для указанного элемента

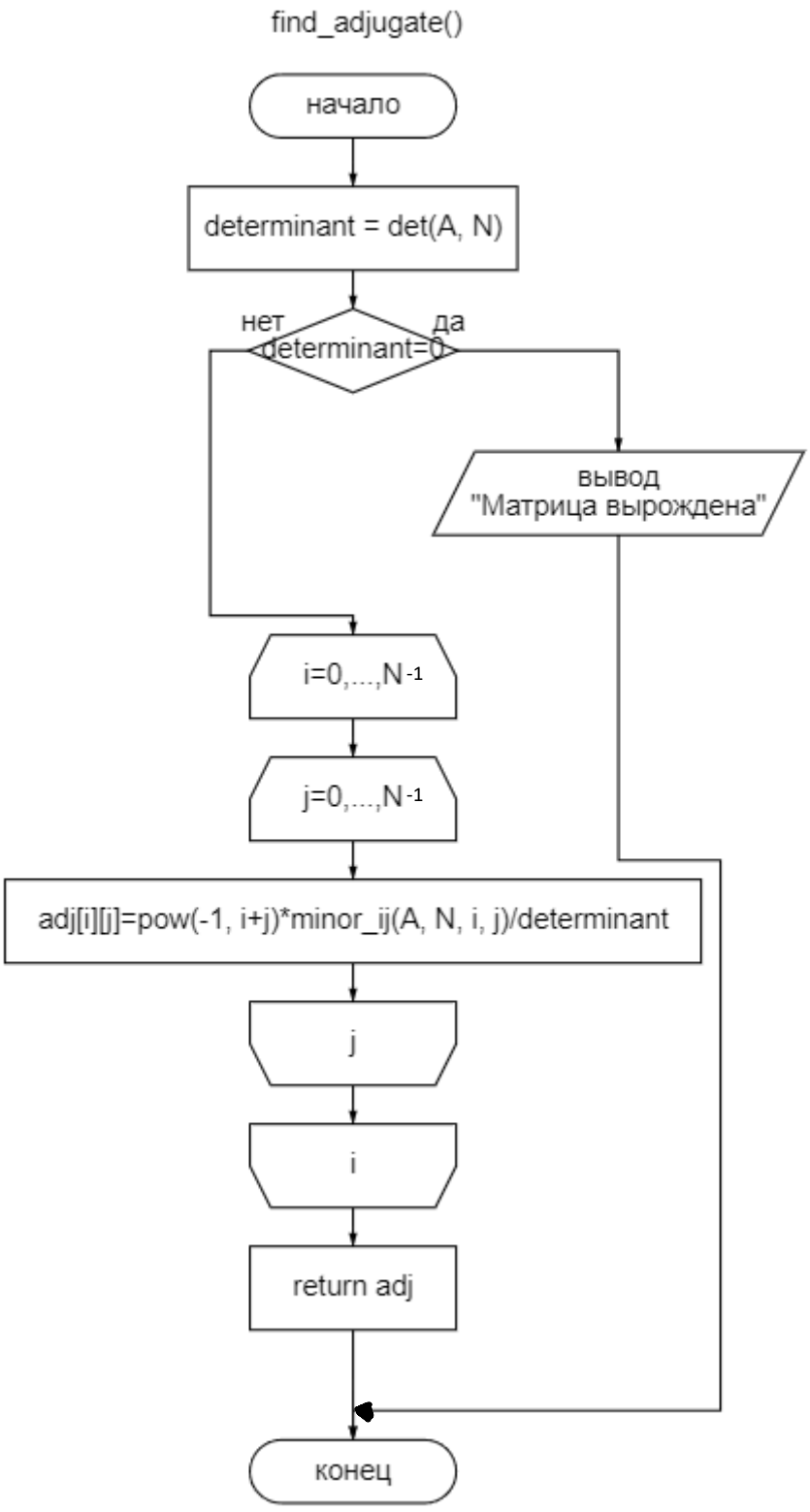


Рис. 8. Алгоритм получения обратной матрицы

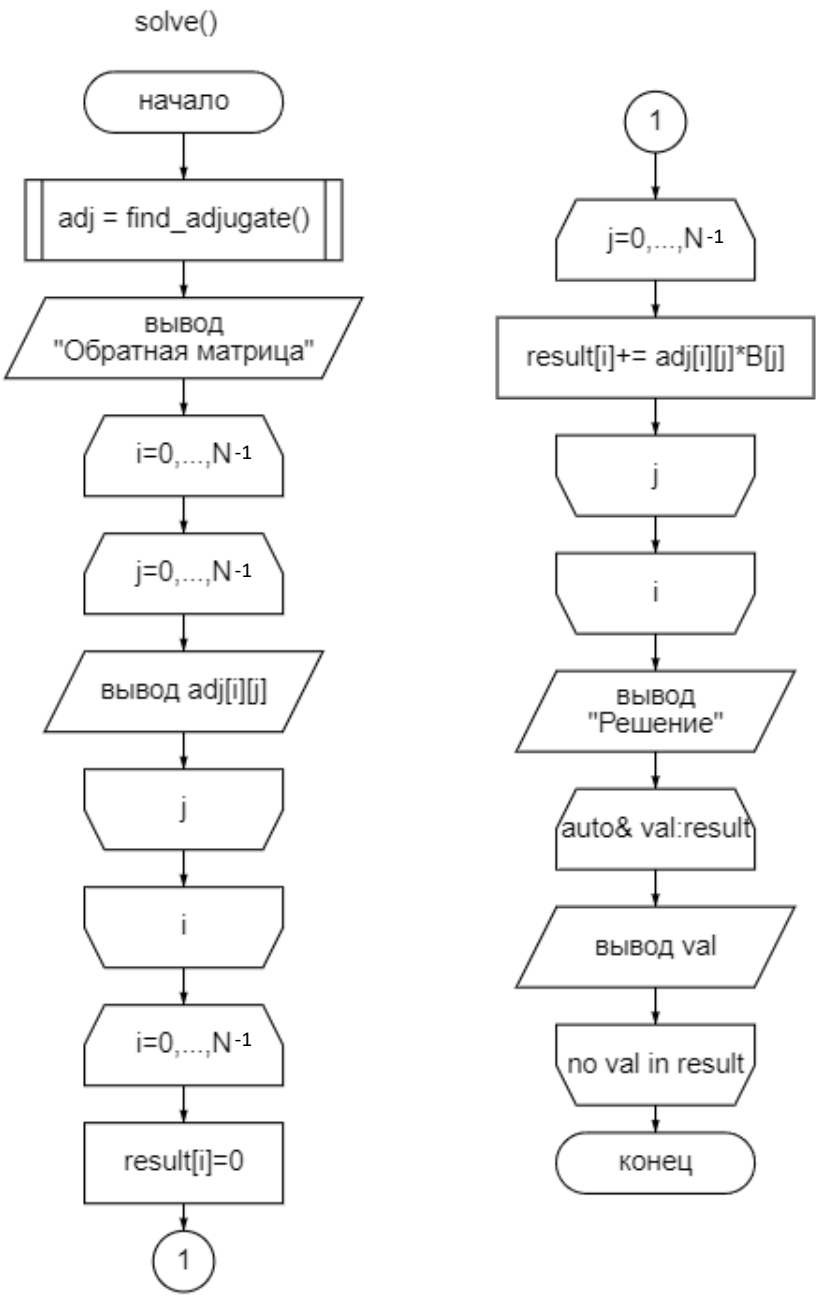


Рис. 9. Алгоритм для подсчета коэффициентов аппроксимации

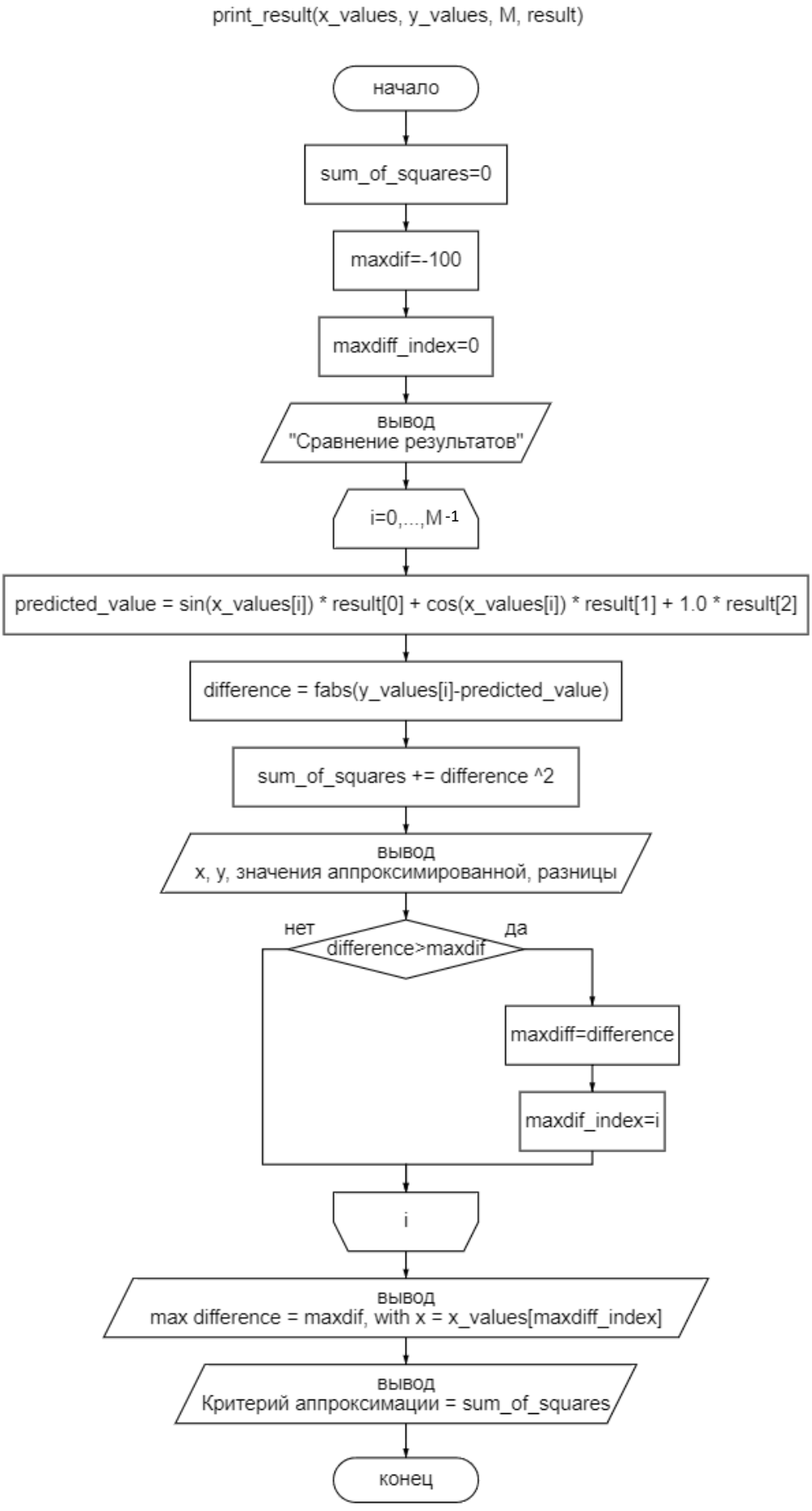


Рис. 10. Алгоритм функции вывода результатов

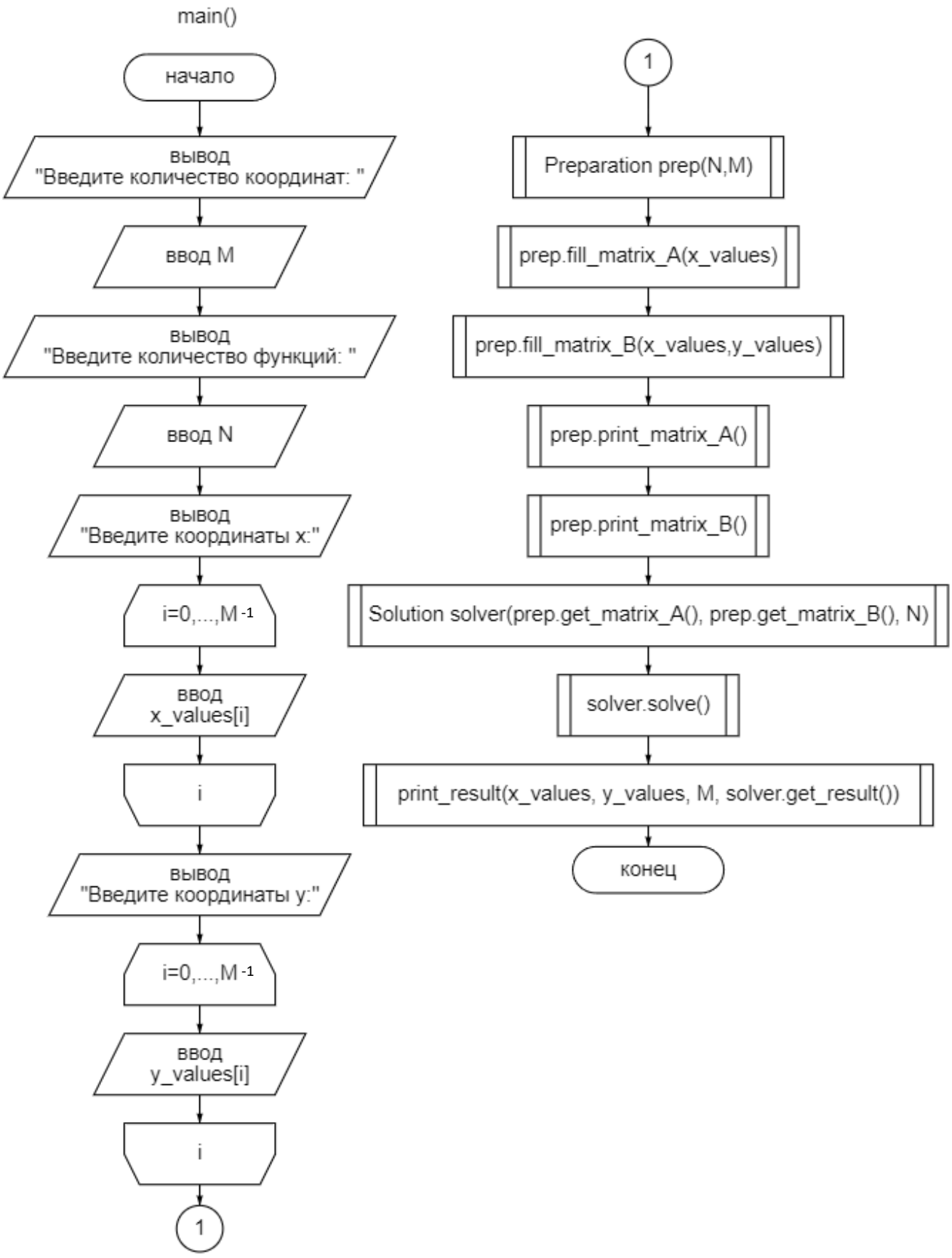


Рис. 11. Алгоритм функции main()

5. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

5.1. Код программы

При выполнении курсовой работы программа была разделена на несколько файлов: main.cpp, source.cpp, source.h

5.1.1. Main.cpp

```
#include <iostream>
#include <Windows.h>
#include "Source_SecondOption.h"

int main() {
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    std::cout.precision(4);
    std::cout << std::fixed;

    int M, N;
    std::cout << "Введите количество координат: ";
    std::cin >> M;
    std::cout << "Введите количество функций: ";
    std::cin >> N;

    double* x_values = new double[M];
    double* y_values = new double[M];
    std::cout << "Введите координаты x:\n";
    for (int i = 0; i < M; ++i) {
        std::cin >> x_values[i];
    }
    std::cout << "Введите координаты y:\n";
    for (int i = 0; i < M; ++i) {
        std::cin >> y_values[i];
    }

    Preparation prep(N, M);
    prep.fill_matrix_A(x_values);
    prep.fill_matrix_B(x_values, y_values);
    prep.print_matrix_A();
    prep.print_matrix_B();

    Solution solver(prep.get_matrix_A(), prep.get_matrix_B(), N);
    solver.solve();
    print_result(x_values, y_values, M, solver.get_result());

    delete[] x_values;
    delete[] y_values;

    return 0;
}
```

5.1.2. Source.cpp

```
#include "Source_SecondOption.h"

void Preparation::fill_matrix_A(double x_values[]) {
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
```

```
            double sum = 0.0;
            for (int k = 0; k < M; ++k) {
                sum += basis_function(i, x_values[k]) * basis_function(j, x_values[k]);
            }
            A[i][j] = sum;
        }
    }
}

void Preparation::fill_matrix_B(double x_values[], double y_values[]) {
    for (int i = 0; i < N; ++i) {
        double sum = 0.0;
        for (int k = 0; k < M; ++k) {
            sum += y_values[k] * basis_function(i, x_values[k]);
        }
        B[i] = sum;
    }
}

double Preparation::basis_function(int i, double x) {
    switch (i) {
    case 0: return sin(x);
    case 1: return cos(x);
    case 2: return 1.0;
    default: exit(1);
    }
}

double** Preparation::get_matrix_A() {
    return A;
}

const std::vector<double>& Preparation::get_matrix_B() const {
    return B;
}

void Preparation::print_matrix_A() {
    std::cout << "\nМатрица A:\n";
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            std::cout << A[i][j] << " ";
        }
        std::cout << std::endl;
    }
}

void Preparation::print_matrix_B() {
    std::cout << "\nМатрица B:\n";
    for (const auto& val : B) {
        std::cout << val << std::endl;
    }
}

double Solution::det(double** matrix, int size) {
    if (size == 1) return matrix[0][0];
    if (size == 2) return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0];

    double determinant = 0;
    double** subMatrix = new double* [size - 1];
    for (int i = 0; i < size - 1; ++i) {
        subMatrix[i] = new double[size - 1];
    }

    for (int col = 0; col < size; ++col) {
        for (int i = 1; i < size; ++i) {
            int subCol = 0;
            for (int j = 0; j < size; ++j) {
```

```
                if (j == col) continue;
                subMatrix[i - 1][subCol++] = matrix[i][j];
            }
        }
        determinant += pow(-1, col) * matrix[0][col] * det(subMatrix, size - 1);
    }

    for (int i = 0; i < size - 1; ++i) {
        delete[] subMatrix[i];
    }
    delete[] subMatrix;

    return determinant;
}

double Solution::minor_ij(double** matrix, int size, int row, int col) {
    double** subMatrix = new double* [size - 1];
    for (int i = 0; i < size - 1; ++i) {
        subMatrix[i] = new double[size - 1];
    }

    int subRow = 0;
    for (int i = 0; i < size; ++i) {
        if (i == row) continue;
        int subCol = 0;
        for (int j = 0; j < size; ++j) {
            if (j == col) continue;
            subMatrix[subRow][subCol++] = matrix[i][j];
        }
        ++subRow;
    }

    double minorValue = det(subMatrix, size - 1);
    for (int i = 0; i < size - 1; ++i) {
        delete[] subMatrix[i];
    }
    delete[] subMatrix;

    return minorValue;
}

double** Solution::find_adjugate() {
    double determinant = det(A, N);
    if (determinant == 0) {
        std::cout << "Матрица вырождена, обратной не существует.\n";
        exit(1);
    }

    double** adj = new double* [N];
    for (int i = 0; i < N; ++i) {
        adj[i] = new double[N];
    }

    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            adj[j][i] = pow(-1, i + j) * minor_ij(A, N, i, j) / determinant;
        }
    }

    return adj;
}

void Solution::solve() {
    double** adj = find_adjugate();
    std::cout << "\nОбратная матрица:\n";
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
```

```
            std::cout << adj[i][j] << " ";
        }
        std::cout << "\n";
    }

    for (int i = 0; i < N; ++i) {
        result[i] = 0;
        for (int j = 0; j < N; ++j) {
            result[i] += adj[i][j] * B[j];
        }
    }

    std::cout << "\nРешение (коэффициенты аппроксимации):\n";
    for (const auto& val : result) {
        std::cout << val << "\n";
    }

    for (int i = 0; i < N; ++i) {
        delete[] adj[i];
    }
    delete[] adj;
}

const std::vector<double> Solution::get_result() const {
    return result;
}

void print_result(double* x_values, double* y_values, int M, std::vector<double> result) {
    double sum_of_squares = 0.0;
    double maxdif = -100;
    int maxdif_index = 0;

    std::cout << "\nСравнение результатов:\n";
    for (int i = 0; i < M; ++i) {
        double predicted_value = sin(x_values[i]) * result[0] + cos(x_values[i]) * result[1] +
1.0 * result[2];
        double difference = y_values[i] - predicted_value;
        sum_of_squares += difference * difference;
        std::cout << "x: " << x_values[i] << " y: " << y_values[i]
            << " значение транспонированной: " << predicted_value << " разница: " <<
difference << "\n";

        if (abs(difference) > maxdif) {
            maxdif = abs(difference);
            maxdif_index = i;
        }
    }

    std::cout << "\nМаксимальное отклонение равно " << maxdif << " и наблюдается при x = " <<
x_values[maxdif_index] << "\n";

    std::cout << "\nКритерий аппроксимации: " << sum_of_squares << "\n";
}
```

5.1.3. Source.h

```
#ifndef SOURCE_H
#define SOURCE_H

#include <vector>
#include <iostream>
#include <cmath>

class Preparation {
private:
```

```
    double** A;
    std::vector<double> B;
    int N, M;

public:
    Preparation(int n, int m) : N(n), M(m), B(n) {
        A = new double* [N];
        for (int i = 0; i < N; ++i) {
            A[i] = new double[N];
        }
    }

    ~Preparation() {
        for (int i = 0; i < N; ++i) {
            delete[] A[i];
        }
        delete[] A;
    }

    void fill_matrix_A(double x_values[]);
    void fill_matrix_B(double x_values[], double y_values[]);
    double** get_matrix_A();
    const std::vector<double>& get_matrix_B() const;
    void print_matrix_A();
    void print_matrix_B();

private:
    double basis_function(int i, double x);
};

class Solution {
private:
    double** A;
    std::vector<double> B;
    std::vector<double> result;
    int N;

    double det(double** matrix, int size);
    double minor_ij(double** matrix, int size, int row, int col);
    double** find_adjugate();

public:
    Solution(double** a, const std::vector<double>& b, int n) : A(a), B(b), N(n), result(n,
0.0) {}
    void solve();
    const std::vector<double> get_result() const;
};

void print_result(double* x_values, double* y_values, int M, std::vector<double> result);

#endif
```

5.2. Результаты работы программы

```
Введите количество координат: 5
Введите количество функций: 3
Введите координаты x:
0.00 0.79 1.57 2.36 3.14
Введите координаты y:
2 6 7 3 0

Матрица A:
2.0008 -0.0008 2.4164
-0.0008 2.9992 -0.0051
2.4164 -0.0051 5.0000

Матрица B:
13.3753
4.0993
18.0000

Обратная матрица:
1.2004 -0.0007 -0.5801
-0.0007 0.3334 0.0007
-0.5801 0.0007 0.4804

Решение (коэффициенты аппроксимации):
5.6109
1.3698
0.8898

Сравнение результатов:
x: 0.0000 y: 2.0000 значение транспонированной: 2.2596 разница: -0.2596
x: 0.7900 y: 6.0000 значение транспонированной: 5.8397 разница: 0.1603
x: 1.5700 y: 7.0000 значение транспонированной: 6.5018 разница: 0.4982
x: 2.3600 y: 3.0000 значение транспонированной: 3.8699 разница: -0.8699
x: 3.1400 y: 0.0000 значение транспонированной: -0.4711 разница: 0.4711

Максимальное отклонение равно 0.8699 и наблюдается при x = 2.3600

Критерий аппроксимации: 1.3200
```

Рис. 12. Вывод консоли после выполнения программы в соответствии с заданием

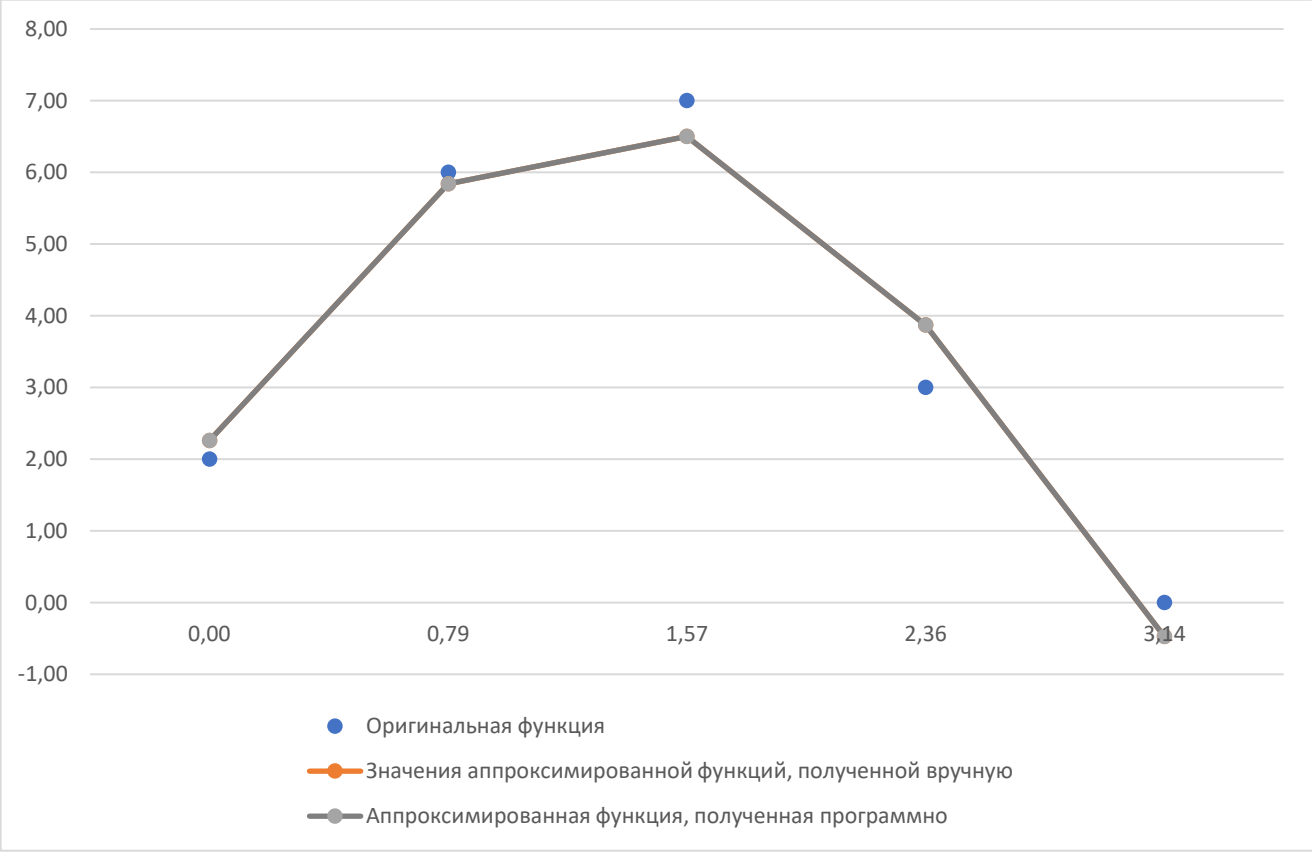
6. ВЫВОДЫ

В результате проделанной работы я освоил основные вычислительные методы прикладной математики, приобрел навыки создания алгоритмов, программ на языке высокого уровня. Был описан критерий аппроксимации, способ его минимизации, составлена система нормальных уравнений, параметры которой вычислили при помощи метода обратных матриц, рассчитаны отклонения аппроксимирующей функции, а также максимальное по модулю отклонение.

Расчеты составлены двумя способами:

- подсчитанные вручную
- подсчитанные на ЭВМ, при помощи составленной программы, написанной в Visual Studio 2022 на языке C++

Исходя из полученных результатов, числовые значения совпали, график ниже подтверждает это совпадение:



7. СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Козенко С.Л., Галанина В.А. Информатика: методические указания к вополнению курсовой работы. – СПб., ГУАП, 2014. – 38 с.
2. Керниган Б.В., Ритчи Д.М. Язык программирования С. – М., Вильямс, 2009.
3. Houston, Robin; Goucher, Adam P.; Johnston, Nathaniel (2023). "A New Formula for the Determinant and Bounds on Its Tensor and Waring Ranks"
4. Скорняков Л. А. Элементы алгебры. — М.: Наука, 1986. — С. 16-23.