МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное
учреждение высшего образования
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
И УПРАВЛЕНИЯ В ТЕХНИЧЕСКИХ СИСТЕМАХ

Кафедра информатики и управления в технических системах

МЕТОДЫ И АЛГОРИТМЫ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ

Методические указания
к выполнению лабораторных работ
для студентов направления подготовки
27.03.04 – Управление в технических системах

Севастополь
СевГУ
2020

УДК 681.5
М54

Р е ц е н з е н т:
В.В. Альчаков - канд. техн. наук, доцент

Составитель: И.Ю. Липко

М54 Методы и алгоритмы вычислительной математики: методические указания к выполнению лабораторных работ для студентов направления подготовки 27.03.04 – «Управление в технических системах» / Сост. И.Ю. Липко ; Министерство науки и высшего образования РФ, Севастопольский государственный университет. – Севастополь: СевГУ, 2020. – 30 с. – Текст : электронный.

Целью лабораторного практикума является освоение студентами методов вычислительной математики и их программной реализации на языке высокого уровня. В практикуме рассматриваются основные алгоритмы из таких разделов как решение систем линейных алгебраических уравнений, систем обыкновенных дифференциальных уравнений, численная оптимизация.

Лабораторный практикум предназначен для студентов по направлению подготовки 27.03.04 – «Управление в технических системах».

УДК 681.5

Рассмотрены и рекомендованы к изданию на заседании кафедры «Информатика и управление в технических системах», протокол № 6 от 15 сентября 2020 г.

Содержание

Введение

Лабораторный практикум включает 3 лабораторные работы, охватывающие темы и вопросы, связанные с методами и алгоритмами вычислительной математики, возникающих при работе специалиста в области управления техническими системами:

1. Погрешности, возникающие при реализации численных алгоритмов;
2. Устойчивость численных алгоритмов;
3. Решение систем алгебраических уравнений;
4. Решение обыкновенных дифференциальных уравнений;
5. Методы численной оптимизации функций.

Каждая лабораторная работа выполняется в течение шести академических часов. В процессе подготовки к выполнению работы студент должен ознакомиться с основными теоретическими положениями, описанием используемых программных инструментов, провести, если требуется, ручные вычисления.

Все лабораторные работы выполняются с обязательной реализацией алгоритмов в виде программного обеспечения, разработанных в среде программирования.

1. Лабораторная работа. Решение систем линейных алгебраических уравнений с помощью LU-разложения

1.1 Цель работы

Изучение способа численного решения систем линейных алгебраических уравнений посредством LU-разложения матриц.

1.2 Краткие теоретические сведения

Системой линейных алгебраических уравнений (СЛАУ) относительно неизвестных $x_1, x_2, \dots, x_m$ называется система уравнений вида

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_m = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{1m}x_m = b_2 \\ \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nm}x_m = b_n \end{cases}, \quad (1)$$

где $a_{ij},\ i = 1,2,\dots,n,\ j = 1,2,\dots,m$ – коэффициенты системы (1), $b_i, i = 1,2,\dots,n$ – правые части.

Решением системы (1) называется упорядоченная совокупность неизвестных $x_1, x_2, \dots, x_m$, удовлетворяющая одновременно каждому из уравнений системы (1).

Системе (1) соответствует матричное представление

$$Ax = b, \quad (2)$$

где

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{1m} \\ \dots & \dots & \dots & \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}.$$

Матрица A называется матрицей коэффициентов, векторы x и b – векторами неизвестных и правых частей системы (2) соответственно.

СЛАУ имеет одно решение, если количество неизвестных равно количеству уравнений, т.е. $n = m$. Далее рассматриваются именно такие СЛАУ.

Решение СЛАУ с нижней треугольной матрицей коэффициентов. Дана система уравнений

$$\begin{cases} l_{11}y_1 = b_1 \\ l_{21}y_1 + l_{22}y_2 = b_2 \\ \cdots \\ l_{n1}y_1 + l_{n2}y_2 + \cdots + l_{nn}y_n = b_n \end{cases}, \quad (3)$$

где l_{ii} – коэффициенты системы, y_i – неизвестные переменные, b_i – правые части.

Соответствующий матричный вид системы (3):

$$Ly = b,$$

где

$$L = \begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & & 0 \\ \vdots & & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}.$$

Алгоритм решения (3):

$$\begin{cases} y_1 = b_1/l_{11}, \\ y_i = (b_{\mathrm{i}} - \sum_{j=1}^{i-1} l_{ij} y_i)/l_{ii} \end{cases}, \mathrm{i} = \overline{2,\mathrm{n}},\ l_{ii} \neq 0 \quad (4)$$

Обратите внимание, что решение начинается с 1-го уравнения и далее в прямой последовательности.

Решение СЛАУ с верхней треугольной матрицей коэффициентов. Дана система уравнений

$$\begin{cases} u_{11}x_1 + u_{12}x_2 + \cdots + u_{1n}x_n = y_1 \\ \cdots \\ u_{\mathrm{n}-1,\mathrm{n}-1}x_1 + u_{\mathrm{n}-1,\mathrm{n}}x_{n-1} = y_{n-1} \\ u_{\mathrm{nn}}x_{\mathrm{n}} = y_{\mathrm{n}} \end{cases}, \quad (5)$$

где u_{ii} – коэффициенты системы, x_i – неизвестные переменные, y_i – правые части.

Соответствующий матричный вид системы (5):

$$Ux = y,$$

где

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1\mathrm{n}} \\ 0 & u_{22} & \cdots & u_{2\mathrm{n}} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix}.$$

Алгоритм решения (5):

$$\begin{cases} x_n = y_n/u_{nn}, \\ x_i = (y_{\mathrm{i}} - \sum_{j=1+1}^{n} u_{ij} x_i)/u_{ii} \end{cases}, \mathrm{i} = \overline{n-1,1},\ l_{ii} \neq 0 \quad (6)$$

Обратите внимание, что решение начинается с n-го уравнения и далее в обратной последовательности.

Решение СЛАУ с матрицей коэффициентов общего вида с помощью LU-разложения. Пусть существует такое мультипликативное представление матрицы коэффициентов *A* системы (2) что

$$A = LU$$

где *L* – треугольная нижняя матрица (с единичной диагональю), *U* – треугольная верхняя матрица.

Тогда задачу (2), посредством подстановки

$$Ax = b,$$
$$LUx = b,$$
$$y \stackrel{\text{def}}{=} Ux,$$

где $\stackrel{\text{def}}{=}$ обозначает «по определению равно» («пусть будет»); можно свести к последовательному решению двух СЛАУ вида (3) и (5)

$$\begin{cases} Ly = b \\ Ux = y \end{cases}. \quad (7)$$

Вышеуказанная последовательность действий и составляет суть решения СЛАУ методом LU-разложения. Решение системы (7) выполняется по алгоритмам (4) и (6). Непосредственное определение неизвестных СЛАУ (2), которое выполняется в процессе решения второго уравнения системы (7), называется *обратным ходом*.

Таким образом, программа для решения СЛАУ должна выполнять следующие шаги:

1. Ввод данных: матрица A, вектор b.
2. Реализация LU-разложения.
3. Решение системы (3) путём реализации алгоритма (4).
4. Решение системы (5) путём реализации алгоритма (6).

5. Вывод на экран:
    a. Матрица A, вектор b.
    b. Матрицы L и U;
    c. Ответ – вектор x

LU-разложение без стратегии выбора ведущего элемента. Рассмотрим один из распространённых алгоритмов получения матриц L и U (LU-разложения) – алгоритм Гаусса. Он основан на идее последовательного исключения неизвестных переменных из уравнений СЛАУ. Схематично этот алгоритм можно представить в виде этапов преобразования матрицы коэффициентов:

$$A = A^{(0)} \Rightarrow A^{(1)} \Rightarrow A^{(2)} \Rightarrow \cdots \Rightarrow A^{(k-1)} \Rightarrow A^{(k)} \Rightarrow \cdots \Rightarrow A^{(n-1)} = U, \quad (8)$$

где $A^{(k)}$ – преобразованная матрица коэффициентов на k-м шаге, $i = \overline{0, n-1}$, n – размер матрицы *A*. В результате получаем треугольные матрицы *U* и *L:*

$$L = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix}, U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix},$$

причём диагональные элементы матрицы *L* равны 1.

Математическое описание *k*-го этапа:

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - l_{ik} a_{kj}^{(k-1)}, \quad (9)$$

$$l_{ik} = a_{ik}^{(k-1)} / a_{kk}^{(k-1)}, \quad (10)$$

где k – номер этапа, $i, j = \overline{k+1, n}$.

Преобразование (8) называется *прямым ходом*.

Некоторые особенности алгоритма LU-разложения и его реализации.

1. При выполнении очередного *к*-го шага прямого хода знаменатель в формуле (10) может оказаться равным нулю. Чтобы этого избежать необходимо выполнить эквивалентные преобразования в матрице A: перестановку столбцов или строк. Этот этап называют выбором ведущего элемента. Выбор ведущего элемента может быть частичным или полным.

В первом случае производится поиск наибольшего по модулю элемента среди элементов *к*-го столбца $a_{pk}^{(k-1)}, p \in \{k, k+1, \dots, n\}$, а затем перестановка в A и b строки k-ой на i-ю, и k-го столбца на j-й. При этом необходимо произвести соответствующую перестановку строк в векторе b (для этого можно использовать массив целых чисел, суть номеров строк).

Во втором случае производится поиск наибольшего по модулю элемента среди элементов блока матрицы $a_{ij}^{(k-1)}, i, j \in \{k, k+1, \dots, n\}$, а затем перестановка k-ой строки на p-ю.

2. При программной реализации необходимо реализовать проверку полученных результатов, а также правильность совершения LU-разложения $A = LU$. Также можно произвести проверку $||A|| - ||LU|| = 0$, что в вычислительном плане будет дешевле.

3. При реализации алгоритма, обычно матрицы *L* и *U* держат в одном двумерном массиве:

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ l_{21} & u_{22} & & u_{2n} \\ \vdots & & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & u_{nn} \end{bmatrix}.$$

1.3 Примеры

1.3.1 LU-разложение

Дана СЛАУ:

$$\begin{cases} 10x_1 - 7x_2 = 8 \\ -3x_1 + 2x_2 + 6x_3 = 24. \\ 5x_1 - x_2 + 5x_3 = 14 \end{cases}$$

Её матричное представление:

$$Ax = b,$$

где

$$A = \begin{bmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{bmatrix}, b = \begin{bmatrix} 8 \\ 24 \\ 14 \end{bmatrix}.$$

Произведём шаги LU-разложения (табл. 1).

Таблица 1.1 – Шаги LU-разложения

Шаг K	Матрица L	Матрица U
0	$L = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix}$	$U = A = \begin{bmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{bmatrix}$
1	$L = \begin{bmatrix} 1 & 0 & 0 \\ -0.3 & l_{22} & 0 \\ 0.5 & l_{32} & l_{33} \end{bmatrix}$	$U = \begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 2.5 & 5 \end{bmatrix}$
2	$L = \begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.5 & -25 & l_{33} \end{bmatrix}$	$U = \begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 0 & 155 \end{bmatrix}$
3	$L = \begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.5 & -25 & 1 \end{bmatrix}$	$U = \begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 0 & 155 \end{bmatrix}$

Проверяем $A = LU$.

Численное решение СЛАУ:

$$x = \begin{bmatrix} -0.5806 \\ -1.2581 \\ 1.1290 \end{bmatrix}.$$

1.3.2 Программа решения уравнения $Ly = b$.

```
// Ly=b.cpp
//
/*
L
2 0 0
-1 1 0
1 2 3

b 2 3 -10

Answer y is
         1
         4
      -19/3
```

```
b 1 4 5
y is
  1/2
  9/2
 -3/2

*/

#include <iostream>

int main()
{
        std::cout << "SLAY_Ly=b.cpp \n";

        const int n = 3;
        double lm[n][n];
        double y[n];
        double b[n];

         // input lm
        for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                        std::cin >> lm[i][j];
                }
                std::cout << std::endl;
        }

        // show lm
        for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                        std::cout << lm[i][j] << " ";
                }
                std::cout << std::endl;
        }

        // input b
        for (int i = 0; i < n; i++) {
                std::cin >> b[i];
        }
        std::cout << std::endl;

        // show b
        for (int i = 0; i < n; i++) {
                std::cout << b[i] << " ";
        }
        std::cout << std::endl;

        // Ly = b;
        //
        // y_1 = b_1/l_11
        // y_i = (b_i- sum{j=(1,i-1)}{l_ij * y_j} ) / l_ii; i=(2,n)
        // i= (2,n);

        y[0] = b[0] / lm[0][0];
        for (int i = 1; i < n; i++) {
                y[i] = b[i];
                for (int j = 0; j < i; j++)
                        y[i] -= lm[i][j] * y[j];
                y[i] /= lm[i][i];
        }

        // show y
        for (int i = 0; i < n; i++) {
```

```
                std::cout << y[i] << " ";
        }
        std::cout << std::endl;
}
```

1.3.3 Программа решения уравнения $Ux = y$.

```
// Ux=y
/*
U
[ 1, 2, 3]
[ 0, 1, 1]
[ 0, 0, 2]

b  -10   3   2

Answer x is
 -17
   2
   1

U
[ 2, 4, 5]
[ 0, 2, 3]
[ 0, 0, 7]

B =  -10   3   2
x is
 -55/7
 15/14
   2/7

Matlab checker:
% eqn1 = x + 2*y + 3*z == -10;
% eqn2 = 0 + y + z == 3;
% eqn3 = 0 + 0 + 2*z == 2;

eqn1 = 2*x + 4*y + 5*z == -10;
eqn2 = 0 + 2*y + 3*z == 3;
eqn3 = 0 + 0 + 7*z == 2;

% Use equationsToMatrix to convert the equations into the form AX = B. The se-
cond input to equationsToMatrix specifies the independent variables in the
equations.
[A,B] = equationsToMatrix([eqn1, eqn2, eqn3], [x, y, z])

% Use linsolve to solve AX = B for the vector of unknowns X.
X = linsolve(A,B)
*/
#include <iostream>

int main()
{
        std::cout << "SLAY_Ux=y.cpp \n";

        const int n = 3;

        double um[n][n];
        double x[n], y[n];

        // input lm
        for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                        std::cin >> um[i][j];
```

```
                }
                std::cout << std::endl;
        }

        // show lm
        for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                        std::cout << um[i][j] << " ";
                }
                std::cout << std::endl;
        }


        // input b
        for (int i = 0; i < n; i++) {
                std::cin >> y[i];
        }
        std::cout << std::endl;

        // show b
        for (int i = 0; i < n; i++) {
                std::cout << y[i] << " ";
        }
        std::cout << std::endl;

        // Ux = y;
        //
        // x_n = y_1/u_nn
        // x_i = (y_i- sum{j=(i+1,n)}{u_ij * x_j} ) / u_ii; i=(n-1,1)
        // i= (2,n);

        x[n-1] = y[n-1] / um[n-1][n-1];
        for (int i = n-2; i > -1; i--) {
                x[i] = y[i];
                for (int j = i+1; j < n; j++)
                        x[i] -= um[i][j] * x[j];
                x[i] /= um[i][i];
        }

        // show y
        for (int i = 0; i < n; i++) {
                std::cout << x[i] << " ";
        }
        std::cout << std::endl;
}
```

1.4 Контрольные вопросы

1. Что такое верхняя/нижняя треугольная матрица?
2. Что такое норма матрицы? Приведите примеры норм матриц.
3. Что такое СЛАУ? Привести примеры.
4. Что такое правая часть СЛАУ?
5. В чём состоит суть метода, идея LU-разложения? Привести пример.
6. Как вычисляются матрицы L и U?
7. Приведите матричное представление СЛАУ
8. Что такое определитель? Расчёт определителя на основе LU-разложения

1.5 Задание для исследования

Выбор исходных данных для выполнения задания определяется вариантом (табл. 1).

1. Произвести ручные вычисления LU-разложения СЛАУ.
2. Разработать программу для численного решения СЛАУ с матрицами коэффициентов треугольного вида (нижней и верхней).
3. Разработать функцию LU-разложения:
    a. Без выбора ведущего элемента.
    б. С выбором ведущего элемента.
4. Разработать программу для решения СЛАУ общего вида.
5. Сверить работу программы с ручными вычислениями.
6. Оценить величину $||A|| - ||LU||$.
7. Разработать функцию для вычисления определителя матрицы.
8. * Оценить влияние изменений в матрице A или векторе правых частей b, путём добавления случайных чисел, на результат решения x.

Таблица 1.2 – Распределение заданий по номеру списка № п/п

	Задание № 1	Задание № 2
1	1	12
2	2	13
3	3	14
4	4	1
5	5	2
6	6	3
7	7	4
8	8	5
9	9	6
10	10	7
11	11	8
12	12	9
13	13	10
14	14	11
15	1	12
16	2	13
17	3	14
18	4	1
19	5	2
20	6	3
21	7	4
22	8	5
23	9	6
24	10	7
25	11	8

Таблица 1.3 – Варианты заданий

№	$A_1 x = b_1$					$A_2 x = b_2$					
	A_1				b_1	A_2					b_2
1	3	2	3	-2	1	3	-1	4.5	2	-0.3	3
	1	5	2	-0.5	-1	1	3	-0.6	1	0	1
	2.1	-3	8	0	0	-0.1	3	4	-3	2	1
	-2	0	-0.7	3.4	3.5	1	0	0.1	-3	0	0

№	$A_1 x = b_1$					$A_2 x = b_2$					
	A_1				b_1	A_2					b_2
						2	-1	0.7	0	4	-2
2	4.2	1.2	-1.7	-2	1.5	5	-3	2	-5	2	4
	1	-4.1	1	2	3	1	1	-3	1	0	-3
	2	1	-4	1	10	1	-3	5	2	3	-5
	1	-2	-1.3	3.5	-2	6	-2	-1	-4	2	1
						6	-7	4	2	9	6
3	3.5	1.2	-1.7	-0.75	3.2	2.4	4	0	-0.8	-0.2	2
	1	-4	1	2	3	2	4	1.2	3.4	-2	1
	2	1	-2.7	1	10	3.6	-2	12	0.4	0.52	-1.5
	1	-2	0	2.2	-2	0.98	-2.3	13.6	9	2.1	0
						-2	-1	0.3	-0.25	-5	2.3
4	3	-3	5	0	3	32	6	4	2.4	8.7	12
	1	3	-0.6	3	0	0.98	4.97	1.23	67	2	6.2
	-0.1	3	8	-3	0	45	6	100.5	1	5	4
	2	0	2	5.5	5	7.2	4.9	-4.5	33.33	4	3
						-5.4	-12	-11.5	4.5	-11	7
5	8	3	2	-3	2	1	1	1	1	1	3
	6	-4	-0.7	0	1	4.5	3.2	6.7	3.7	4.5	5
	-5	-1	8	12	0	-1	1	-1	0	2	2.5
	0	0.2	-0.5	9.8	1.5	4	5	1	8	4	0
						0	2	0	1	3	2
6	6	2	1.7	-0.5	-0.75		3	2	-5	6	3
	4	-4	-0.7	0	1		3	4	9	9	5
	-3.5	-1	8	8.9	-0.5		1	-3	1	-1	-2
	-1	0.2	-0.5	5.8	1.5		3	2	-1	-1	-4
7	9	2	4.5	-7	3		2	9	-12	9	4
	1	5	-3.2	1	3.4		-3	4	-6	7	9
	4	0	-3	-5	0		1	5	-6	2	-2
	0	4	-1	-19	-4.5		1	-2	-3	7	0
8	12	45	-32	-51	32	2	5	-4	-6	1	5
	-34	-21	45	89	-72	1	-1	1	-1	1	1
	45	9.2	6	6.3	0	1	3	5	-3	2	-5
	-6	5	1.3	8	-16	1	2	-9	-3	-1	0
						1	3	6	-2	-4	3
9	3.56	4	12	-0.8	3		2	5	-4	-6	5
	2	4	1.2	3.4	5		1	-1	1	-1	-1
	3.6	-2	13.33	0.4	-3		1	3	5	-3	-5
	0.98	-2.3	16	-9.8	-1		1	2	-9	-3	0
10	3.56	4	3	-0.8	1		5	-3	2	-5	4
	2	4	1.2	3.4	2		1	1	-3	1	-3
	3.6	-2	-13.33	0.4	-3		1	-3	5	2	-5

№	$A_1 x = b_1$					$A_2 x = b_2$					
	A_1				b_1	A_2					b_2
	0.98	-2.3	-16	-9.8	-1		6	-2	-1	-4	1
11	1	2	3	1	6	5	-3	2	-5	2	4
	2	3	-1	0	4	1	1	-3	1	1	-3
	3	1	-4	1	0	1	-3	5	2	3	-5
	1	0	-2	4	6	6	-2	-1	-4	2	1
						6	-7	4	2	9	6
12	5	2	0	-2	6		6	4	6	1	1
	1	9	-1	3	3		-6	2	5	-6	4
	-1	3	7	-2	1		2	6	-3	-2	6
	-2	4	-5	-8	3		1	2	-5	3	7
13	8	3	2	-3	2	1	1	1	1	1	3
	6	-4	-0.7	0	1	4.5	3.2	6.7	3.7	4.5	5
	-5	-1	8	12	0	-1	1	-1	0	2	2.5
	0	0.2	-0.5	9.8	1.5	4	5	1	8	4	0
						0	2	0	1	3	2
14	2	3	-5	1	0	2.4	4	3	-0.8	0.2	1
	1	2	3	2	3	2	4	1.2	3.4	-2	2
	7	1	5	4	1	3.6	-2	-13.33	0.4	0.52	-3
	5	9	4	7	8	0.98	-2.3	-16	-9.8	2.1	-1
						-2	-1	0.3	-0.25	-5	3

1.6 Содержание отчёта

Отчёт выполняется в текстовом редакторе Word, LibreOffice Writer или подобных. Структура отчёта:

1. Титульный лист.
2. Введение и условия задания.
3. Результаты решения.
4. Результаты исследования.
5. Заключение.

Отчёт должен включать:

- условия решаемых задач, номер варианта;
- схему или алгоритм программы;
- текст программ (не скриншоты);
- текстовые комментарии к результатам исследования и краткие выводы.

2. Лабораторная работа. Решение систем обыкновенных дифференциальных уравнений

2.1 Цель работы

Изучение способов численного решения систем обыкновенных дифференциальных уравнений посредством реализации алгоритмов интегрирования и сравнения их качества с точным решением.

2.2 Краткие теоретические сведения

Решение дифференциального уравнения. *Дифференциальным уравнением* называют уравнение, содержащее неизвестную функцию под знаком производной или дифференциала [Эльсгольц, с.10]. Например,

$$\dot{x}(t) = f(x(t), t), \tag{1}$$

где $x(t)$ – неизвестная функция, $\dot{x}(t)$ – производная функции, t – параметр функции (например, время), f – известная функция. Далее для краткости вместо $x(t)$ пишется x.

Обыкновенное дифференциальное уравнение первого порядка (1) можно представить в виде

$$\frac{dx}{dt} = f(x, t).$$

В отличие от алгебраических уравнений, в результате решения которых ищется число (или несколько чисел), при решении дифференциальных уравнений ищется функция. *Решением дифференциального уравнения* называется функция, которая при подстановке в дифференциальное уравнение обращает его в тождество.

Решение (1) содержит произвольную постоянную c [Эльсгольц, с.15]

$$x = \int f(t)\mathrm{dt} + \mathrm{c},$$

которая может быть определена, если известно значение $x(t_0) = x_0$ в момент времени t_0, как

$$x = \int_{t_0}^{t} f(t)\mathrm{dt} + x_0. \tag{2}$$

В случае, если функция x представляет собой *вектор-функцию* (т.е. содержит несколько компонент), то всё вышесказанное сохраняется, но в записи, для обозначения компоненты, используют нижние индексы

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \cdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2, \dots, x_n, t) \\ f_2(x_1, x_2, \dots, x_n, t) \\ \cdots \\ f_3(x_1, x_2, \dots, x_n, t) \end{bmatrix},$$

где $x_1, x_2, \dots, x_n$ – компоненты вектор-функции x, n – количество компонент вектор-функции.

Чтобы не записывать уравнения так громоздко, пишут следующим образом

$$\dot{x}_i = f_i(x_1, x_2, \dots, x_n, t), \ i = 1,2, \dots, n, \tag{3}$$

где i – индекс компоненты.

Запись (3) является формой записи обыкновенных дифференциальных уравнений, называемой нормальной формой (или форма Коши). При такой записи дифференциальное уравнение (1) представляется в виде системы n дифференциальных уравнений первого порядка (3), разрешённых относительно первых производных. К такому виду может быть приведено любое дифференциальное уравнение -ого порядка, допускающее возможность

разрешения относительно старшей производной. Число уравнений в системе равно порядку исходного дифференциального уравнения (обозначаем его n).

Задача Коши. Задачей Коши интегрирования уравнения (3) называется задача определения функций $x_i(t)$, $i = 1,2,\dots,n$, на заданном отрезке изменения независимой переменной t (отрезке интегрирования)

$$t_0 \le t \le t_f, \tag{4}$$

удовлетворяющих заданным начальным условиям

$$x_i(t_0) = x_{i0},\ i = 1,2,\dots,n. \tag{5}$$

и являющихся решениями системы (3). Во многих практических задачах требуется найти только одну из n функций, например $x_1(t)$. Но и в этом случае, поскольку первое уравнение системы (3) зависит от решений других уравнений системы, приходится решать совместно все n уравнений.

Для краткости записи системы (3) убирают индексы и явно указывают, что работают с векторными величинами: следующая запись эквивалентна (3), (4), (5):

$$\dot{x} = f(x,t), \tag{6}$$

$$x(t_0) = x_0,\ t_0 \le t \le t_f. \tag{7}$$

Здесь и далее x, $\dot{x}$, x_0, $f(x,t)$ есть соответственно n-мерные векторы решений $x_i(t)$ и их производных, начальных значений решений $x_i(t_0)$ и правых частей f_i системы (3), (4), (5).

Численное решение задачи Коши. При численном решении задачи Коши вместо поиска вектора решения $x(t)$ уравнения (6) на отрезке интегрирования (7) ограничиваются определением его значений при N значениях независимой переменной $t = t_1, t_2, \dots, t_N = t_f$, называемых *узлами интегрирования*. Расстояние между узлами называют *шагом интегрирования* $h = t_k - t_{k-1}$, $k = 1,2,\dots,N$ (рис. 2.1).

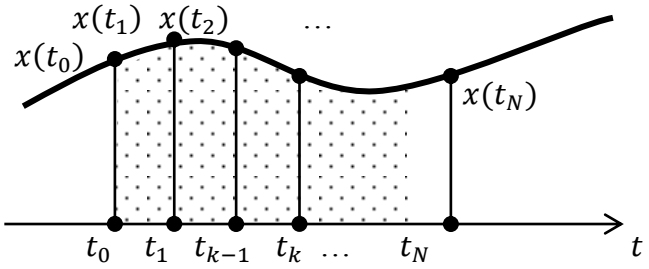

Рисунок 2.1 – Узлы интегрирования и площадь под кривой функции

Тогда значение решения уравнения (6) в $k+1$-м узле можно представить в виде, подобном (2)

$$x(t_{k+1}) = x(t_k) + \int_{t_k}^{t_{k+1}} f(x(t),t)\mathrm{dt}. \tag{8}$$

Переход от одного узла к другому осуществляется посредством интегрирования функции $f(x(t),t)$ на интервале $[t_0, t_1]$, затем на $[t_1, t_2]$, на $[t_2, t_3]$ и т.д.

При численном решении задачи Коши интеграл в (8) заменяется приближённой формулой.

Известно, что геометрическая интерпретация интеграла состоит в том, что он соответствует площади под графиком функции $x(t)$. Если график делится сеткой узлов на части, то очевидно, что интеграл можно заменить суммой площадей этих частей [Волков, гл.2, п.15], например:

$$\int_{t_0}^{t_f} f(x(t),t)\mathrm{dt} \approx \sum_{i=1}^{N} q_i f(x_i), \tag{9}$$

где q_i – некоторые числа, x_i – некоторые числа на отрезке $[t_k, t_{k+1}]$. Это приближённое равенство называется *квадратурной формулой* с *весами* q_i и *узлами в* t_i на всём интервале интегрирования (7).

Квадратурные формулы применяют в (8), аппроксимировав интеграл простыми фигурами (рис. 2.2), например:

– прямоугольник,

– трапеция,

– парабола,

тогда можно использовать формулы (10а) – (10в).

$$\int_{t_k}^{t_{k+1}} f(x(t),t)\mathrm{d}t \approx hf\left(x\left(\frac{t_{k+1}-t_k}{2}\right),\frac{t_{k+1}-t_k}{2}\right), \quad (10а)$$

$$\int_{t_k}^{t_{k+1}} f(x(t),t)\mathrm{d}t \approx \frac{h}{2}[f(x(t_k),t_k)+f(x(t_{k+1}),t_{k+1})], \quad (10б)$$

$$\int_{t_k}^{t_{k+1}} f(x(t),t)\mathrm{d}t \approx \frac{h}{3}\left[f(x(t_k),t_k)+f\left(x\left(\frac{t_{k+1}-t_k}{2}\right),\frac{t_{k+1}-t_k}{2}\right)+f(x(t_{k+1}),t_{k+1})\right]. \quad (10в)$$

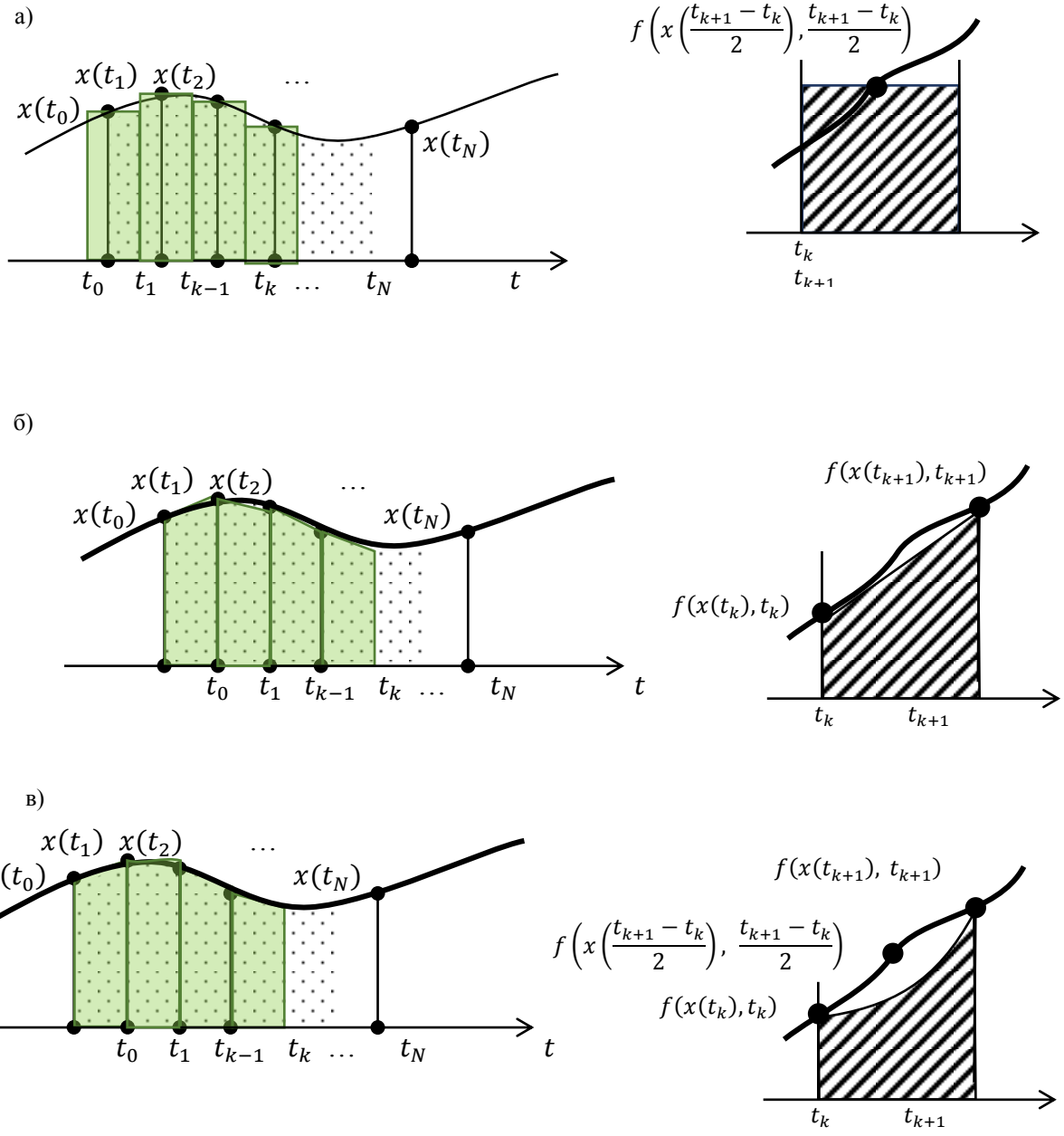

Рисунок 2.2 – К аппроксимации прямоугольником (а), трапецией (б), параболой (в).

На рисунке (рис. 2.2в) показан случай, когда метод параболы визуально выглядит «не качественно», но на практике он показывает результат лучше.

Подставляя (10а) – (10в) в (8) получаем значения решения уравнения (6) в $k+1$-м узле:

$$x(t_{k+1}) = x(t_k) + hf\left(x\left(\frac{t_{k+1}-t_k}{2}\right)\right). \quad (11а)$$

$$x(t_{k+1}) = x(t_k) + \frac{h}{2}[f(x(t_k), t_k) + f(x(t_{k+1}), t_{k+1})] \quad (11б)$$

$$x(t_{k+1}) = x(t_k) + \frac{h}{3}\left[f(x(t_k), t_k) + f\left(x\left(\frac{t_{k+1}-t_k}{2}\right), \frac{t_{k+1}-t_k}{2}\right) + f(x(t_{k+1}), t_{k+1})\right]. \quad (11в)$$

В (11б) и (11в) требуется вычислить значение в узлах t_{k+1} или $\frac{t_{k+1}-t_k}{2}$. Чтобы получить значения в этих узлах выполняют грубый расчёт $x^*(t_{k+1})$, а затем его коррекцию. Например, для (11б):

$$x^*(t_{k+1}) = x(t_k) + hf(x(t_{k+1}), t_{k+1}) \quad (11б*)$$

$$x(t_{k+1}) = x(t_k) + \frac{h}{2}[f(x(t_k), t_k) + f(x^*(t_{k+1}), t_{k+1})] \quad (11б**)$$

Значение $x(t_{k+1})$ есть искомое решение в заданном узле.

Аналогично расчёт выполняется и для (11в).

Формула (11а) составляет суть *метода Эйлера* численного решения уравнения (6).

Методы Рунге-Кутты. Описанные выше методы численного решения дифференциальных уравнений, основанные на квадратурных формулах, часто показывают плохой результат для многомерных случаев ($x(t)$ – векторная величина). Для векторных случаев обычно используют алгоритмы из класса методов Рунге-Кутты. Первые методы данного класса были предложены около 1900 года немецкими математиками К. Рунге и М. В. Куттой.

На практике хорошо зарекомендовал себя *метод Рунге-Кутта 4 порядка точности*:

$$x(t_{k+1}) = x(t_k) + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \quad (12)$$

где

$$k_1 = f(x(t_k), t_k)$$

$$k_2 = f\left(x(t_k) + \frac{k_1}{2}, t_k + \frac{h}{2}\right)$$

$$k_3 = f\left(x(t_k) + \frac{k_2}{2}, t_k + \frac{h}{2}\right)$$

$$k_4 = f(x(t_k) + k_3, t_k + h)$$

Векторы k_i, $i = 1,2,3,4$ – суть значения вектор-функции правых частей $f(x, \mathrm{t})$, вычисленные при разных значениях аргументов. Зависимость каждого вектора от предыдущего обуславливает необходимость последовательного вычисления в порядке нумерации.

Погрешности вычислений. Известно [2-4, 6], что в инженерной практике точное решение задачи Коши возможно лишь в редких случаях. Приходится довольствоваться лишь приближённым решением задачи на основе одного из численных методов.

Например, формы аппроксимации (10) будут влиять на точность интегрирования поскольку появляются не покрываемые промежутки (рис. 2.2).

Соответственно формула (8) при изменениях вида (9) вместо точных значений $x(t_{k+1})$ позволяет определить лишь приближённые значения $\tilde{x}(t_{k+1})$. Возникающая при этом в каждом узле ошибка

$$E(t_k) = x(t_k) - \tilde{x}(t_k) \tag{13}$$

называем о*шибкой алгоритма на шаге*.

Обычно, при $n > 1$, для характеристики точности численного метода применяется не величина алгоритмической ошибки, а её скалярная оценка вида

$$e(t_k) = \|E(t_k)\|. \tag{14}$$

Оценки ошибки на отрезке интегрирования (7) обычно определяют как

$$e = max_{1<k<N}|e(t_k)|. \tag{15}$$

Устойчивость вычислительного алгоритма зависит от вида уравнения (6), численного метода и величины шага интегрирования h. Для любого численного метода и вида уравнения (6) существует такое критическое значение величины шага, превышение которого приводит к потере устойчивости алгоритма.

2.3 Примеры

2.3.1 Нормальная форма ОДУ для простейшего колёсного робота. Простейшее описание робота на колёсах (рис. 2.3) можно получить используя второй закон Ньютона:

$$F = ma,$$
$$F = m\ddot{s},$$

где m – масса робота, a – ускорение, которое приобретает робот при движении, F – сила, с которой двигаем робота, s – пройденное расстояние. Ускорение a является второй производной от расстояния, которое будет пройдено роботом при воздействии на него силы F: $\ddot{s} = \dot{v} = a$, где v – скорость движения. Источником силы F может быть двигатель, который крутит колёса. Силы трения F_{mp} о поверхности не учитываем.

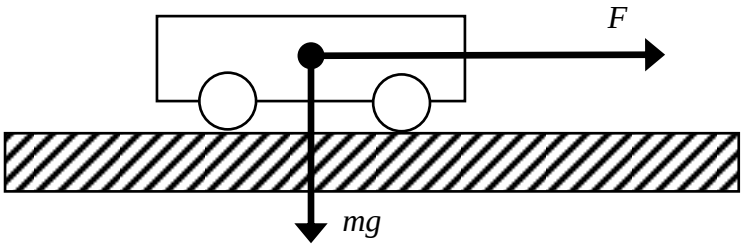

Рисунок 2.3 – Схема сил, действующих на колёсного робота

Приведём это дифференциальное уравнение второго порядка к системе дифференциальных уравнений первого порядка. Для этого введём следующие обозначения:

$$\begin{cases} x_1 \stackrel{\text{def}}{=} s \\ x_2 \stackrel{\text{def}}{=} \dot{s} \end{cases}$$

$$\begin{cases} \dot{x_1} = \dot{s} = \dot{x}_1 \\ \dot{x_2} = \ddot{s} = \frac{1}{m}F \end{cases}$$

где $\stackrel{\text{def}}{=}$ обозначает «по определению равно», «пусть будет». Обратите внимание, что в x индекс равен степени производной. Записываем нормальную форму, выполняя матричные операции и учитывая коэффициенты (равные 0 и 1) при x_1, x_2

$$\begin{cases} \dot{x_1} = 0 \times x_1 + 1 \times x_2 + 0 \times F \\ \dot{x_2} = 0 \times x_1 + 0 \times x_2 + \frac{1}{m} \times F \end{cases},$$
$$\dot{x}(t) = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} \dot{x_1} \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F.$$

Отметим, что такое уравнение является простейшим примером того, как довольно быстро можно получить математическую модель робота на колёсной платформе. Для дальнейшего использования в практике, стоит добавить уравнения для радиусов колёс, обмоток двигателя постоянного тока и учитывать силы трения.

2.3.2 Программа для интегрирования методом Эйлера. Численное решение уравнения $\dot{x} = x \cdot (1 - t)$. Результаты интегрирования, ошибка интегрирования на каждом шаге сохраняются в CSV-файл. В программе комментариями указаны строки, которые можно использовать для применения функций с другими алгоритмами интегрирования.

```
#include <iostream>
#include <fstream>
#include <math.h>

using namespace std;

float solution(float t){
    /* Точное решение
     * for x(0) = 1:
     * solution exp(t-t*t/2)
     * right dot x = x*(1-t)
    */
    return exp(t-t*t/2);
}

float right(float t, float x) {
    // функция правых частей, первой производной
    return x*(1-t);
}

float euler(float x, float t, float h){
    return x + h * right(t, x);
}

float err(float a, float b){
    // смещённая норма как мера ошибки на шаге
    // её удобно рисовать в логарифмической шкале
    return (a-b)*(a-b);
}

int main()
{
    ofstream f;
    f.open("data.csv");

    float t = 0.0;
    float tf = 5.0;
    float h = 0.05;
    float x0 = 1.0;

    float xEu1 = x0;
    //float xRk4 = x0;

    while(t < tf) {
        f << t << "," << solution(t) << ","
          << xEu1 << ","
          //<< xRk4 << ","
          << err(solution(t), xEu1) << ","
```

```
            //<< err(solution(t), xRk4)
            << endl;

        xEu1 = euler(xEu1, t, h);
        // xRk4 = rk4(xRk4, t, h);

        t += h;
    }

    f.close();

    return 0;
}
```

2.4 Контрольные вопросы

1. Что такое ОДУ, запишите примеры?
2. Что такое решение ОДУ?
3. Запишите форму Коши для дифференциального уравнения.
4. Запишите формулу решения ОДУ методом Эйлера.
5. Запишите формулу решения ОДУ методами Рунге-Кутты.
6. Что такое ошибка в узле?
7. Исходная постановка задачи численного решения ОДУ (ОДУ, интеграл, замены интеграла).
8. Суть методов Рунге-Кутта, их отличие (формулы и результаты работы, графики здесь полезны).

2.5 Задачи

Выбор исходных данных для выполнения задания определяется вариантом (табл. 2.1).

1. Провести ручные вычисления: получить форму Коши для уравнения по своему варианту.
2. Реализовать метод Эйлера с модификацией:
   а. Метод трапеций.
   б. Метод парабол.

   Исследовать влияние размера шага на погрешность результата.
3. Реализовать метод Рунге-Кутта 4-го порядка.
4. Исследовать влияние размера шага на погрешность результата для обоих методов.
5. Сравнить точность решения обоих методов с точным решением.

Таблица 2.1 – Варианты постановки задачи Коши

№	Уравнение	t_0	t_f	y_0	y'_0	Точное решение
1	2	3	4	5	6	7
1	$y''-y=-2(\sin t+\cos t)$	0	10	1	1	$\cos t+\sin t$
2	$(1+t^2)y''+y'^2+1=0$	0	10	1	1	$1-t+2\ln(1+t)$
3	$y''+2y'+2y=2e^{-t}\cos t$	0	10	1	0	$e^{-t}(\cos t+\sin t+t\sin t)$
4	$y''+4y=e^{3t}(13t-7)$	0	4	0	-4	$\cos 2t-\sin 2t+e^{3t}(t-1)$
5	$y''+4y'+4y=e^t$	0	10	1	-1	$(8/9+2/3t)e^{-2t}+1/9e^t$
6	$y''-y=\sin t+\cos 2t$	0	10	1,8	-0,5	$e^t+e^{-t}-0,5\sin t-0,2\cos 2t$
7	$y''+4y=\cos 3t$	0	10	0,8	2	$\cos 2t+\sin 2t-0,2\cos 3t$
8	$y''-2y'+y=5te^t$	0	5	1	2	$e^t(1+t+5t^3/6)$
9	$y''+y'-6y=3t^2-t-1$	0	5	-0,9	3,2	$0,1e^{2t}-e^{-3t}-0,5\,t^2$

10	$8y''+2y'-3y = t+5$	0	20	1/9	-7/12	$e^{t/2}+e^{-3t/4}-t/3-17/9$
11	$t^2y''-2y = 0$	1	30	5/6	2/3	$1/2t^2+1/3t^{-1}$
12	$y''-4y'+5y = 3t$	0	5	1,48	3,6	$e^{2t}(\cos t+\sin t)+0,6t+0,48$
13	$y''-3y'+2y = t^2+3t$	0	6	5,1	4,2	$e^t+0,1e^{2t}+t^2/2+3t+4$
14	$y''+ y = 1+e^t$	0	11	2,5	1,5	$\cos t+\sin t+e^t/2+1$
15	$t^2y''+t\,y'-y = 8t^3$	1	30	4	4	t^3+2t+t^{-1}
16	$t^2y''+ty' = 0$	1	30	5	-1	$5-\ln t$
17	$y''-2y'+y = te^t$	0	6	1	2	$e^t(t^3/6+t+1)$
18	$t^2y''+2,5ty'-y = 0$	1	30	2	3,5	$3t^{0,5}- t^{-2}$
19	$4ty''+2y'+y=0$	1	10^2	1,38177	-0,15058	$\sin t^{0,5}+\cos t^{0,5}$
20	$t^2y''-4ty'+6y = 2$	1	11	43/30	2,3	$0,1t^3+t^2+1/3$
21	$y''- y = e^{2t}(t-1)$	0	5	11/9	-11/9	$e^t+e^{-t}+e^{2t}(1/3t-7/9)$
22	$2y''-y'- y = 6e^{t/2}$	0	10	-4	-2,5	$e^t- 6e^{t/2}+e^{-t/2}$
23	$y''+4y'+4y = 2t-3$	0	10	-1/4	-1/2	$(1+t)\,e^{-2t}+t/2- 5/4$
24	$y''+y = t^2-t+2$	0	20	1	0	$\cos t+\sin t+t^2- t$
25	$y''+4y=\sin t+\sin 2t$	0	20	1	-23/12	$(1-t/4)\cos 2t-\sin 2t+1/3\sin t$
26	$t^2y''+y'-2y=t-t^{-2}/3-7$	1	30	26/6	2/3	$t^2/2+t^{-1}/3+3,5$
27	$t^2y''+6y=8t^2+1,2t^3$	1	11	1,1	2,3	$(0,1t+1)\,t^2$
28	$y''+y' =2t-2\sin t-3$	0	20	8	-4	$\cos t+\sin t+t^2-5t+7$
29	$y''+t^{-1}y'=9t+5t^{-1}+t^{-3}$	1	30	13	7	$t^3+5t+t^{-1}+6$
30	$t^2y''+2y=11-2\ln t+4t$	1	30	7	1	$5-\ln t+2t$

2.6 Содержание отчёта

Отчёт выполняется в текстовом редакторе Word, LibreOffice Writer или подобных.
Структура отчёта:
1. Титульный лист.
2. Введение и условия задания.
3. Результаты решения.
4. Результаты исследования. Приводятся в виде графиков и таблиц.
5. Заключение.

Отчёт должен включать:
- условия решаемых задач, номер варианта;
- скриншоты схем;
- текст программ (не скриншоты);
- текстовые комментарии к результатам исследования и краткие выводы.

3. Лабораторная работа. Методы численной оптимизации

3.1 Цель работы

Изучение методов численной минимизации функций посредством их программирования.

3.2 Краткие теоретические сведения

Общие принципы оптимизации. В процессе проектирования обычно ставится задача определения наилучших, в некотором смысле, структуры или значений параметров объектов. Такая задача называется *оптимизационной*. Если оптимизация связана с расчётом оптимальных значений параметров при заданной структуре объекта, то она называется *параметрической оптимизацией*. Задача выбора оптимальной структуры является *структурной оптимизацией*.

Оптимальное решение – это решение, которое по тем или иным критериям предпочтительнее других. *Критерий оптимальности* является необходимым элементом оптимизационной задачи. Например, движение робота из точки А в точку Б по пути 1 может быть предпочтительнее пути 2 по критерию расхода энергии, а путь 2 может быть лучше пути 1 по длине (короче) (рис. 3.1).

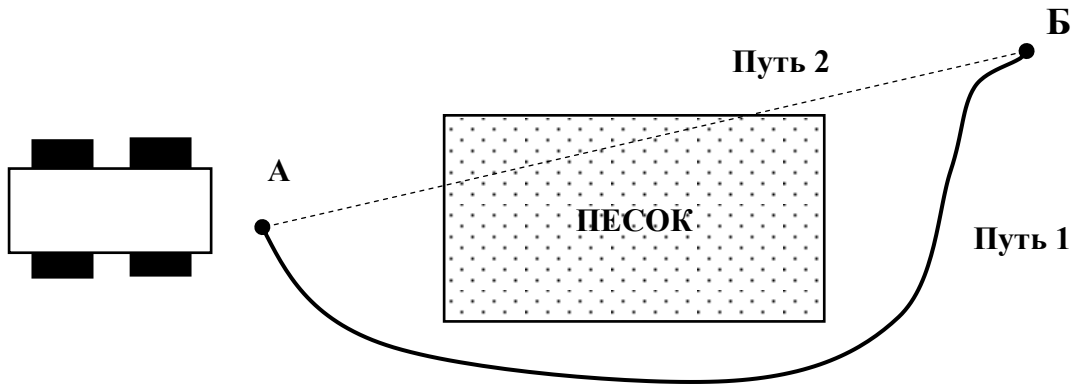

Рисунок 3.1 – Оптимальность путей

Для решения задач оптимизации разработаны численные методы, позволяющие решать задачи с помощью ЭВМ. Данные методы применяются для:

1. Методов идентификации объектов управления.
2. Систем машинного обучения и искусственного интеллекта (нейронные сети в том числе).

Как правило, при решении оптимизационной задачи выполняют следующие этапы:

1. Анализ ситуации и формулировка задачи;
2. Определение параметров решения, подлежащих оптимизации (то есть тех, которые могут быть изменены в ходе решения);
3. Установление допустимой области существования параметров, то есть ограничений, налагаемых на параметры и их сочетания;
4. Выбор и оценка влияния внешних факторов, учитываемых в ходе решения;
5. Выбор критериев оптимальности;
6. Построение целевой функции (и математической модели), которая выдавала бы показатели, соответствующие выбранным критериям;
7. Выбор математического метода оптимизационных расчётов;
8. Проведение расчётов и оценка полученных решений по выбранным критериям;
9. Окончательное принятие решения с учётом неопределённости и риска.

В данной лабораторной работе сосредоточимся на изучении методов оптимизации.

Постановка задачи. *Оптимизация* – это процесс нахождения *экстремума* (минимума или максимума) *целевой функции* в некоторой области конечномерного векторного пространства, ограниченной набором линейных и/или нелинейных равенств и/или неравенств.

Задача оптимизации формулируется следующим образом [Горячев, Осокин]. Среди элементов x, образующих множество X, найти такой элемент x^*, который доставляет минимальное (min) значение $f(x^*)$ заданной функции $f(x)$. Для того, чтобы корректно поставить задачу оптимизации, необходимо задать:

1. *Допустимое множество*, т.е. множество $X = \{x \mid g_i(x) \le 0, i = 1, \dots, m\} \subset R^n$;
2. *Целевую функцию*, т.е. отображение $f: X \to R$;
3. *Критерий поиска* (*max* или *min*).

Тогда решить задачу $f(x) \to min_{x \in X}$ (в другом обозначении $min_{x \in X} f(x)$) означает одно из:

1. Показать, что решений нет $X = \emptyset$.
2. Показать, что целевая функция $f(x)$ не ограничена снизу.
3. Найти решение $x^* \in X: f(x^*) = min_{x \in X} f(x)$.
4. Если x^* не существует, то найти $inf_{x \in X} f(x)$.

Пример постановки задачи оптимизации. Пусть дана функция $f(x, y) = x^2 + y^3 + 1$. Необходимо найти минимум этой функции на плоскости, ограниченной $x \in [a, b],\ y \in [c, d]$. В данном случае:

1. Допустимое множество есть $X = \{x \in [a, b],\ y \in [c, d]\}$.
2. Целевая функция $f(x, y)$. Она зависит от x и y, возвращает одно число в множестве R.
3. Критерий поиска – это минимум (*min*) функции $f(x, y)$.

При решении задачи численным образом, обычно точное решение получить не удаётся. Вместо этого получают приближённое значение $\tilde{x}^*$.

Метод деления отрезка пополам. Этот метод относится к группе т.н. *симметричных* методов. Эти методы гарантировано работают только для *унимодальных функций*. Однако, их возможно применять тогда, когда производная функции либо не существует, либо сложно вычислима, что не позволяет свести задачу к поиску корней производной $f'(x) = 0$.

Функция $f(x)$ называется *унимодальной* на отрезке $[a, b]$, если на этом отрезке существует точка минимума x^* такая, что для любых точек x_1 и x_2 этого отрезка выполняются

$$x^* \le x_1 \le x_2 \Rightarrow f(x^*) \le f(x_1) \le f(x_2),$$
$$x^* \ge x_1 \ge x_2 \Rightarrow f(x^*) \le f(x_1) \le f(x_2).$$

Другими словами, унимодальная функция *монотонна* на обе стороны от точки минимума x^*. Аналогично определяется унимодальная функция и для задачи на максимум. Унимодальные функции могут быть непрерывными, разрывными, дискретными. На рис. 3.2 показан пример унимодальных функций: *а* и *б* имеют один минимум; *в* имеет 2 минимума и является не унимодальной на отрезке АС, но унимодальной на отрезках АВ и ВС; в случае *г* функция не унимодальная.

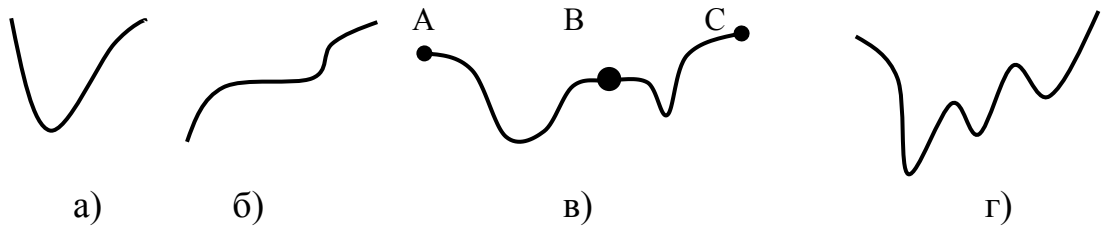

Рисунок 3.2 – Унимодальные (а, б, в) и не унимодальная функции (г)

Суть методов заключаются в построении последовательности отрезков $[a_n, b_n]$ (где n – количество шагов), стягивающихся к точке $x^* = argmin\{f(x): x \in [a, b]\}$. На каждом шаге поиска выбирается две точки отрезка x_1 и x_2, симметрично расположенных относитель-

но центра этого отрезка. Дальнейшие действия определяются свойством унимодальной функции.

Пусть функция $f(x)$ унимодальна на отрезке $[a, b]$, а ее минимум достигается в точке x^* Для любых точек x_1 и x_2 этого отрезка и таких, что $a < x_1 < x_2 < b$ верно следующее:

если $f(x_1) > f(x_2)$, то точка минимума $x^* \in [x_1, b)$,
если $f(x_1) < f(x_2)$, то точка минимума $x^* \in (a, x_2]$.

Исходя из определения методов, видно, что всякий симметричный метод полностью определяется заданием отрезка $[a, b]$ и правилом выбора первой точки. Тогда другая ка x_2 находится по правилу общему для всех симметричных методов: $x_2 = a + b - x_1$.

Соответственно, методы различаются способом выбора симметричных точек x_1 и x_2.

Алгоритм метода деления отрезка пополам. Задана функция $f(x)$ и отрезок $[a, b]$. Параметры δ, ϵ – достаточно малые положительные константы. Результат: найденная точка оптимума $\tilde{x}^*$.

1. Повторять:
   a. $x_1 = \frac{a+b-\delta}{2}$, $x_2 = \frac{a+b+\delta}{2}$;
   b. Если $f(x_1) > f(x_2)$, то $a = x_1$;
   c. Если $f(x_1) < f(x_2)$, то $b = x_2$;
2. Пока $\frac{b-a}{2} \geq \epsilon$;
3. $\tilde{x}^* = \frac{a+b}{2}$.

В процессе работы алгоритма имеем последовательность отрезков $[a_i, b_i]$, где i – номер шага, которые уменьшаются по длине (рис. 3.3). Получившийся на последнем шаге отрезок $[a_n, b_n]$ имеет длину меньше 2ϵ. Считаем, что оптимальное значение лежит посередине этого отрезка.

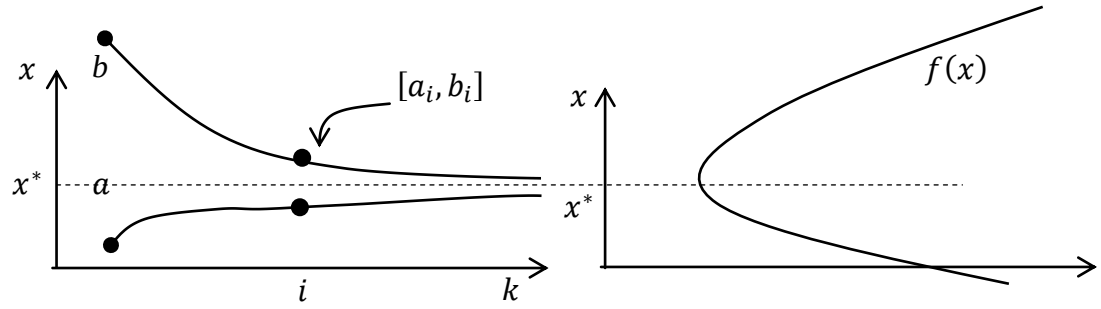

Рисунок 3.3 – Значения границ отрезка на каждом шаге и целевая функция

Метод градиентного спуска с постоянным шагом. Данный метод относится к группе т.н. градиентных методов [Поляк, с.29]. Исходя из названия, основную роль в данных методах играет *градиент* (или производная для одномерной функции), поскольку его значение указывает на направление к экстремуму (рис. 3.4).

Рассматривается задача поиска минимума функции $f(x)$: $R^n \to R$, такой что, что можно вычислить ее *градиент* ∇f:

$$f(x) \to \min_{x \in R^n} \quad .$$

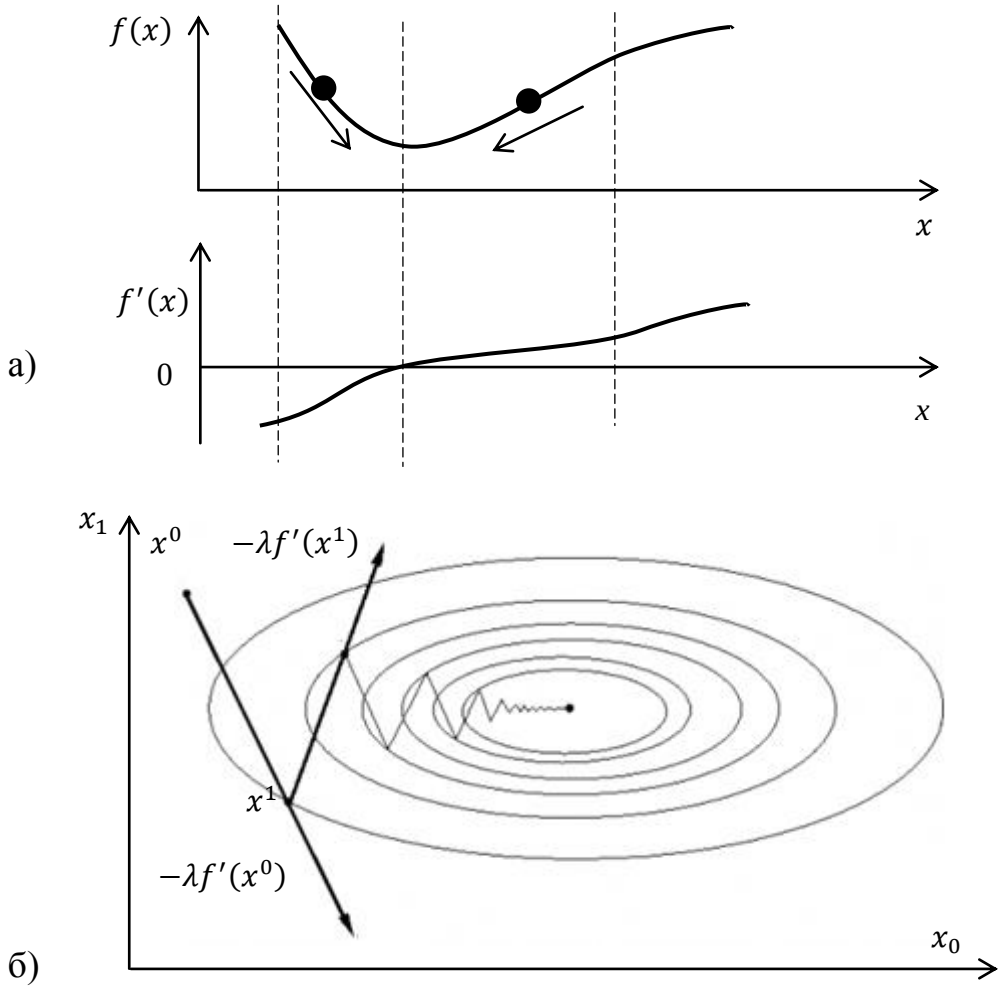

Рисунок 3.4 – Геометрическая интерпретация метода градиентного спуска для одномерного (а) и многомерного (б) случаев

Основная идея метода заключается в том, чтобы осуществлять оптимизацию в направлении наискорейшего спуска, которое задаётся антиградиентом (минус градиентом) $-\nabla f$, на шаг λ:

$$x^{[k+1]} = x^{[k]} - \lambda^{[k]} \nabla f(x^{[k]}),$$

где λ выбирается:

1. Постоянной (в этом случае метод может расходиться).
2. Дробным шагом, т.е. длина шага в процессе спуска делится на некое число.
3. Наискорейшим спуском: $\lambda^{[k]} = \arg\min_{\lambda} f(x^{[k]} - \lambda \nabla f(x^{[k]}))$.

В данной работе рассматривается первый способ (но проницательный студент может попробовать другие). Результат выбора λ напрямую влияет на качество работы алгоритма. Например, на рис. 3.5 он слишком маленький, что приводит к тому, что движение к минимуму происходит очень медленно.

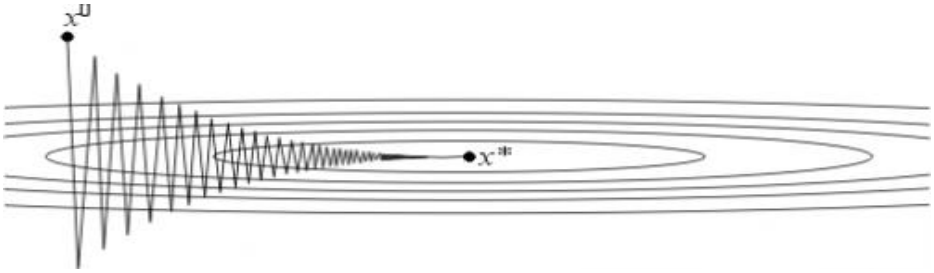

Рисунок 3.5 – Ситуация, когда метод градиентного спуска сходится плохо

Алгоритм градиентного спуска. Дана функция $f(x): R^n \to R$. Результат**:** найденная точка оптимума $\tilde{x}^*$.

1. Повторять

$$x^{[k+1]} = x^{[k]} - \lambda^{[k]} \nabla f(x^{[k]}),$$

где $\lambda^{[k]}$ выбирается одним из описанных выше способов

2. Если выполнен критерий останова, то возвращаем текущее значение $x^{[k+1]}$. Это значение является приближением $\tilde{x}^*$ к точке минимума x^*.

Критерии остановки процесса приближенного нахождения минимума могут быть основаны на различных соображениях, например:

1. $\|x^{[k+1]} - x^{[k]}\| \leq \epsilon$;
2. $\|f(x^{[k+1]}) - f(x^{[k]})\| \leq \epsilon$.

Здесь $x^{[k]} \in R^n$ – значение, полученное после k-го шага оптимизации, ϵ – наперед заданное положительное число, $\|x\|$ – норма вектора x.

Всегда важно выбирать такие параметры алгоритмов, которые позволят дёшево (вычислительные затраты) и быстро (количество шагов) получить хороший (как можно более точный) результат.

3.3 Пример исследования

Сравним градиентные методы с разным подход к выбору шага λ для одной и той же задачи: достигнутые точности и количество шагов, выполняемых алгоритмами.

Дана функция $f(x_1, x_2) = 10 * x_1^2 + x_2^2$. Начальное приближение – точка $(x_1, x_2) = (10,10)$. Найти её минимум. Использован критерий останова: $\|f(x^{[k+1]}) - f(x^{[k]})\| \leq 10^{-5}$

Результаты эксперимента отражены в табл. 3.1 и 3.2.

Таблица 3.1 – Градиентный спуск с постоянным шагом

Значение шага λ	**Достигнутая точность**	**Количество итераций**
0.1	метод расходится	
0.01	$2 \cdot 10^{-4}$	320
0.001	$2 \cdot 10^{-3}$	2648
0.0001	$1 \cdot 10^{-2}$	20734

Таблица 3.2 – Градиентный спуск с дроблением шага

Значение параметра ϵ	**Значение параметра δ**	**Значение параметра λ**	**Достигнутая точность**	**Количество итераций**
0.95	0.95	1	$5 \cdot 10^{-4}$	629
0.1	0.95	1	$1 \cdot 10^{-5}$	41
0.1	0.1	1	$2 \cdot 10^{-4}$	320
0.1	0.95	0.01	$2 \cdot 10^{-4}$	320

Для градиентного спуска с дроблением шага (δ – параметр алгоритма) видно, что есть хороший выбор параметров: $\epsilon = 0.1, \delta = 0.95, \lambda^{[0]} = 1$, хотя метод не сильно чувствителен к выбору параметров.

Метод наискорейшего спуска получил точность $6 \cdot 10^{-8}$ за 9 итераций. Однако для него необходимо использовать решения одномерных задач оптимизации (например, метод золотого сечения).

3.4 Контрольные вопросы

1. Что такое экстремум?

2. Как найти экстремум для функции одной переменной?
3. В чём заключается основная идея метода? Записать формулы
4. Что такое допустимое множество?
5. Что такое целевая функция?
6. Как перевести задачу минимизации в задачу максимизации (или наоборот)?
7. Что такое inf? Чем отличается от min?
8. Что такое градиент? Какая математическая сущность? Как вычисляется?
9. Какие параметры используются в методе? Как они влияют на качество результатов/процесс получения результатов? (на этот вопрос помогают ответить графики)
10. Что такое норма вектора, матрицы?
11. Вопросы по обозначениям в формулах (например, что такое [k+1] при x[k+1])
12. Что такое критерий останова?
13. Что такое оптимизация? Запишите постановку задачи оптимизации.
14. Что означает запись $f: X \to R$? Привести примеры.
15. Как описать вектор на языке программирования?

3.5 Задачи

Выбор исходных данных для выполнения задания определяется вариантом (табл. 3.3).

Разработать программу численной оптимизации функции методом деления отрезка пополам и градиентным методом. Разрабатываемая программа должна выводить приближённое решение на каждом шаге; точное решение; количество выполненных шагов.

В результате исследования должна быть получена зависимость качества работы алгоритмов от их параметров (ϵ, δ). Шаг изменения параметров можно выбирать интуитивно, например $\epsilon_1 = 0.5,\ \epsilon_2 = 0.1,\ \epsilon_3 = 10^{-2},\ \epsilon_3 = 10^{-5}$.

Таблица 3.3 – Варианты заданий

№ Варианта	Метод деления отрезка пополам	Градиентный метод
1	$f(x) = x + 2/x,$ $x \in [0.5,3.5]$	$f(x) = 9x_1^2 + x_2^2,$ $x^{[0]} = (1,1)^T$
2	$f(x) = (x+1)^2 + (1-x)^3,$ $x \in [-2,3]$	$f(x) = 9x_1^2 + (x_2+1)^2,$ $x^{[0]} = (1,1)^T$
3	$f(x) = (x+1)^3 + (1-x)^3,$ $x \in [-2,3]$	$f(x) = 9x_1^2 + (x_2+1)^2,$ $x^{[0]} = (-2,-2)^T$
4	$f(x) = x^2 + 1/(4-3x),$ $x \in [-2,1]$	$f(x) = (x_1 + x_2^2)^2,$ $x^{[0]} = (4,4)^T$
5	$f(x) = x^{-2} + 3x^4,$ $x \in [-2,-0.3]$	$f(x) = (x_1 + x_2^2)^2,$ $x^{[0]} = (-4,3)^T$
6	$f(x) = x^2 + 4/x,$ $x \in [0.5,3.5]$	$f(x) = (x_1 + x_2^2)^2,$ $x^{[0]} = (-1,5)^T$

3.6 Содержание отчёта

Отчёт выполняется в текстовом редакторе Word, LibreOffice Writer или подобных.

Структура отчёта:

1. Титульный лист.
2. Введение и условия задания.
3. Результаты решения.
4. Результаты исследования.
5. Заключение.

Отчёт должен включать:

- условия решаемых задач, номер варианта;
- скриншоты схем;
- текст программ (не скриншоты);
- текстовые комментарии к результатам исследования и краткие выводы.

Для лучшего представления результатов рекомендуется строить графики и приводит таблицы значений.

Библиографический список

1. Ахмеров Р.Р. Методы оптимизации гладких функций. [Электронный ресурс]. – URL: http://w.ict.nsc.ru/books/textbooks/akhmerov/mo/index.html.
2. Бахвалов Н.С. Численные методы / Н.С. Бахвалов, Н.П. Жидков, Г.М. Кобельков. – М.: Наука, 1987. – 487с.
3. Боглаев Ю.П. Вычислительная математика и программирование / Ю.П. Боглаев. – М. Высшая школа, 1990. – 544с.
4. Волков Е.А. Численные методы: Учеб. Пособие для вузов – 2-е изд., испр. / Е.А. Волков. – М.: Наука, 1987. – 248 с.
5. Горячев Л.В. Одномерная минимизация. Методические указания к самостоятельной работе студентов по курсу “Методы оптимизации” кафедры процессов управления ДВГУ / Л.В. Горячев. – Владивосток. – 2003. – 35с.
6. Марчук Г.И. Методы вычислительной математики. М.: Мир, 1989. – 375с.
7. Осокин А. Метод градиентного спуска / А. Осокин. [Электронный ресурс]. – URL: http://www.machinelearning.ru/wiki/index.php?title=Метод градиентного спуска.
8. Поляк Б.Т. Введение в минимизацию / Б.Т. Поляк. – Москва: Наука. – 1983. – 384 с.
9. Эльсгольц Л.Э. Дифференциальные уравнения и вариационное исчисление. – Москва: наука. – 1969. – 424 с. ил.
10. Решение СЛАУ методом LU-разложения [Электронный ресурс]. – URL: https://habr.com/ru/sandbox/35982/.

МЕТОДЫ И АЛГОРИТМЫ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ

Методические указания
к выполнению лабораторных работ

Составитель: **Липко** Иван Юрьевич

В авторской редакции

Изд. № 305/2020. Объем 2 п.л.
РИИЦМ ФГАОУ ВО «Севастопольский государственный университет»