|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| Факультет | «Информатика и системы управления» (ИУ) |
| Кафедра | «Информационная безопасность» (ИУ8) |

**Отчёт по учебной практике**

Тип практики: учебная

Название предприятия: НУК ИУ МГТУ им. Н. Э. Баумана

|  |  |
| --- | --- |
| Студент: Месяцева Наталья Вячеславовна,  группа ИУ8Ц-41 (2 курс) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |
| Руководитель практики: старший преподаватель кафедры ИУ8 Глинская Елена Вячеславовна | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| Факультет | «Информатика и системы управления» (ИУ) |
| Кафедра | «Информационная безопасность» (ИУ8) |

**Индивидуальное задание на практику**

Тип практики: учебная

Название предприятия: НУК ИУ МГТУ им. Н. Э. Баумана

Сроки практики: с 04 июня 2019 г. по 17 июня 2019 г.

Специальность: 10.05.03 «Информационная безопасность автоматизированных систем»

Индивидуальный вариант задания

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Задание 1 | Задание 2.1 | Задание 2.2 | Задание 3 | Задание 4.1 | Задание 4.2 | Задание  4.3 | Задание  5.1 | Задание 5.2 |
| 4 | 4 | 4.1 | 4 | 4 | 4 | 4 | 4 | 4 |

|  |  |
| --- | --- |
| Студент: Месяцева Наталья Вячеславовна,  группа ИУ8Ц-41 (2 курс) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |
| Руководитель практики: старший преподаватель кафедры ИУ8 Глинская Елена Вячеславовна | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Оглавление

[Введение 4](#_Toc10185954)

[Основная часть 5](#_Toc10185955)

[Характеристика организации 5](#_Toc10185956)

[Задание №1 6](#_Toc10185957)

[Задание №2 12](#_Toc10185958)

[Задание №3 16](#_Toc10185959)

[Задание №4 23](#_Toc10185960)

[Задание №5 31](#_Toc10185961)

[Заключение 39](#_Toc10185962)

[Список использованных источников 40](#_Toc10185963)

# Введение

За время прохождения практики студенту надлежит согласно программе практики:

* **Изучить:**

**-** основные операции над векторами и матрицами в MATLAB;

- простейшие графические средства MATLAB;

- основные встроенные функции MATLAB;

- основы программирования MATLAB.

* **Получить практические навыки:**

- решения задач по вычислению суммы числового ряда и анализу погрешностей полученных результатов;

- численного решения нелинейных уравнений встроенными средствами MATLAB, в том числе задач из предметной области;

- решения систем линейных алгебраических уравнений с помощью матричных операторов MATLAB;

- решения систем линейных алгебраических уравнений методом итераций;

- численного решения задачи Коши для обыкновенного дифференциального уравнения 1-го порядка и для системы обыкновенных дифференциальных уравнений;

- численного решения задачи Коши из предметной области.

# Основная часть

## Характеристика организации

НУК ИУ образован в 1988 году. В его состав входят следующие структурные подразделения:

* Факультет ИУ «Информатика и системы управления»;
* Научно-исследовательский институт – НИИ ИСУ;
* Вычислительный центр;
* Специальные научно-учебные центры.

Факультет «Информатика и системы управления» является одним из самых востребованных в наши дни, так как имеет широкий спектр направлений в научной и практической деятельности. Это ведущий факультет по подготовке кадров в области разработки программного обеспечения, электронной и микросистемной техники для предприятий ракетно-космической отрасли, государственных служб и частных компаний. В его состав входят 11 кафедр, в том числе кафедра ИУ8 (основана в 1997 г.).

Направление подготовки «Информационная безопасность автоматизированных систем», на которой я обучаюсь и прохожу практику, входит в состав кафедры ИУ8 и занимается подготовкой специалистов, осуществляющих комплексное обеспечение информационной безопасности автоматизированных систем.

Безопасность автоматизированных систем – направление науки и техники, охватывающее совокупность программно-аппаратных, криптографических, технических и организационно-правовых методов и средств обеспечения безопасности информации в автоматизированных системах при ее обработке, хранении и передаче с использованием современных информационных технологий.

## Задание №1

“НАЧАЛО РАБОТЫ С MATLAB”

ЦЕЛЬ РАБОТЫ

1. Получение первых навыков работы с системой компьютерной математики MATLAB.
2. Знакомство с основными операциями над векторами и матрицами в MATLAB.
3. Знакомство с простейшими графическими средствами MATLAB.
4. Решение задачи по вычислению суммы числового ряда и анализу погрешностей полученных результатов.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

**Задача 1.**

Дан ряд

Найти сумму ряда аналитически.

Вычислить значения частичных сумм ряда

и найти величину погрешности при значениях N={102, 103, 104, 105}. Определить количество верных цифр результатов.

СОДЕРЖАНИЕ РАБОТЫ

1. Найдем сумму ряда S аналитически с использованием средств MATLAB[1].

Вводим символьную переменную n:

>> syms n

>> S\_inf=symsum((48/(5\*(n^2+6\*n+8))),n,0,inf)

S\_inf =

4

1. Для решения задачи сумм нам необходимо знать значение частичных сумм S(N). Решим эту задачу в общем случае:

>> syms N

>> S=symsum((48/(5\*(n^2+6\*n+8))),n,0,N)

и в результате получим:

S =

4 - (24\*(2\*N + 7))/(5\*(N + 3)\*(N + 4))  
  
Очевидно, что это означает:

1. Сформируем вектор N={102, 103, 104, 105}.

>> N=[10^2, 10^3, 10^4, 10^5]

N =

100 1000 10000 100000  
  
В результате применения вышеприведенной команды мы потеряли символьную переменную N, вместо которой теперь мы имеем вектор N из четырех элементов.

1. Вычислим значения частичных сумм Si = S (Ni) ряда при соответствующих значениях Ni.

>> S = 4 - (24.\*(2.\*N + 7))./(5.\*(N+3).\*(N+4))

S =

1. 3.9072 3.9904 3.9990 3.9999  
     
   5. Для каждой величины S ( Ni) вычислим абсолютную погрешность C:\Users\E3B7~1\AppData\Local\Temp\Rar$EXa16436.3615\matlab\Lab1\index5.files\image008.gif.   
   >> D = abs(S-4)

D =

0.0928 0.0096 0.0010 0.0001  
  
Это ***неправильный*** результат, говорящий нам о том, что при N = 105 погрешность вычисления равна нулю. Задание формата сказывается только на форме вывода чисел. Вычисления же происходят в режиме двойной точности, а ввод чисел осуществляется в любом удобном виде. Т.е. для получения правильного результата введем следующие две строки.  
  
>> format long % длинное представление в фиксированном формате (15 знаков)

>> D

D =

0.092755787901419 0.009566519564812 0.000959664119956 0.000095996640120

1. Для каждой величины S ( Ni) вычислим относительную погрешность C:\Users\E3B7~1\AppData\Local\Temp\Rar$EXa16436.3615\matlab\Lab1\index5.files\image007.gifи определим количество верных цифр.

>> format short e % Для отображения погрешности

>> d = D./4

d =

2.3189e-02 2.3916e-03 2.3992e-04 2.3999e-05

1. Введем вспомогательный вектор **a**, который в нашем случае примет вид:

>> a = [3 3 3 3]

a =

3 3 3 3  
  
Вычислим количество верных цифр:

>> n = 1-log10(a.\*d)

n =  
2.1576e+00 3.1442e+00 4.1428e+00 5.1427e+00

Здесь log10 - функция вычисения десятичного логарифма, .\* означает операцию поэлементного умножения двух векторов в отличие от скалярного произведения векторов \*.

Округлим полученные значения до целых, меньших или равных n:

>> n = floor(n)

n =

2 3 4 5

1. Запишем численные значения найденных частичных сумм, округлив их до найденного ранее количества верных цифр. Для этого нам необходимы более точные значения частичных сумм:

>> format long

>> S

S =

3.907244212098581 3.990433480435188 3.999040335880044 3.999904003359880

|  |  |  |  |
| --- | --- | --- | --- |
| S1 | S2 | S3 | S4 |
| 4,0 | 4,00 | 4,000 | 4,0000 |

Округляя полученные ранее значения S c учётом количества верных чисел, получим:  
  
Анализируя полученные значения, видим, что они не выходят за рамки абсолютной погрешности D, найденной ранее, и это значит, что результаты частичных сумм получены верно и совпадают с суммой, найденной аналитически с бесконечным пределом суммирования.

1. Для построения графика зависимости относительной погрешности в процентах от N напишем следующее:

>>semilogx(N, d\*100) % Строим логарифмический график

>> grid on % Включаем сетку на графике

>> xlabel('N') % Подписываем оси координат

>> ylabel('d(%)')  
% Даём название графику

>> title('График зависимости относительной погрешности в процентах от N')

Сумма ряда: 4

Таблица. Результаты расчета погрешностей

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| i | Ni | Частичная сумма, S(Ni) | Абсолютная погрешность, D | Относительная погрешность, d | Кол-во верных цифр, n | Частичная сумма (округленное значение), S(Ni) |
| 1 | 10^2 | 3.9072 | 0.0928 | 0.2 | 2 | 4.0 |
| 2 | 10^3 | 3.9904 | 0.0096 | 0.002 | 3 | 4.00 |
| 3 | 10^4 | 3.9990 | 0.0010 | 0.0002 | 4 | 4.000 |
| 4 | 10^5 | 3.9999 | 0.0001 | 0.00002 | 5 | 4.0000 |

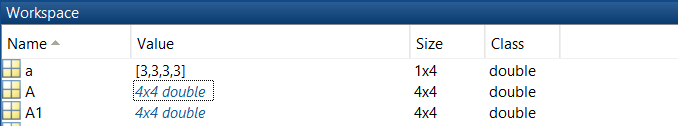
Изображение выглядит как текст

Автоматически созданное описание

ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ

1. Определить количество верных цифр для частичных сумм, используя понятие абсолютной погрешности (т.е. определение количества верных цифр). Сравнить результаты.  
     
   По определению n первых значащих цифр являются верными, если абсолютная погрешность этого числа не превышает половины разряда, выражаемого n – й значащей цифрой, считая слева направо.
2. Провести анализ рабочей области. Какие характеристики переменных здесь приведены. Привести 1 - 2 примера в отчете. Как добавить или удалить данные столбцы - характеристики (описать в отчете)?

В рабочей области (WORKSPACE) содержатся имена (NAME), значения переменных (VALUE), размер переменных (SIZE) и класс, к которому они принадлежат (CLASS). Для того, чтобы удалить или добавить столбцы, необходимо нажать правой кнопкой мыши на белое поле с названиями столбцов, после чего выпадет список характеристик (существующие в рабочей области отмечены галочкой, при удалении столбца нужно снять галочку, при добавлении – поставить).



**Вывод**

В данной лабораторной работе я получила первые навыки работы с MATLAB: ознакомилась с основными операциями над векторами и матрицами, а так с простейшими графическими средствами. Разобравшись с интерфейсом и основами работы MATLAB, было получено решение задачи по вычислению суммы числового ряда и анализу погрешностей полученных результатов.

## Задание №2

“РЕШЕНИЕ УРАВНЕНИЙ В MATLAB”

ЦЕЛЬ РАБОТЫ

1. Введение в программирование MATLAB.
2. Численное решение нелинейных уравнений в MATLAB[2].
3. Решение нелинейного уравнения из предметной области.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

**Задача 2.1.**

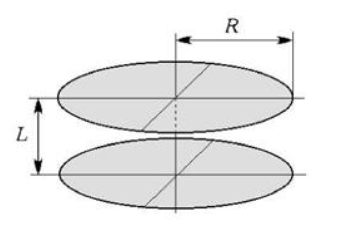
Построить график с целью нахождения отрезка локализации, на котором находится только один корень, и найти **данный** корень нелинейного уравнениях.

**Задача 2.2.**

Электрическая емкость двух коаксиальных плоских дисков (см. рисунок) при L/R<1 рассчитывается по формуле:

где ε1 − относительная диэлектрическая проницаемость среды, Ф/м, R − радиус дисков, L − расстояние между дисками, π =3,14... .

Найдите радиус R, удовлетворяющий требуемому значению емкости при заданных в таблице параметрах и L.



|  |  |
| --- | --- |
| Параметр | Вариант |
| 4-1 |
|  | 1 |
| , мм | 1 |
| , пФ | 100 |

СОДЕРЖАНИЕ РАБОТЫ

**Задача 2.1**

Файл **function\_1.m** (Функция)

function res = function\_1(x)

res = exp(x)-(1./x)-1;

end

Файл **task\_2.m**

x = -2:0.001:-1; % Определяем интересующий интервал переменной x

y = function\_1(x); % Присваиваем результат выполнения функции f(x)

plot(x, y); % Строим график функции

grid on; % Включаем сетку на графике

title('График 1'); % Даём название графику

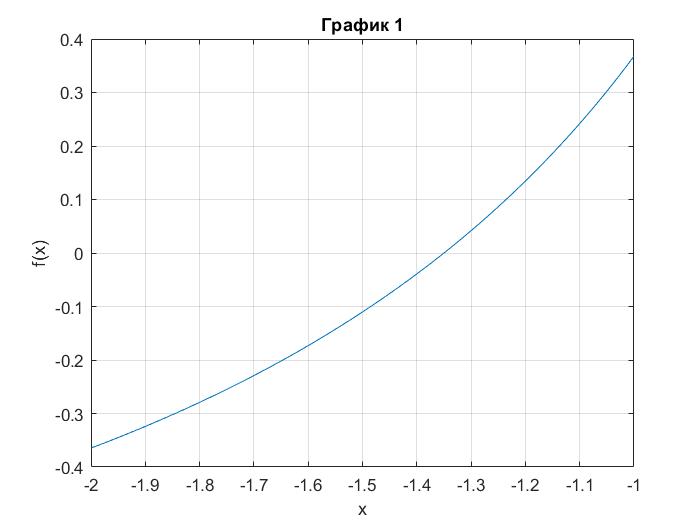
xlabel('x');

ylabel('f(x)');

% Убеждаемся в существовании корня по графику

x = fzero('f(x)', [-2; -1]); % Вычисляем корень

disp(x); % Вывод значения переменной в окно

**Результат:

x = -1.35

**Задача 2.2**

Необходимо найти радиус, при котором достигается требуемое значение ёмкости. Преобразуем формулу таким образом, чтобы использовать стандартную функцию fzero(). Для этого перенесём C в правую часть уравнения и получим:  
Будем работать с этим уравнением и искать его корень. Кроме того, переведём все переменные в СИ, чтобы получить ответ для радиуса в метрах.

Файл **function\_2.m** (Функция)

function res = function\_2(x)

e = 8.85\*(10^-12);

E=1;

L=10^-3; % Переведено в СИ

C=10^-10; % Переведено в СИ

res = e.\*E.\*x.\*((pi.\*x./L)+log(16.\*pi.\*x./L)-1)-C;

end

Файл **task\_2.m**

x = 0.01:0.001:30; % Определяем интересующий интервал переменной x

y=function\_2(x); % Присваиваем результат выполнения функции f(x)

plot(x,y); % Строим график функции

grid on; % Включаем сетку на графике

title('График 2'); % Даём название графику

xlabel('x');

ylabel('f(x)');

x = fzero('function\_2(x)', [0.01, 30]); % Вычисляем корень

disp(x); % Вывод значения переменной в окно

Изображение выглядит как текст

Автоматически созданное описаниеРезультат: R = 0.0589 м  
  
  
  **Вывод**

В данной лабораторной работе я ознакомилась с введением в программирование в MATLAB: узнала, что программы сохраняются в виде текстовых m-файлов и что программа может менять структуру алгоритмов вычислений в зависимости от входных данных и данных, создаваемых в ходе вычислений. С помощью программных средств был построен график с целью нахождения отрезка локализации, на котором находится только один корень, и найденкорень нелинейного уравнения, а также решена предметная задача.

## Задание №3

“РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ”

ЦЕЛЬ РАБОТЫ

1. Дальнейшее введение в программирование в MATLAB.
2. Решение систем линейных алгебраических уравнений с помощью матричных операторов MATLAB.
3. Решение систем линейных алгебраических уравнений методом итераций.

ПОСТАНОВКА ЗАДАЧИ

**Задача 3.1.**

Решить систему линейных алгебраических уравнений (СЛАУ), заданную матрицами

;

матричными средствами MATLAB.

Дополнение:

* Предварительно убедиться, что система уравнений невырожденная.
* Выполнить проверку решения.
* Решить задачу, используя 4 различных способа.

**Задача 3.2.**

Решить исходную систему линейных алгебраических уравнений методом простых итераций (двумя способами).

СОДЕРЖАНИЕ РАБОТЫ

**Задача 3.1**

Матричные средства MATLAB

Решим систему уравнений различными методами.

1. Использование обратной матрицы (два способа)

СЛАУ в матричной форме записывается в виде AX=B, где X = .

Отсюда следует, что если матрица невырожденная (определитель отличен от нуля), то X=A-1B.

Получить обратную матрицу в MATLAB можно двумя способами, которые описаны в файле программы **task\_3.m**.

1. Методом Гаусса (два способа)

Алгоритм метода Гаусса основан на приведении матрицы А к треугольному виду и последовательном вычислении неизвестных. Данный метод применим только для невырожденных матриц.

В MATLAB для решения системы уравнения методом исключения Гаусса существуют специальные операторы, которые описаны в файле программы **Task1.m**.

Файл **task\_3.m**

format short;

A = [5 -2 32 0; 4 25 0 -3; 20 0 2 -7; 0 0 -9 40];

B = [27; 34; -28; 5];

% Проверка системы на невырожденность

d = det(A);

if(d == 0)

disp('Матрица вырожденная, обратная матрица не существует');

else

X1 = (A^-1)\*B; % 1 способ

X2 = inv(A)\*B; % 2 способ

X3 = A \ B; % 3 способ

X4 = ((B') / A')'; % 4 способ

% Проверка решений

Check\_1 = A \* X1;

Check\_2 = A \* X2;

Check\_3 = A \* X3;

Check\_4 = A \* X4;

end

**Результат** выполнения программы:

**X1 = -1.3809 1.6273 1.1612 0.3863**

**X2 = -1.3809 1.6273 1.1612 0.3863**

**X3 = -1.3809 1.6273 1.1612 0.3863**

**X4 = -1.3809 1.6273 1.1612 0.3863**

Проверка решений. Критерий успешности – матрица совпадает с исходной матрицей B.

**Check\_1 = 27.0000 34.0000 -28.0000 5.0000**

**Check\_2 = 27.0000 34.0000 -28.0000 5.0000**

**Check\_3 = 27.0000 34.0000 -28.0000 5.0000**

**Check\_4 = 27.0000 34.0000 -28.0000 5.0000**

**Задача 3.2**

Метод простых итераций

Для решения системы уравнений необходимо привести ее к виду .

1. Первый способ

Проверяем условие сходимости итерационного процесса, т.е. является ли каждый диагональный элемент матрицы большим по модулю, чем сумма модулей остальных элементов строки, в которой он находится. При невыполнении условия меняем местами соответствующие строки матрицы .

Приводим результирующие матрицы системы уравнений:

;

Реализация первого способа метода итераций приведена в файле программы **task3\_2.m**

Файл **task3\_2.m**

format short;

% Создаем матрицы СЛАУ, приведенные к требуемому виду

A1=[20 0 2 -7;4 25 0 -3; 5 -2 32 0; 0 0 -9 40;];

B1=[-28; 34; 27; 5];

%Задание матрицы a и вектора b

a = zeros(4,4); % Создаём матрицу 4х4 из 0

b = zeros(4,1); % Создаём столбец 4х1 из 0

% Приведение уравнения к методу простых итераций

for i = 1: 4 % Запускаем цикл прохода по матрице А

b(i) = B1(i) / A1(i, i); % Заполняем вектор 'b'

a(i, i) = 0; % Диагональ матрицы 'a' заполняем 0

for j = 1:4

if (i ~= j) % Если не равно, то

a(i, j) = -A1(i, j) / A1(i, i); % Заполянем недиагональные элементы матрицы 'a'

end

end

end

% Проверка нормы

if (norm(a) < 1)

x = func3\_2(a, b, 10^(-5)); % Вычисление результата методом итераций

% В качестве параметров передаются матрица 'a', вектор 'b' и необходимая

% точность вычисления

accur = norm(A1 \* x - B1); % Вычисление погрешности

% Вывод в консоль

disp('X21 = ');

disp(x);

disp('Погрешность вычислений:');

disp(accur)

else

disp('Error: norm(a) >= 1');

end

**Файл func3\_2.m**

function res=func3\_2(A,B,eps)

% Решение системы уравнений AX=B

% методом простых итераций

% A - матрица системы уравнений

% b - вектор столбец свободных членов

% norma - норма матрицы А

% eps - точность численного решения

% Абсолютная разница между предыдущим и текущим значением

delta=eps\*(1-norm(A, 1))/norm(A, 1);

X0=B; % Задание начального приближения

X1=A\*X0+B; % Текущий результат итерации

while norm(X1-X0) > delta

X0=X1;

X1=A\*X0+B;

end

res=X1; % Возвращение результата

end

**Результат** выполнения программы:

X21 =

-1.3809

1.6273

1.1612

0.3863

Погрешность вычислений:

8.5791e-05

1. Второй способ

Для выполнения условия сходимости невырожденную систему уравнений можно заменить эквивалентной системой путем умножения исходного уравнения на матрицу , где  – матрица с малыми по модулю элементами. Последовательно получим:

Обозначим Тогда .

Все элементы матрицы нужно выбрать достаточно малыми по модулю для того, чтобы обеспечить выполнение условия

В качестве нулевого приближения можно выбрать .

Реализация второго способа метода итераций приведена в файле программы **task3\_22.m**

Файл **task3\_22.m**

format short;

% Объявление матриц

A=[5 -2 32 0; 4 25 0 -3; 20 0 2 -7; 0 0 -9 40];

B=[27; 34; -28; 5];

% Матрица малых приращений

delta = [

18^-8 10^-6 10^-4 10^-5;

10^-7 10^-5 10^-3 10^-6;

10^-9 10^-7 25^-8 2\*10^-6;

10^-6 10^-8 10^-7 2\*10^-5

];

% Вспомогательные коэффициенты

alpha = delta \* A;

beta = (A^(-1) - delta) \* B; % Вычисление вектора b

% Проверка нормы

if (norm(alpha) < 1)

x = func3\_2(alpha, beta, 10^(-5)); % Вычисление результата

accur = norm(A \* x - B); % Вычисление погрешности

% Вывод в консоль

disp('X22 = ');

disp(x);

disp('Погрешность:');

disp(accur);

else

disp('norm(alpha) >= 1');

end

**Результат** выполнения программы:

X22 =

-1.3809

1.6273

1.1612

0.3863

Погрешность:

3.5783e-06

**Вывод**

В данной лабораторной работе я продолжила знакомиться с программированием в MATLAB. Была решена система линейных алгебраических уравнений матричными средствами MATLAB и методом простых итераций (двумя способами).

## Задание №4

ЦЕЛЬ РАБОТЫ

1. Исследование методов численного интегрирования в MATLAB.
2. Изучение средств аналитического и численного интегрирования, имеющихся в MATLAB.
3. Программирование циклического алгоритмов в MATLAB.

ПОСТАНОВКА ЗАДАЧИ

**Задача 4.1.**

Вычислить значение интеграла

с помощью квадратурных формул левых прямоугольников, средних прямоугольников, трапеций и Симпсона для элементарного отрезка интегрирования. Оценить величину погрешности. Применяя те же квадратурные формулы для составного отрезка интегрирования, вычислить интеграл I с точностью 0.0001. Предварительно оценить шаг интегрирования, при котором достигается заданная точность.

**Задача 4.2.**

Вычислить заданный интеграл аналитически и используя квадратурную формулу, указанную в индивидуальном варианте, с шагом *h*=(*b-a*)/8. Оценить погрешность по правилу Рунге. Сравнить данную оценку погрешности с точным значением погрешности. Сделать выводы.

**Задача 4.3.**

**А.** Оценить погрешность численного интегрирования с помощью функции **trapz(x,y)** (метод трапеций).

**Б.** Для функции **quad('fun',a,b,tol)** (метод Симпсона) убедиться, что результат действительно вычисляется с точностью tol.

СОДЕРЖАНИЕ РАБОТЫ

**Задача 4.1**

В файле **task\_4.m** представлена реализация программы подсчёта погрешностей численных вычислений интегралов[2].

**Файл task\_4.m.**

a = 1;

b = 1.44;

format long e;

syms x;

f = x^4+x^3+x^2-0.1\*x+0.1;

Io = int(f,a,b);

I = double(Io);

disp('Точное значение:');

disp(I);

% Построение графика подынтегральной функции

Ymin = 0;

Ymax = 2;

fplot(@function\_4, [a b])

title('f(x)'), grid on

xlabel('x'), ylabel('y')

% ЧАСТЬ 1

% Вычисление интеграла по формулам

disp('###### ЧАСТЬ 1 ######');

h = (b-a);

I\_L\_rect = h \* function\_4(a);

I\_R\_rect = h \* function\_4(b);

I\_M\_rect = h \* function\_4((a+b)/2);

I\_trap = h \* ((function\_4(a)+function\_4(b))/2);

I\_simp = h/6 \* (function\_4(a)+4\*function\_4((a+b)/2)+function\_4(b));

% Вывод вычисленных значений

disp('Результаты численного интегрирования:');

disp(I\_L\_rect);

disp(I\_R\_rect);

disp(I\_M\_rect);

disp(I\_trap);

disp(I\_simp);

% Подсчёт абсолютных и относительных погрешностей

accur\_LR = abs(I\_L\_rect-I);

accur\_RR = abs(I\_R\_rect-I);

accur\_MR = abs(I\_M\_rect-I);

accur\_tr = abs(I\_trap-I);

accur\_simp = abs(I\_simp-I);

% Вывод абсолютной и относительной погрешностей

disp('Погрешности:');

disp(accur\_LR);

disp(accur\_RR);

disp(accur\_MR);

disp(accur\_tr);

disp(accur\_simp);

disp('###### ЧАСТЬ 2 ######');

% Вычисление производных

f\_diff1 = diff(f);

f\_diff2 = diff(f,2);

f\_diff4 = diff(f,4);

% Вывод производных

disp('Производные 1,2,4 порядков соответственно:');

disp(f\_diff1);

disp(f\_diff2);

disp(f\_diff4);

x = a:0.0001:b;

m1 = 2.\*x.^3 + (21.\*x.^2)./10 + (8.\*x)./5 + 9./10;

m2 = 6.\*x.^2 + (21.\*x)./5 + 8./5;

m4 = 12;

% Вычисление максимумов

M1 = max(abs(m1));

M2 = max(abs(m2));

M4 = max(abs(m4));

R = 0.0001;

h\_LR = 2\*R/M1/(b-a);

h\_RR = h\_LR;

h\_MR = (24\*R/M2/(b-a))^(1/2);

h\_tr = (12\*R/M2/(b-a))^(1/2);

h\_simp = (2880\*R/M4/(b-a))^(1/4);

% Вывод погрешностей

disp('Вывод погрешностей');

disp(h\_LR);

disp(h\_RR);

disp(h\_MR);

disp(h\_tr);

disp(h\_simp);

% Количество узлов интегрирования

disp('Количество узлов интегрирования');

disp(ceil((b-a)/h\_LR));

disp(ceil((b-a)/h\_RR));

disp(ceil((b-a)/h\_MR));

disp(ceil((b-a)/h\_tr));

disp(ceil((b-a)/h\_simp));

Файл **function\_4.m**

function res = function\_4(x)

% Функция ff(x)= x.^4+x.^3+x.^2-0.1.\*x+0.1

res = x.^4+x.^3+x.^2-0.1.\*x+0.1;

end

**Результат** выполнения программы:

**Точное значение:**

**2.515616191146667e+00**

**###### ЧАСТЬ 1 ######**

**Результаты численного интегрирования:**

**1.320000000000000e+00**

**4.098776422399999e+00**

**2.418936326399999e+00**

**2.709388211199999e+00**

**2.515753621333332e+00**

**Погрешности:**

**1.195616191146667e+00**

**1.583160231253332e+00**

**9.667986474666712e-02**

**1.937720200533328e-01**

**1.374301866658989e-04**

**###### ЧАСТЬ 2 ######**

**Производные 1,2,4 порядков соответственно:**

**4\*x^3 + 3\*x^2 + 2\*x - 1/10**

**12\*x^2 + 6\*x + 2**

**24**

**Вывод погрешностей**

**3.359406628813410e-05**

**3.359406628813410e-05**

**1.647758792657165e-02**

**1.165141416047640e-02**

**4.832697830906221e-01**

**Количество узлов интегрирования**

**13098**

**13098**

**27**

**38**

**1**

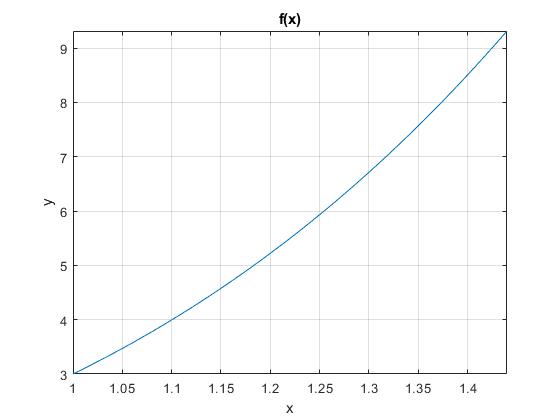
****

График подынтегральной функции

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Метод вычислений** | **Элементарный метод** | | | | **Составной метод.**  **Заданная погрешность**  **R = 0.0001** | | |
| **Эксперимент** | | **Теория** | |
| **** | **** | **** | **** | **n** | **h** | **** |
| Левых прямоуг - ков | 1.1956e+00 | 3.2688e-01 | 6.8652e+00 | 1.1001e+00 | 13098 | 6.8652e+04 | 3.3594e-05 |
| Средних прямоуг - ков | 9.6679e-02 | 2.1835e-02 | 5.5459e-01 | 8.8873e-02 | 27 | 7.5000e+01 | 1.6477e-02 |
| Трапеций | 1.9377e-01 | 4.3751e-02 | 1.1092e+00 | 1.7775e-01 | 38 | 1.0600e+02 | 1.1651e-02 |
| Симпсона | 1.3743e-04 | 2.6953e-05 | 8.7955e-04 | 1.4095e-04 | 1 | 2.0000e+00 | 4.8326e-01 |

**Задача 4.2**

Файл **task4\_2.m**

% Вычисление интеграла аналитически

syms x;

f = 1+cos(x);

a = 0; b = 1;

Io = int(f,a,b);

I = double(Io);

h =(b-a)/8;

disp('Аналитический расчет определённого интеграла');

disp(I);

I\_theor = h \* f4\_2((a+b)/2);

disp('Теоритический расчет определённого интеграла');

disp(I\_theor);

N = 8; % Количество узлов

H = (b - a) / N; % Размер элементарного отрезка интегрирования

I\_2n = H \* f4\_2((a+b)/2);

delta\_Runge = abs(I\_2n - I\_theor)/15;

disp('Погрешность по правилу Рунге');

disp(delta\_Runge);

delta\_Abs = abs(I\_theor- I);

disp('Абсолютная погрешность');

disp(delta\_Abs);

Файл **f4\_2.m** (Функция)

function res = f4\_2(x)

res = 1 + cos(x);

end

Результат:

**Аналитический расчет определённого интеграла**

**1.8415**

**Теоретический расчет определённого интеграла**

**0.2347**

**Погрешность по правилу Рунге**

**0**

**Абсолютная погрешность**

**1.6068**

**Задача 4.3**

Файл **task4\_3.m**

% Функция x^2\*e^(3x)-51;

syms x;

f = x^(2)\*exp(3\*x)-51;

a = 0; b = 1;

I0 = int(f,a,b);

I = double(Io);

for n\_trapz = 10:70:178

h\_trapz = (b - a) / n\_trapz; % Элементарный отрезок

x = [a:h\_trapz:b]; % Пределы интегрирования

I\_ch = trapz(x, func4\_3(x));

accur\_abs = abs(I-I\_ch);

accur\_otn = I\_ch / I;

disp('Количество узлов, результат, абс. и отн. погрешности:');

disp(n\_trapz);

disp( I\_ch);

disp(accur\_abs);

disp( accur\_otn);

end

Файл **func4\_3.m**

function res = func4\_3(x)

res = x.^2.\*exp(3.\*x)-51;

end

Результат:

**Метод трапеций  
Количество узлов, результат, абс. и отн. погрешности:**

**10**

**-47.2711**

**49.1126**

**-25.6703**

**Количество узлов, результат, абс. и отн. погрешности:**

**80**

**-47.3532**

**49.1947**

**-25.7149**

**Количество узлов, результат, абс. и отн. погрешности:**

**150**

**-47.3542**

**49.1956**

**-25.7154**

**Вывод**

В данной лабораторной работе я освоила навыки аналитического и численного интегрирования с помощью средств MATLAB. Результаты решенных задач показали, что с погрешность уменьшается с увеличением количества узлов и точное значение погрешности не превышает оценку погрешности.

## Задание №5

ЦЕЛЬ РАБОТЫ

1. Численное решение задачи Коши для обыкновенного дифференциального уравнения 1-го порядка.
2. Численное решение задачи Коши для системы обыкновенных дифференциальных уравнений.
3. Численное решение задачи Коши из предметной области.
4. Введение в программирование в MATLAB. Продолжение.

ПОСТАНОВКА ЗАДАЧИ

**Задача 5.1.**

Построить график и вывести в виде таблицы решение задачи Коши на интервале [0; 1] методом Рунге-Кутта 4-го порядка[1].

**Задача 5.2.** Вертикальные колебания механической системы  под действием последовательности полусинусоидальных импульсов описывается дифференциальным уравнением вида

где x - отклонение системы от исходного положения, t - время, m - масса блока, β - коэффициент трения, k - коэффициент жесткости амортизаторов, Fm и ω - параметры вынуждающей силы.

Решите уравнение для следующих данных:  масса m = 3 к*г*; коэффициент трения β = 1 к*г*/с, коэффициент жесткости k = 4 Н/м. Начальные условия x = 0 и dx/dt = 0 при t = 0. Остальные параметры даны в таблице.

Получите участок решения x(t), на котором устанавливаются устойчивые колебания в системе. Постройте зависимости F(t) = |Fmcos(ωt)| и x(t).

СОДЕРЖАНИЕ РАБОТЫ

**Задача 5.1**

Решение дифференциального уравнения методом Рунге-Кутта и построение графика решения

Файл **task\_5.m**

[X, Y] = ode45(@function\_5 , [0 1] , (0.3) ); % Запускаем метод Рунге-Кутта на отрезке [0, 1] с начальным значением функции в т. 0 равным 0.3 и записываем полученные значения переменных в массив

% Дескриптор @ обеспечивает связь с файлом - функцией правой части

% [0 1] - интервал, на котором необходимо получить решение

% (0.3) - начальное значение решения

plot(X, Y); % Строим график

grid on; % Включаем сетку на графике

hold on; % Включаем режим сохранения графика

% Ставим в любом месте графика его название с помощью мышки

title('График численного решения задачи Коши');

hold off; % Отключаем режим сохранения графика

format short; % Включаем формат короткого представления (5 знаков)

disp([X, Y]);

% Последняя команда выводит таблицу численного решения задачи

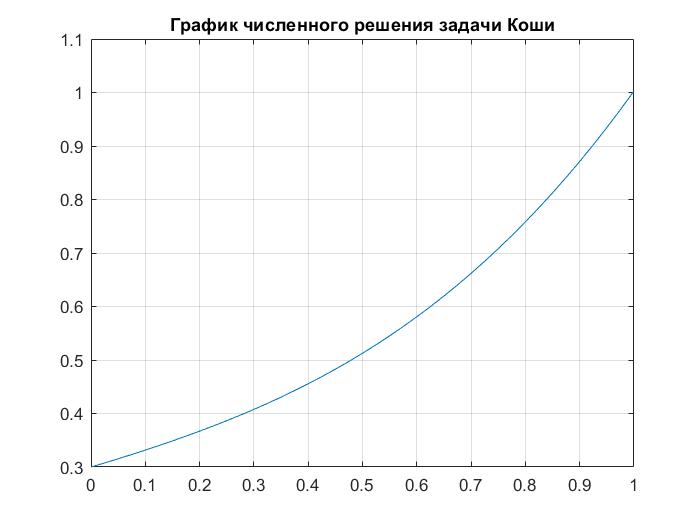
Файл **function\_5.m** (Функция)

function dydx = function\_5(x, y) % Имя функциии F, параметры x и y, возвращаемое значение dydx

dydx = zeros(1, 1);

dydx(1) = ((y^2)+(x^3))^0.5;

end

Результат:

**ans =**

**0 0.3000**

**0.0250 0.3076**

**0.0500 0.3154**

**0.0750 0.3234**

**0.1000 0.3316**

**0.1250 0.3400**

**0.1500 0.3487**

**0.1750 0.3577**

**0.2000 0.3670**

**0.2250 0.3766**

**0.2500 0.3866**

**0.2750 0.3969**

**0.3000 0.4077**

**0.3250 0.4189**

**0.3500 0.4306**

**0.3750 0.4428**

**0.4000 0.4556**

**0.4250 0.4689**

**0.4500 0.4828**

**0.4750 0.4974**

**0.5000 0.5126**

**0.5250 0.5285**

**0.5500 0.5452**

**0.5750 0.5626**

**0.6000 0.5808**

**0.6250 0.5998**

**0.6500 0.6196**

**0.6750 0.6404**

**0.7000 0.6620**

**0.7250 0.6846**

**0.7500 0.7081**

**0.7750 0.7326**

**0.8000 0.7582**

**0.8250 0.7848**

**0.8500 0.8124**

**0.8750 0.8412**

**0.9000 0.8711**

**0.9250 0.9022**

**0.9500 0.9345**

**0.9750 0.9680**

**1.0000 1.0028**

**Задача 5.2**

Все физические величины даны в СИ и в дополнительных преобразованиях не нуждаются. Имеем ЛНДУ 2-го порядка. Составим систему из двух уравнений:

где .

Файл **task5\_2.m**

xo = [0, 0]; % Задаём начальные условия

tspan = [0, 20];

% Численное решение задачи Коши

[T, X] = ode45(@function5\_2, tspan, xo);

plot(T, X); % Строим график

grid on; % Включаем сетку на графике

format short;

disp([T, X]);

a=0:0.01:120;

f = abs(150\*cos(0.1\*a));

figure;

plot(a, f);

Файл **function5\_2.m** (Функция)

function dydx = function5\_2(t, x)

m = 3;

b = 1;

k = 4;

f = 150;

w = 0.1;

dydx = zeros(2, 1);

dydx(1) = x(2);

dydx(2) = (abs(f\*cos(w\*t))-k\*x(1)-b\*x(2))/m;

end

Результат:

**Ans =   
 0 0 0**

**0.0000 0.0000 0.0001**

**0.0000 0.0000 0.0001**

**0.0000 0.0000 0.0002**

**0.0000 0.0000 0.0002**

**0.0000 0.0000 0.0005**

**0.0000 0.0000 0.0007**

**0.0000 0.0000 0.0010**

**0.0000 0.0000 0.0012**

**0.0000 0.0000 0.0025**

**0.0001 0.0000 0.0037**

**0.0001 0.0000 0.0050**

**0.0001 0.0000 0.0062**

**0.0003 0.0000 0.0125**

**0.0004 0.0000 0.0188**

**0.0005 0.0000 0.0251**

**0.0006 0.0000 0.0313**

**0.0013 0.0000 0.0627**

**0.0019 0.0001 0.0941**

**0.0025 0.0002 0.1255**

**0.0031 0.0002 0.1569**

**0.0063 0.0010 0.3136**

**0.0094 0.0022 0.4702**

**0.0126 0.0039 0.6266**

**0.0157 0.0062 0.7828**

**0.0314 0.0246 1.5613**

**0.0471 0.0552 2.3353**

**0.0628 0.0979 3.1044**

**0.0785 0.1526 3.8685**

**0.1464 0.5256 7.1079**

**0.2142 1.1146 10.2313**

**0.2821 1.9112 13.2222**

**0.3499 2.9057 16.0654**

**0.4730 5.1807 20.8010**

**0.5961 8.0015 24.9315**

**0.7191 11.2899 28.3980**

**0.8422 14.9621 31.1587**

**1.0044 20.2408 33.6783**

**1.1666 25.8217 34.9054**

**1.3288 31.4947 34.8613**

**1.4911 37.0625 33.6148**

**1.6756 43.0396 30.8697**

**1.8602 48.3922 26.9072**

**2.0448 52.9110 21.9698**

**2.2293 56.4488 16.3284**

**2.3870 58.6210 11.1661**

**2.5447 59.9670 5.8808**

**2.7024 60.4791 0.6491**

**2.8601 60.1803 -4.3678**

**3.0179 59.1184 -9.0248**

**3.1756 57.3601 -13.1877**

**3.3333 54.9913 -16.7471**

**3.4910 52.1134 -19.6240**

**3.6635 48.5145 -21.9281**

**3.8360 44.5985 -23.3121**

**4.0085 40.5260 -23.7733**

**4.1810 36.4503 -23.3508**

**4.3990 31.5115 -21.6642**

**4.6171 27.0679 -18.8760**

**4.8351 23.3430 -15.2334**

**5.0532 20.4803 -11.0240**

**5.2136 18.9732 -7.7436**

**5.3740 17.9957 -4.4388**

**5.5344 17.5441 -1.2242**

**5.6948 17.5944 1.7982**

**5.8552 18.1072 4.5383**

**6.0156 19.0309 6.9152**

**6.1760 20.3022 8.8652**

**6.3364 21.8497 10.3458**

**6.5064 23.7053 11.3759**

**6.6764 25.6870 11.8395**

**6.8464 27.6984 11.7472**

**7.0164 29.6493 11.1324**

**7.2285 31.8771 9.7105**

**7.4407 33.7350 7.6835**

**7.6528 35.1048 5.2049**

**7.8650 35.9152 2.4449**

**8.0609 36.1369 -0.2030**

**8.2569 35.8421 -2.8061**

**8.4528 35.0491 -5.2385**

**8.6487 33.8038 -7.3969**

**8.8102 32.4832 -8.9101**

**8.9717 30.9410 -10.1406**

**9.1332 29.2251 -11.0621**

**9.2947 27.3860 -11.6619**

**9.4939 25.0227 -11.9587**

**9.6932 22.6490 -11.7820**

**9.8924 20.3572 -11.1714**

**10.0917 18.2250 -10.1897**

**10.3193 16.0631 -8.7121**

**10.5470 14.2701 -6.9798**

**10.7746 12.8939 -5.1294**

**11.0023 11.9409 -3.2931**

**11.1581 11.5211 -2.1086**

**11.3139 11.2785 -1.0239**

**11.4697 11.1951 -0.0694**

**11.6255 11.2490 0.7314**

**11.7813 11.4146 1.3612**

**11.9371 11.6639 1.8072**

**12.0929 11.9677 2.0636**

**12.2487 12.2968 2.1316**

**12.4230 12.6601 1.9938**

**12.5972 12.9805 1.6466**

**12.7714 13.2229 1.1143**

**12.9456 13.3586 0.4276**

**13.1343 13.3592 -0.4503**

**13.3231 13.1845 -1.4200**

**13.5118 12.8206 -2.4317**

**13.7005 12.2653 -3.4382**

**13.8697 11.6103 -4.2991**

**14.0388 10.8154 -5.0900**

**14.2079 9.8942 -5.7853**

**14.3770 8.8648 -6.3657**

**14.5706 7.5803 -6.8714**

**14.7642 6.2159 -7.1937**

**14.9577 4.8079 -7.3280**

**15.1513 3.3914 -7.2807**

**15.3292 2.1143 -7.0642**

**15.5071 0.8799 -6.7672**

**15.6850 -0.2951 -6.4091**

**15.8628 -1.3843 -5.8187**

**16.0407 -2.3385 -4.8389**

**16.2186 -3.0908 -3.5652**

**16.3965 -3.5935 -2.0667**

**16.5744 -3.8148 -0.4173**

**16.7304 -3.7627 1.0942**

**16.8865 -3.4737 2.6119**

**17.0425 -2.9499 4.0869**

**17.1986 -2.2020 5.4755**

**17.3612 -1.2028 6.7887**

**17.5237 -0.0042 7.9266**

**17.6863 1.3630 8.8585**

**17.8489 2.8638 9.5639**

**18.0219 4.5658 10.0532**

**18.1950 6.3282 10.2674**

**18.3680 8.1033 10.2122**

**18.5411 9.8470 9.9049**

**18.7391 11.7524 9.2775**

**18.9371 13.5077 8.4033**

**19.1351 15.0679 7.3408**

**19.3332 16.4042 6.1540**

**19.4999 17.3434 5.1073**

**19.6666 18.1076 4.0591**

**19.8333 18.6989 3.0467**

**Изображение выглядит как текст, карта

Автоматически созданное описание 20.0000 19.1266 2.1031**

# Изображение выглядит как сидит Автоматически созданное описание

**Вывод**

В данной лабораторной работе я научилась решать обыкновенные дифференциальные уравнения (ОДУ) и системы ОДУ, используя функцию ode45 (метод Рунге-Кутта 4 и 5-го порядков) в программе MATLAB.

# Заключение

MatLab – MATrix LABoratory – это система, предназначенная для осуществления любых численных расчётов и моделирования технических и физических систем, а также выполнения научных и инженерных расчётов при работе с массивами данных. Система использует математический сопроцессор. Также MATLAB – это одновременно и операционная среда, и высокоуровневый язык программирования. Одна из наиболее сильных сторон системы состоит в том, что на языке MATLAB могут быть написаны программы для многократного использования. Пользователь может сам написать специализированные функции и программы, которые оформляются в виде m-файлов.  
В лабораторных работах мы ознакомились с основными возможностями данной среды программирования:

1. Работа с системой компьютерной математики MATLAB и с простейшими графическими средствами, операции над векторами и матрицами в MATLAB, вычислению суммы числового ряда и анализ погрешностей полученных результатов.  
2. Программирование в MATLAB, решение систем линейных алгебраических уравнений с помощью матричных операторов, численное решение нелинейных уравнений.  
3. Решение систем линейных алгебраических уравнений с помощью матричных операторов MATLAB, решение систем линейных алгебраических уравнений методом итераций.  
4. Изучение средств аналитического и численного интегрирования, имеющихся в MATLAB.  
5. Численное решение задачи Коши для обыкновенного дифференциального уравнения 1-го порядка и для системы обыкновенных дифференциальных уравнений.

# Список использованных источников

**I. Основная литература**   
**1.**Курбатова Е.А. MATLAB 7. Самоучитель. Изд-во: Вильямс. 2005.  
**2.** Половко А. М., Бутусов П. Н. MATLAB для студента. — СПб.: БХВ-Петербург, 2005. —320 с.  
**3.**Дьяконов, В. Matlab 6 : Учебный курс / В. Дьяконов. — СПб. : Питер, 2001. — 592 с. : ил. — (Учебный курс).

**II. Internet - ресурсы**  
www.exponenta.ru - образовательный математический сайт.  
www.matlab.ru/matlab/default.asp- Консультационный центр MATLAB: Раздел "MATLAB" .