**Оглавление**

Цель учебной практики. ..................................................................................3 1.Решение нелинейных уравнений. ................................................................4 2.Интерполяция................................................................................................12 3.Интегрирование. ...........................................................................................19 Заключение........................................................................................................25

**Цель учебной практики**

1. Закрепление теоретических знаний, полученных студентами на лекционных, лабораторных и практических занятиях;

2. Развитие навыков самостоятельной работы по анализу информационных материалов печатных и электронных источников;

3. Закрепление навыков программирования на языке MATLAB;

4. Освоение работы с функциями и массивами данных, получение навыков построения интерфейса пользователя.

**Раздел 1**

**19 вариант**

Решение нелинейного уравнения методом простых итераций.

Требуется решить уравнение методом простых итераций. Для отделения корней использовать аналитический метод.

Согласно варианту а = 5, b = 2, с = 3, d = 0,2

**Постановка задачи:**

Пусть функция y = f(t) непрерывна на промежутке [a; b]. Требуется найти корни, такие, что f(t) = 0. Для решения этой задачи требуется использовать метод простых итераций, при котором входными данными будут являться:

𝜀 - точность вычислений;

**Суть метода численной реализации:**

Метод простых итераций заключается в том, что функцию преобразуют к виду: t = φ(t). Задается интервал, в котором и будут производиться расчеты и начальное приближение. Процесс строится по схеме . Вычисления оканчивается, как только выполнится условие: || ≤ 𝜀, то есть, как только будет достигнута заданная точность.

Описание ручного счета тестового примера и аналитический метод отделения корней:

Найдем корни уравнения: . Для этого выполним следующие шаги:

dF/dt =

16 - 480 = -464 <0

Тогда функция возрастает на всей области определения, значит, уравнение имеет лишь один корень, 𝑡 = 0, тогда 𝑓(0) = −0.2 < 0; 𝑡 = 1, 𝑓(1) = 9,8 > 0.

Возьмем промежуток [0;1]. Первым элементом последовательности в промежутке [a; b] будет являться середина промежутка, то есть. Следующие элементы будут считаться по формуле , где c = .

Итак, пусть 𝜀 = 0,1, промежуток [0;1], тогда:

|0.5 - 0.306| 0.1; получим, что 0,194 0.1, неверно.

Повторяем вычисления:

Повторяем вычисления:

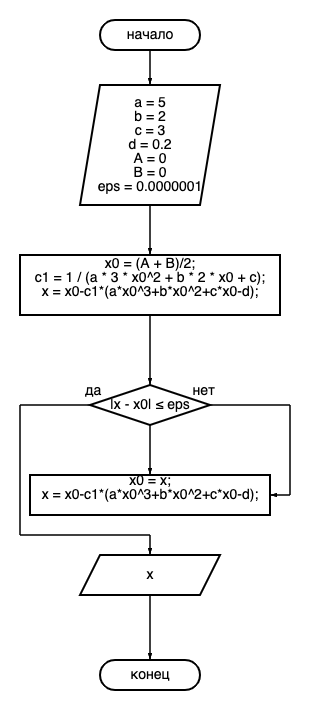
Повторяем вычисления:

Повторяем вычисления:

Повторяем вычисления:

Повторяем вычисления:

Блок-схема:



**Основная функция:**

function func()

%Необходимы коэффициенты a, b, c, d; отрезок - A,B; точность - eps

%Первое х0

while 1

a1 = check('Введите коэффицент а = ');

a = str2double(a1);

break

end

while 1

b2 = check('Введите коэффицент b = ');

b = str2double(b2);

break

end

while 1

c3 = check('Введите коэффицент c = ');

c = str2double(c3);

break

end

while 1

d4 = check('Введите коэффицент d = ');

d = str2double(d4);

break

end

while 1

A6 = check('Введите коэффицент A = ');

A = str2double(A6);

break

end

while 1

B7 = check('Введите коэффицент B = ');

B = str2double(B7);

break

end

while 1

eps5 = check('Введите коэффицент eps = ');

eps = str2double(eps5);

break

end

x0 = (A + B)/2;

c1 = 1 / (a \* 3 \* x0^2 + b \* 2 \* x0 + c);

%Первая итерация

x = x0-c1\*(a\*x0^3+b\*x0^2+c\*x0-d);

%Итерации

while abs(x - x0) >= eps

x0 = x;

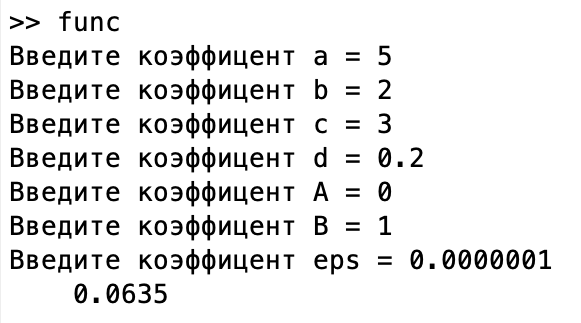
x = x0-c1\*(a\*x0^3+b\*x0^2+c\*x0-d);

end

%Вывод результата

disp(x);

end



**Функция проверки:**

function a = check(str)

while 1

a = input(str, 's');

x = str2double(a);

if (isnan(x))

disp('Содержатся символы или пробел')

else

break

end

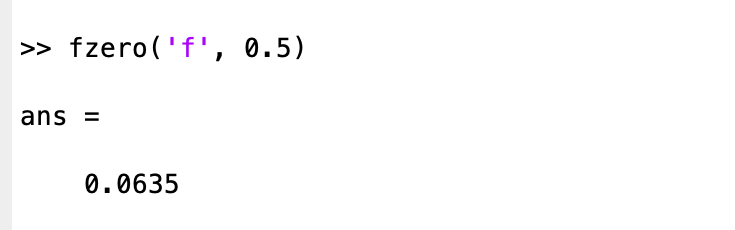
end

**Встроенная функция Matlab:**

function y = f(x0)

y = 5 \* x0^3 + 2 \* x0^2 + 3 \* x0 - 0.2;

end



**Вывод:** Задача была решена двумя способами, написанной и встроенной пользовательскими функциями MATLAB. Ответы оказались довольно схожими, поэтому мы можем сделать вывод, что функции можно считать с большой точностью.

**Раздел 2**

**19 вариант**

Интерполирование функции по формулам Ньютона.

**Математическая постановка задачи:**

С помощью интерполяционного многочлена в форме Ньютона для заданной табличной функции найти значения в точках 1,25; 3,55;





Так как заданные точки принадлежат промежутку из таблицы, найти значения по интерполяции возможно.

**Составим интерполяционный многочлен**:

+

Где 𝑛 – степень полинома, 𝑓(𝑥0, … 𝑥𝑘) *–* разделенная разность 𝑘-того порядка.

**Ручной счёт**:

Для примера построения возьмём первые три узла. Тогда многочлен будет выглядеть следующим образом:

𝑃(𝑥) = 𝑓(𝑥0) + 𝑓(𝑥0, 𝑥1) ∗ (𝑥 − 𝑥0) + 𝑓(𝑥0, 𝑥2) ∗ (𝑥 − 𝑥0) ∗ (𝑥 − 𝑥1);

Этот многочлен будет проходить через первые три заданные таблицей точки.

Распишем:

Тогда:

= 0,893

Получаем наш многочлен:

𝑃(𝑥) = 7,277 + 0,893(𝑥 − 1.2) + 0,874(𝑥 − 1.2)(𝑥 − 1.5) =

=;

Сделаем проверку и подставим искомое значение.

𝑃(1.2) = 7,277;

𝑃(1.5) = 7,545;

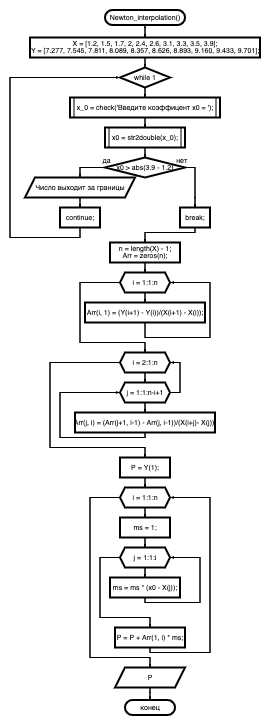
𝑃(1.7) = 7,811;

Наконец, найдем значение в точке 1.25:

## 𝑃(1.25) = 7,310725;

## **Блок-схема**:

На следующей странице

Входные данные: точки и значения функций в этих точках, промежуточная точка в которой требуется найти значение

Выходные данные: значение функции в заданной промежуточной точке

**Реализация в системе MATLAB:**

function Newton\_interpolation()

X = [1.2, 1.5, 1.7, 2, 2.4, 2.6, 3.1, 3.3, 3.5, 3.9];

Y = [7.277, 7.545, 7.811, 8.089, 8.357, 8.626, 8.893, 9.160, 9.433, 9.701];

while 1

x\_0 = check('Введите коэффицент x0 = ');

x0 = str2double(x\_0);

if x0 > abs(3.9 - 1.2)

disp('Число выходит за границы');

continue;

end

break;

end

n = length(X) - 1;

Arr = zeros(n);%Создаем матрицу с 0

%Заполняем первый столбец начальными разностями

for i = 1:1:n

Arr(i, 1) = (Y(i+1) - Y(i))/(X(i+1) - X(i));

end

%Заканчиваем заполнять матрицу с разностями

for i = 2:1:n

for j = 1:1:n-i+1

Arr(j, i) = (Arr(j+1, i-1) - Arr(j, i-1))/(X(i+j)- X(j));

end

end

%Считаем каждое слагаемое и прибавляем к ответу

P = Y(1);

for i = 1:1:n

ms = 1;

for j = 1:1:i

ms = ms \* (x0 - X(j));

end

P = P + Arr(1, i) \* ms;

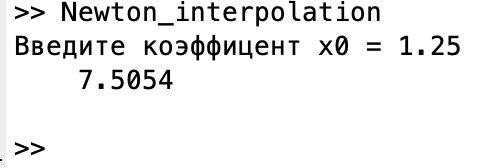
end

%Выводим ответ

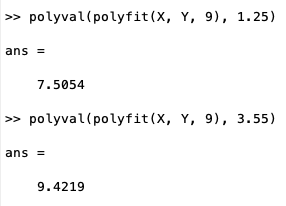
disp(P);

end

Пример работы пользовательской функции.



Проверка с помощью встроенной функции MATLAB.



Используем две функции в комбинации, где 9 — это количество точек, по которым строится полином, а 1.25 и 3.55 точки, в которых необходимо найти значение.

**Вывод:**

Так как значения в заданных точках встроенной и написанной функций совпадают мы может сказать, что погрешность в наших измерениях достаточно мала, соответственно, мы нашли близкие к истинным значениям табличной функции.

**Раздел 3**

**Численное интегрирование методом Симпсона.**

19 вариант

***Математическая постановка задачи:***

Найти значение интеграла Лапласа с заданной точностью методом

Симпсона.

Точность 𝐸𝑝𝑠 = 0.001, интеграл , 𝐶 = 0.061.

**Погрешность считается с помощью метода Рунге:**

Погрешность вычисления значения интеграла при числе шагов, равном 2n, определяется по формуле Рунге:  
{\displaystyle \Delta \_{2n}\approx \Theta |I\_{2n}-I\_{n}|}, где {\displaystyle \Theta ={\frac {1}{3}}} {\displaystyle \Theta ={\frac {1}{15}}}

**Метод Симпсона:**

Подсчет интеграла Лапласа просто так не произвести, поэтому мы пользуемся одним из численных методов - методом Симпсона.

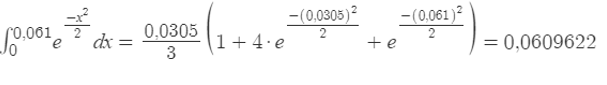
Интервал разбиваем на 2𝑛 (четное количество) отрезков одинаковой длины, после чего применяется следующая формула:

Где h =

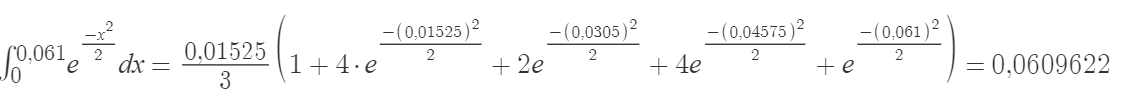
По итогу мы умножаем, кроме начала и конца, значения с четным индексом на 2, а с нечётным на 4 и всё складываем.

Ручной счет.  
Возьмём 2 и 4 отрезка. Начнем с 2.

h = = 0.0305;

**

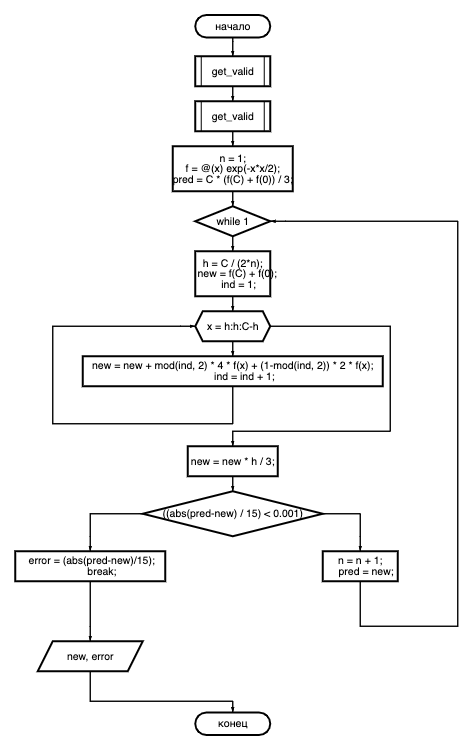
Для случая с разбиением интервала интегрирования на 4 отрезка:



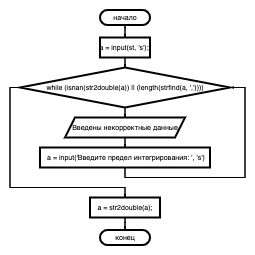
Ответы из-за округления до тысячной получились одинаковые, но уточним, что чем больше узлов, тем точнее.

**Входные данные:** предел интегрирования, точность интегрирования  
**Выходные данные:** посчитанный интеграл

Блок-схема самой пользовательской функции: Переход на следующую страницу.



Блок-схема подфункции get\_valid:



Код пользовательской функции:

main:

C = get\_valid('Введите предел интегрирования: '); % предел интегрирования

e = get\_valid('Введите точность интегрирования: '); % точность интегрирования

n = 1;

f = @(x) exp(-x\*x/2); % интегрируемая функция

pred = C \* (f(C) + f(0)) / 3; % посчитаем значение интеграла при h = C

while (1)

h = C / (2\*n); % считаем шаг интегрирования

new = f(C) + f(0); % начальное значение интеграла

ind = 1; % индекс х-итого

for x = h:h:C-h

new = new + mod(ind, 2) \* 4 \* f(x) + (1-mod(ind, 2)) \* 2 \* f(x); % реализацияформулы

ind = ind + 1; % сдвиг индекса х-итого

end

new = new \* h / 3; % умножаем на шаг интегрирования

if ((abs(pred-new) / 15) < 0.001) % если достигнута требуемая точность

error = (abs(pred-new)/15);

break; % выйти из цикла

end

n = n + 1; % если выход из цикла не произошел - увеличиваем точность при помощи уменьшения шага

pred = new; % предыдущий = новый

end

disp("Значение, найденное при помощи метода Симпсона:")

disp(new); % вывод посчитанного значения интеграла

disp (['Погрешность: ',num2str(error)]);

-----------------------------------------------------------------------------

get\_valid:

function a = get\_valid

a = input('Введите предел интегрирования: ', 's');

while (isnan(str2double(a)) || (length(strfind(a, ','))))

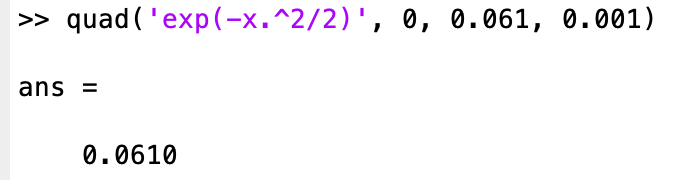
disp()

a = input('Введите предел интегрирования: ', 's');

end

a = str2double(a);

end



Вывод.  
Значение интеграла Лапласа с помощью численного метода Симпсона было найдено мною довольно точно, встроенная функция дала похожий на наш ответ.

**Заключение:** В ходе выполнения учебной практики, я научился пользоваться встроенными функциями MATLAB, а также производить там расчеты. Мои знания математики и пользования информационным материалом были расширены и дополнены. В целом, знания программирования в MATLAB были отточены и закреплены.

**Список используемой литературы:**

1.[https://ru.wikipedia.org/wiki/Метод\_простой\_итерации](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%BE%D0%B9_%D0%B8%D1%82%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8)

2. [https://ru.wikipedia.org/wiki/Интерполяционные\_формулы\_Ньютона](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D0%BE%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%84%D0%BE%D1%80%D0%BC%D1%83%D0%BB%D1%8B_%D0%9D%D1%8C%D1%8E%D1%82%D0%BE%D0%BD%D0%B0)

3.[https://ru.wikipedia.org/wiki/Формула\_Симпсона](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D1%83%D0%BB%D0%B0_%D0%A1%D0%B8%D0%BC%D0%BF%D1%81%D0%BE%D0%BD%D0%B0)

4.[https://ru.wikipedia.org/wiki/Правило\_Рунге](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B0%D0%B2%D0%B8%D0%BB%D0%BE_%D0%A0%D1%83%D0%BD%D0%B3%D0%B5)