|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное государственное автономное образовательное учреждение высшего образования  «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   1. АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ» |

Кафедра № 43 Компьютерных технологий и программной инженерии

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ОТЧЁТ ПО ПРАКТИКЕ  ЗАЩИЩЁН С ОЦЕНКОЙ  Руководитель |  | | | |
| Доцент, к.ф.-м.н., доцент |  |  |  | М. В. Фаттахова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

ОТЧЁТ ПО ПРАКТИКЕ

|  |  |  |
| --- | --- | --- |
| вид практики | Учебная | |
| тип практики | ознакомительная | |
| на тему индивидуального задания | | Реализация численных методов в среде MATLAB |
|  | |  |

|  |  |
| --- | --- |
| выполнен | Костяковым Никитой Андреевичем |
| фамилия, имя, отчество обучающегося в творительном падеже | |

|  |  |  |  |
| --- | --- | --- | --- |
| по направлению подготовки | 09.03.04 |  | Программная инженерия |
|  | код |  | наименование направления |
| направленности | 01 |  | Разработка программно-инфомационных систем |
|  | код |  | наименование направленности |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обучающийся группы № | 4134к |  | 14.05.22 |  | Н.А. Костяков |
|  | номер |  | подпись, дата |  | инициалы, фамилия |

Санкт–Петербург 2022

Оглавление

Описание работы3

1 Раздел Метод Ньютона4

**2 Раздел Интерполяция12**

3 Раздел Метод трапеций21

Источники27

***Цель работы:*** Применение вычислительных методов на языке MATLAB

***Задачи:*** 1) Решить нелинейное уравнение методом Ньютона

2) Интерполировать функцию методом Лагранжа

3) Численное интегрирование методом трапеций

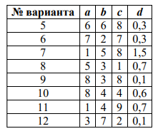
***Ход решения***

***1 раздел***

**Выполняю задание под вариантом 7**

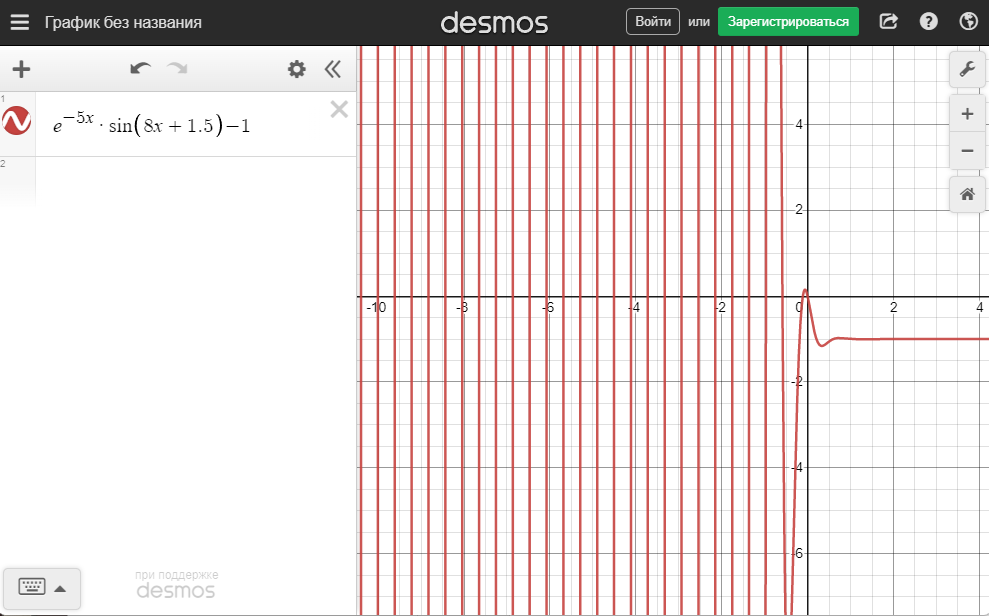
**Варианты 5-12 Решение нелинейного уравнения метод Ньютона (методом касательных)**

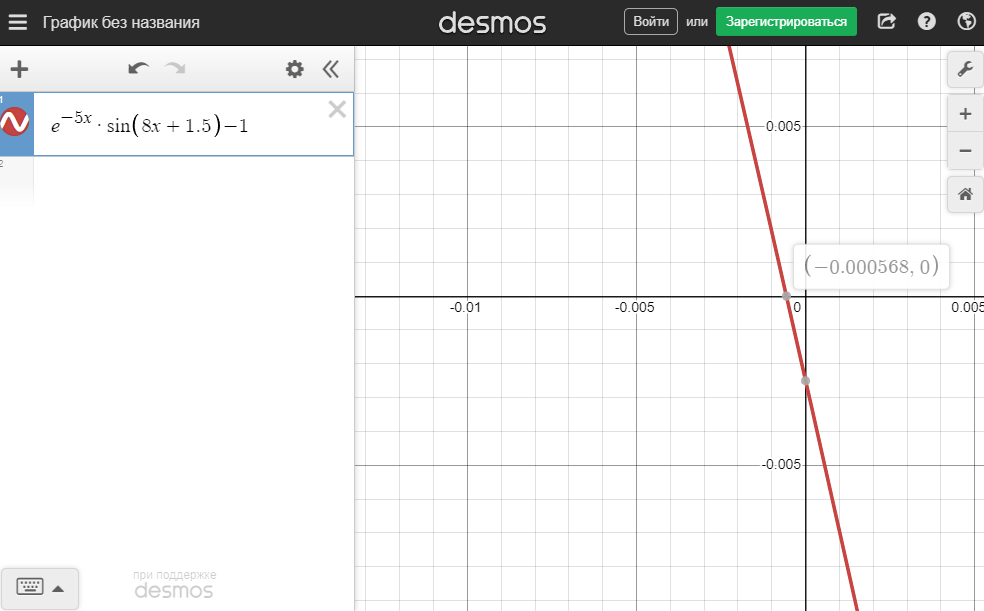
**Решите уравнения *ae-btsin(ct+d)=1* методом касательных. Для отделения корней используйте графический метод**



[1] Чтобы решить уравнение методом простой итерации, его необходимо привести к эквивалентному уравнению х=ϕ(х), где ϕ – сжимающее отображение. Для наилучшей сходимости метода в точке очередного приближения должно выполняться условие ϕ`(х)=0. Тогда функция ϕ(х) определяется

**Анализ функции**  
Проведем графическое исследование функции в графическом калькуляторе Desmos[7]:

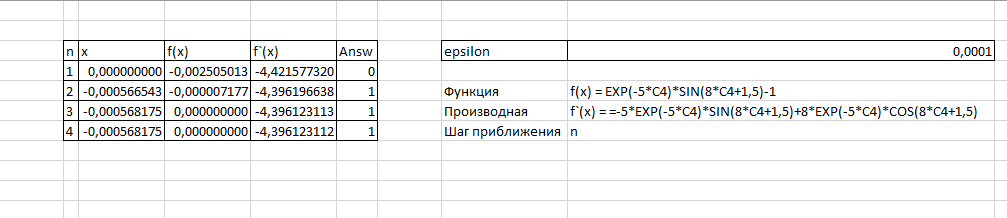


Наглядно видно, что корни имеют периодичность. Также имеется корень, крайне близко лежащий к началу оси:

Заметим, что в точке х = 0, значение функции не обращается в 0.

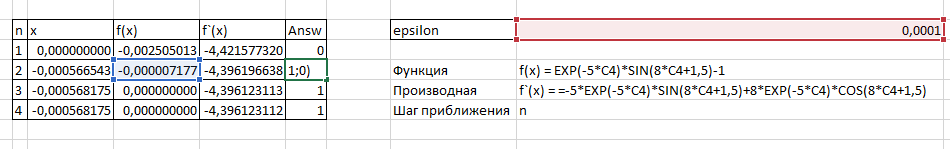
**Ход решения**  
  
Алгоритм- [2]

**Общий вид готового решения:**



   
Для упрощения восприятия вводится переменная Answ, которая изменяет свое значение на один, если корень удовлетворяет точности eps :

**Формула Answ:**



Таким образом, первое значение Answ, равное единице, указывает на то, что мы нашли корень

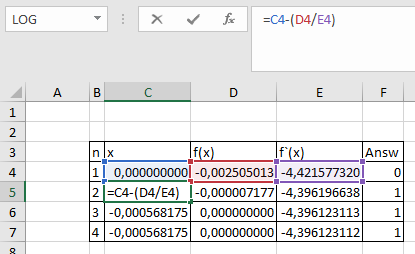
**Ход решения самого уравнения по алгоритму:**  
1) Считаем значение функции и её производной в точке х = 0 (начало графика)



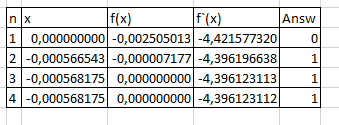
Значение функции близко к нулю, но больше значения точности eps (0.0001)

**2) Приступаем к следующему шагу точности**

Теперь за х возьмем следующее значение по формуле: x1 = x0 – (f(x)/f1(x)):



**Получаем следующий x = -0,000566543**

  
В этой точке значение функции уже достаточно близко к нулю, чтобы пренебречь погрешностью.

Модуль f(x)<0.0001, значит следующие приближения можно не выполнять. Таким образом мы нашли приближенное значение

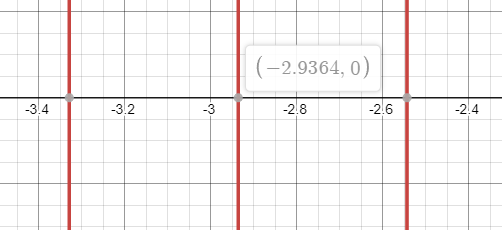
**Для проверки сравним результат с программой Desmos**



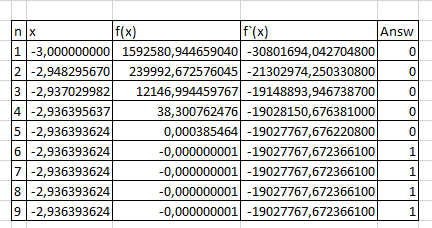
Получаем корень **0.000568**, что отличается от полученного нами корня на **~0.0000015**, заметим, что при следующем приближении n=3 , функция равна нулю, а корень по точности совпадает с корнем по Desmos и даже превышает на 3 знака после запятой.

Таким образом мы нашли ближайший корень к х = 0. Можно найти и ближайший корень для любой точки.

**Попробуем еще раз, но для x = -3**

**По Desmos** [7]

**Моё решение**



На этот раз понадобилось 6 шагов приближения, ответ удовлетворяет точности

***Листинг программы :***

***%adv\_input***

function adv\_input = adv\_input(str)

%проверка, что пользователь ввел только цифры

while 1

error = 0;

data = str2double(input(str, 's'));

res = isnan(data);

for i = 1:1:length(res)

if res(i)==1 & data(i)~="."

fprintf("ОШИБКА, введено не число\n")

error = 1;

break;

end

end

if error ==0

adv\_input = data;

break

end

end

end

***%f***

function f =f(x, a, b, c ,d)

%функция, согласно варианту 7

f = a\*exp(1)^(-b\*x)\*sin(c\*x+d)-1;

end

***%f1***

function f1 = f1(x,a,b,c,d)

%производная функции варианта 7

f1 =a\*-b\* exp(1)^(-b\*x)\*sin(c\*x+d)+c\*a\*exp(1)^(-b\*x)\*cos(c\*x+d);

end

***%solve***

function solve = solve(x, eps,a,b,c,d)

%функция решения методом Ньютона

n=0; %номер приближения

while abs(f(x,a,b,c,d)/f1(x,a,b,c,d))>eps %привближаем х по методу ньютона, пока не будет достигнута точность

n= n+1;

x = x-f(x,a,b,c,d)/f1(x,a,b,c,d);

end

solve = x %возвращаем значение

end

***%main***

a = 0.3; %входные параметры

b = 1.3;

C = 0.042;

D = 0.8;

n=100; %количество трапеций

var = input("Ведите 1, если вы хотите задать свои значения, иначе на вход пойдут данные согласно варианту: ")

if var == 1

a = adv\_input("Введите нижнюю границу: ")

b = adv\_input("Введите верхнюю границу: ")

C = adv\_input("Введите С: ")

D = adv\_input("Введите D: ")

end

dx = (b-a) / n; %рассчет шага

x = a:dx:b;

func = (1./sqrt(C.\*x.\*x+D));% что будем считать

F = 0; %сумма

i = 1; %итератор

while i <= n

s(i) = abs(func(i)+func(i+1))/2\*(x(i+1)-x(i));

F = F + s(i) ;%сумма площади всех трапеций

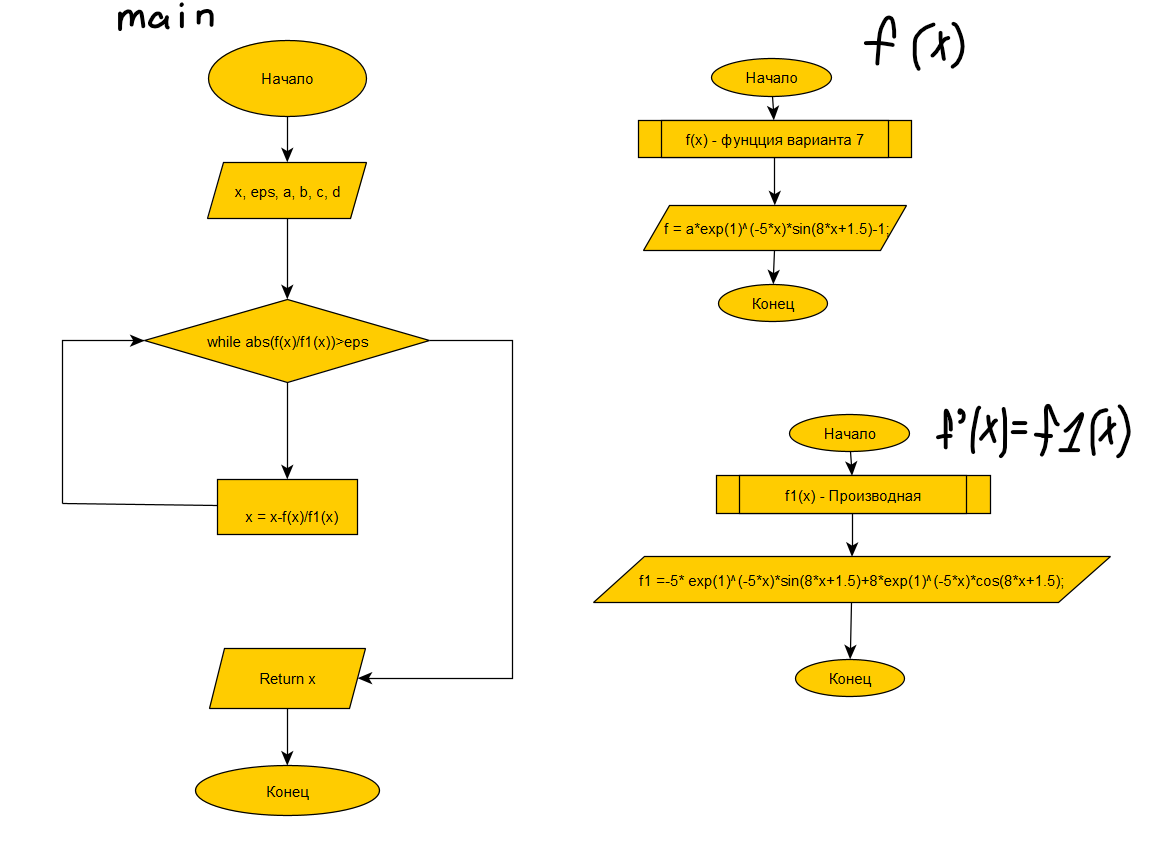
i = i +1;

end

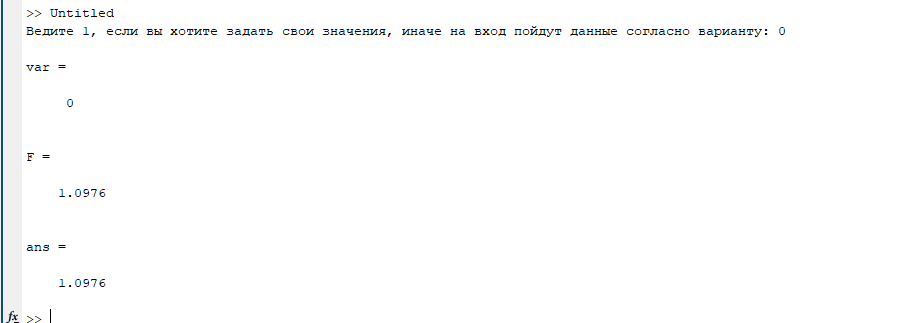
F %вывод

fun = @(x) (1./sqrt(C.\*x.\*x+D));% создание анонимной функции

integral(fun,a,b) %матлабовский интеграл

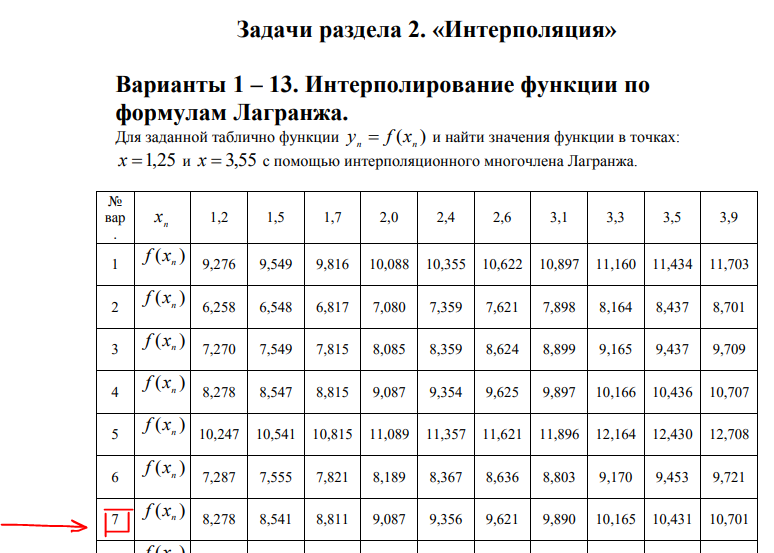
***Блок-схема***

***Результат работы программы***



**Вывод по разделу 1:** Результаты работы функции, написанной мной совпадают с результатами работы матлабовской функции с точностью до 0.0001

***2 раздел «Интерполяция»***

**Вариант 7**

**Для заданной таблично функции найти значения фунции в точках х = 1,25 и х=3,55 с помощью интерполяционного многочлена Лагранжа**

**Анализ метода решения:**

[3]Интерполяция методом Лагранжа позволяет найти значение функции в точке, основываясь на значениях в других известных точках. В отличии от обычного нахождения среднего значения(Линейного) , Метод Лагранжа описывает и кривые линии, что является безусловным преимуществом.

**Описание формулы:**

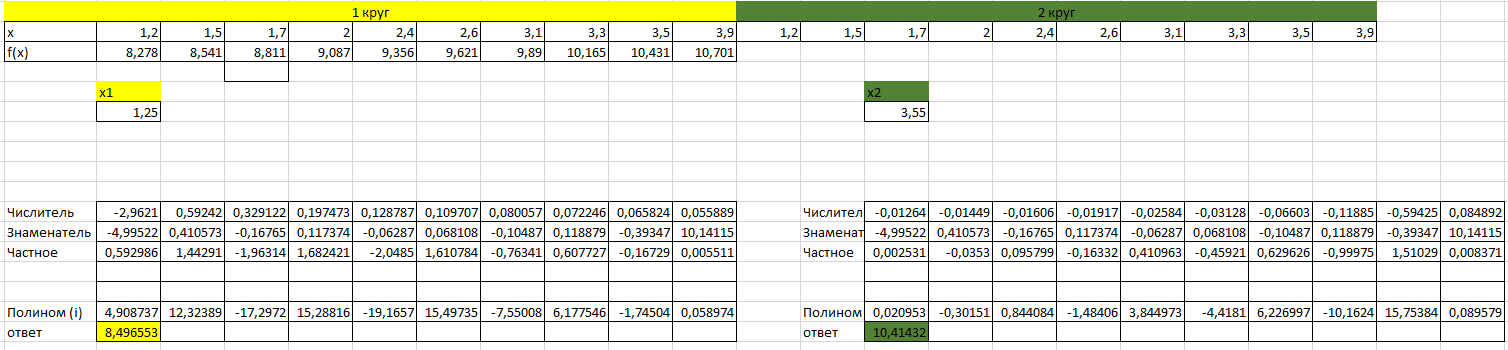
**Где базисные полиномы определяются по формул**

**Для любого i многочлен L имеет степень n и**

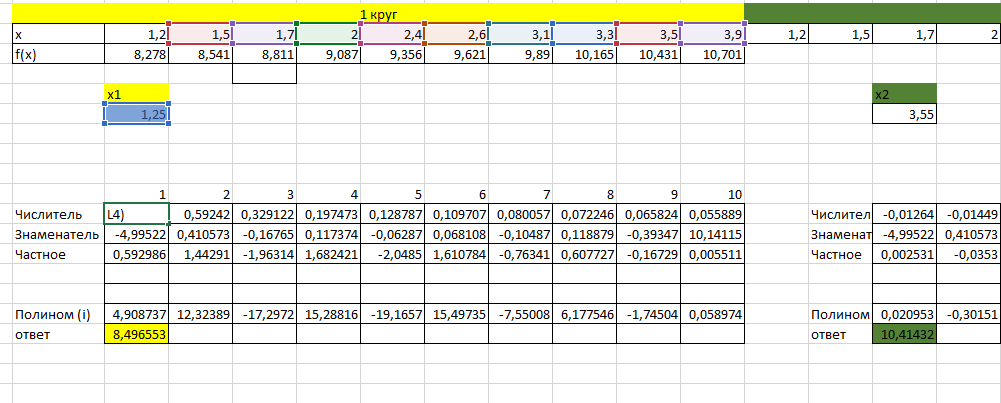
Алгоритм [4]

**1.1)Перенесем все данные в таблицу**

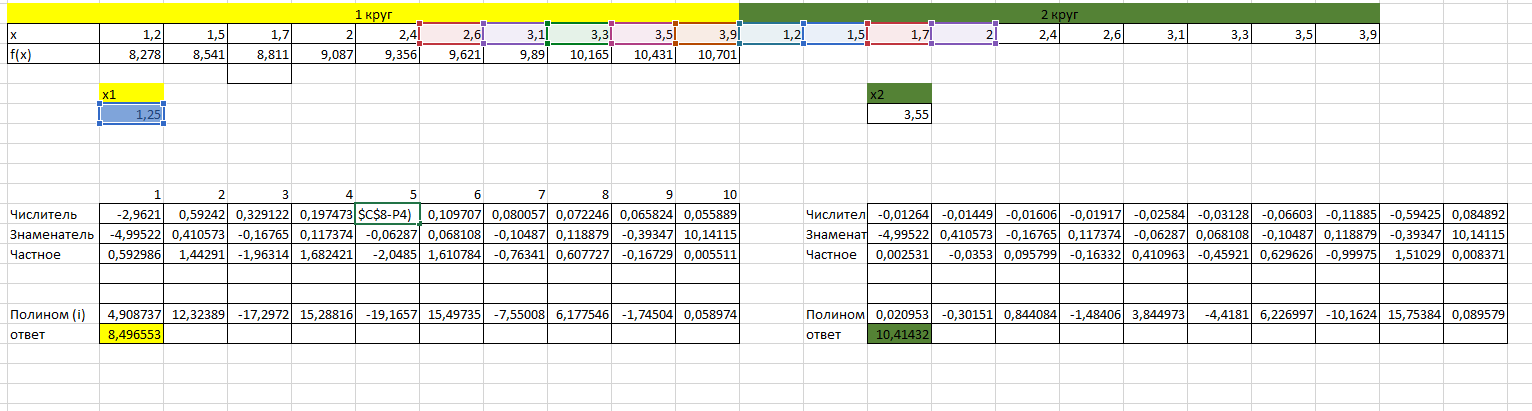


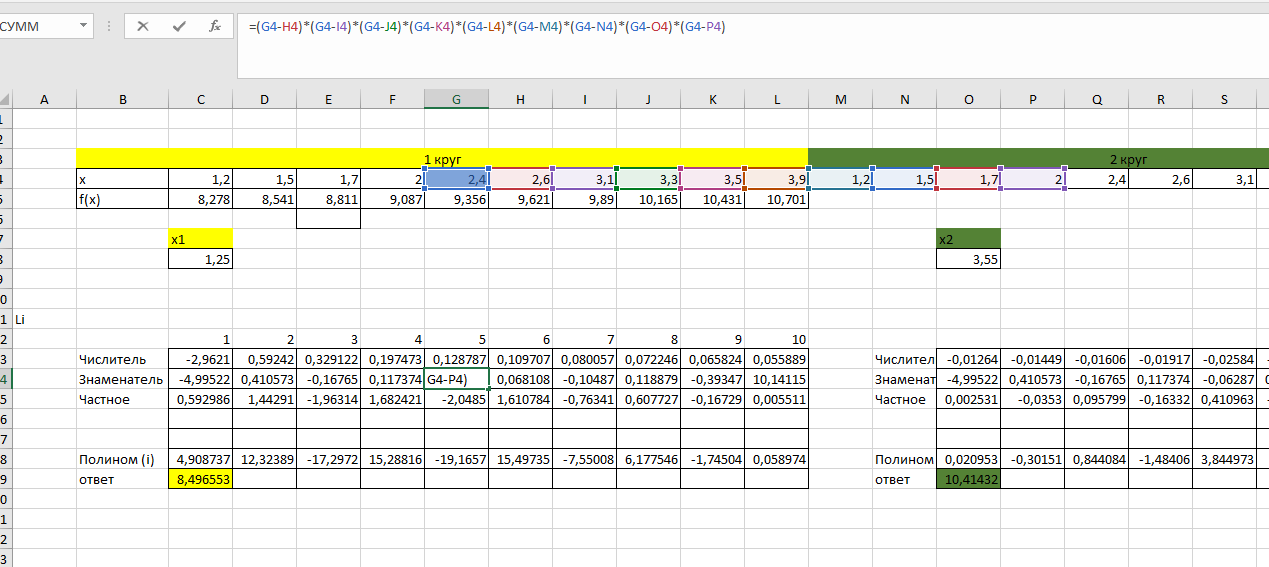
**1.2)Конечный вид решения:**   
  
  
2) Чтобы автоматизировать процесс вычисления хоть как-то, запишем числитель и знаменатель каждого полинома в отдельные ячейки. Так как формулы для каждой ячейки отличаются только порядковым номером, по которому исключается одна из 10 переменных, можем продлить ряд на 1 круг. Теперь у нас 20 ячеек, а каждый элемент X повторяется 2 раза, что облегчит написание формулы

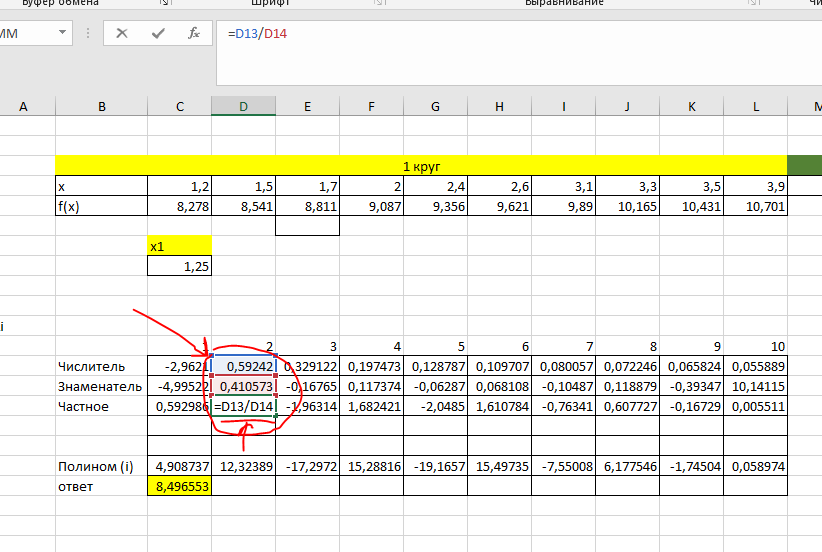
3)Так первый элемент охватывает всю полоску кроме 1 ячейки:



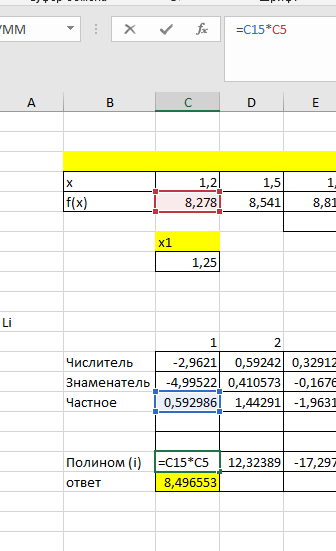
4) А 5 заходит на следующий круг, но не берет лишних ячеек:



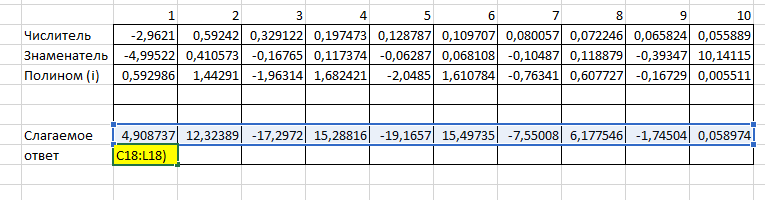
**5) Аналогично со знаменателем :**

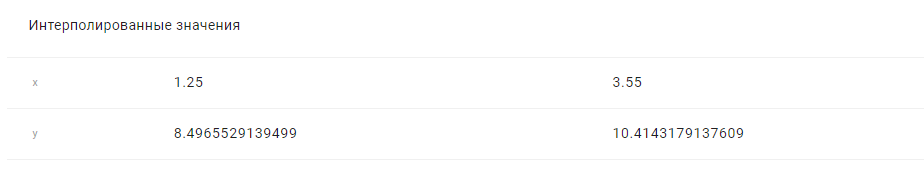
**6) Находим частное** 

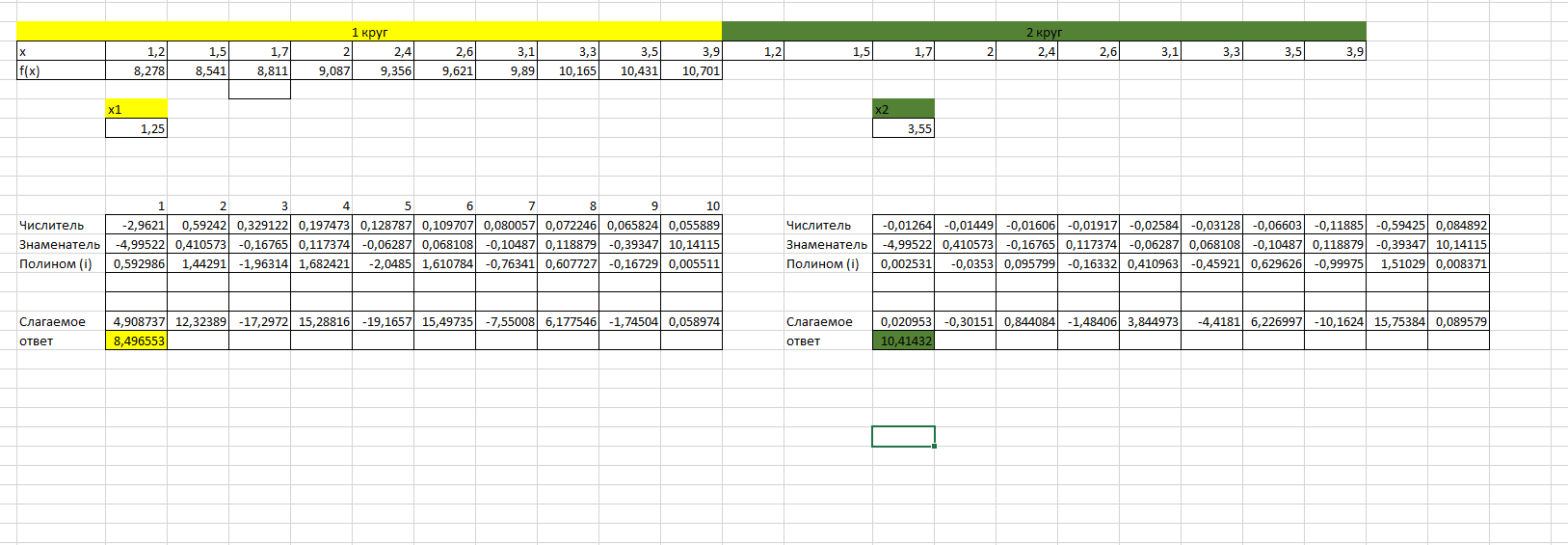
Полином – и есть частное этих выражений



**Ответ** – сумма всех произведений полиномов и значений функций, аналогично для x=3.55



Проверим на онлайн калькуляторе [5]



**Ответы совпадают**

***Листинг программы***

***%adv\_input***

function adv\_input = adv\_input(str)

%проверка, что пользователь ввел только цифры

while 1

error = 0;

data = str2num(input(str, 's'));

res = isnan(data);

for i = 1:1:length(res)

if res(i)==1 & data(i)~="." & data(i)~=" "

fprintf("ОШИБКА, введено не число\n")

error = 1;

break;

end

end

if error ==0

adv\_input = data;

break

end

end

end

**%lagr**

function lagr = lagr(x, y, s)

%Поиск по Лагранжу

res=0;

for i =1:1:length(x)

upst(i)=1; %числитель

downst(i)=1; %знаменатель

for m = 1:1:length(x)

if m~=i

upst(i)=(s-x(m))\*upst(i); %заполняем

downst(i) = downst(i)\* (x(i)-x(m));

end

end

poly(i) = upst(i)/downst(i); %шаг за шагом по формуле Лагранжа получаем результат

slag(i) =poly(i)\*y(i);

res = res+ slag(i);

end

lagr=res %вывод

end

***%main***

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

y = [8.278 8.541 8.811 9.087 9.356 9.621 9.89 10.165 10.431 10.701];

var = input("Ведите 1, если вы хотите задать свой массив, иначе на вход пойдут данные согласно варианту: ")

if var == 1

x = adv\_input("Введите вектор с узлами х через пробел: ")

y = adv\_input("Введите вектор со значениями y через пробел: ")

s = adv\_input("Введите, что вы ищите: ")

lagr(x,y,s)

interp1(x,y,s)

else

lagr(x,y,1.25);

lagr(x,y,3.55);

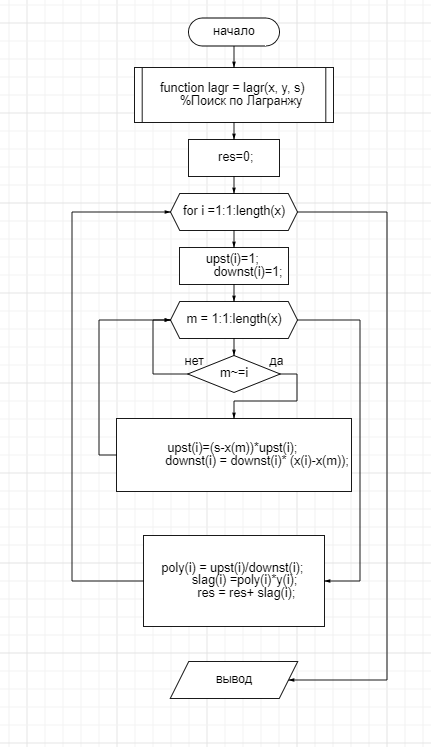
interp1(x,y,1.25)

interp1(x,y,3.55)

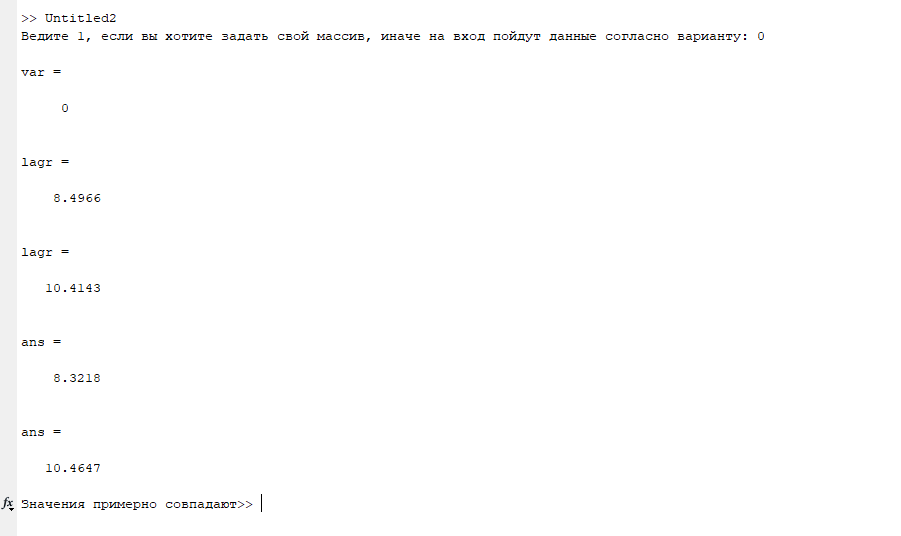
fprintf("Значения примерно совпадают")

end

**Блок-схема**

****

**Результат работы программы**

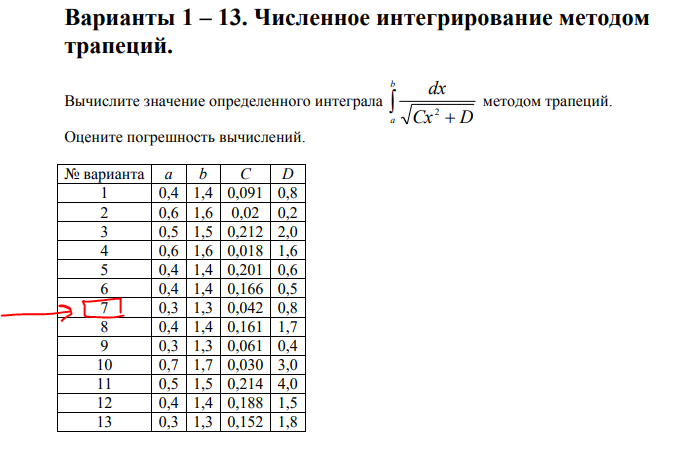


***Вывод по разделу 2:***  вывод переменной lagr – результат полученный по моим вычислениям. Можно заметить, что вычисления отличаются на значение около 0.17 и 0.05

***Раздел 3***

**Раздел 3**

**Цель вычислить определенный интеграл методом трапеций, оценить погрешность вычислений.**

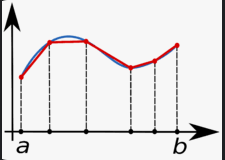


**Для начала посмотрим через онлайн калькулятор [6], имеет ли интеграл решение. Ответ по калькулятору – 1.09762013…**

**Имеет.**

Описание метода [8]:

Метод Трапеций схож с методом прямоугольников. Отличие заключается в том, что верхние вершины задаются значением функции в этой точке



Нужно разбить График на несколько частей, высчитать площадь каждого сектора, а затем сложить получившиеся площади

Площадь трапеции = Полусумма оснований \* высоту

**Так график выглядит в Desmos** [7]

Данный Метод без проблем применяется к этой функции. Нужно разбить сектор на n частей и сложить площади каждой трапеции. Это и будет ответом.

**Если разбить график на 100 равных частей, получим следующие площади:**

0.0111531309638994 0.0111513250617859 0.0111494618245865 0.0111475413390861 0.0111455636946546 0.0111435289832360 0.0111414372993364 0.0111392887400119 0.0111370834048563 0.0111348213959884 0.0111325028180389 0.0111301277781373 0.0111276963858978 0.0111252087534061 0.0111226649952047 0.0111200652282789 0.0111174095720416 0.0111146981483185 0.0111119310813329 0.0111091084976899 0.0111062305263609 0.0111032972986671 0.0111003089482634 0.0110972656111221 0.0110941674255153 0.0110910145319984 0.0110878070733925 0.0110845451947671 0.0110812290434216 0.0110778587688685 0.0110744345228136 0.0110709564591386 0.0110674247338819 0.0110638395052200 0.0110602009334481 0.0110565091809607 0.0110527644122327 0.0110489667937984 0.0110451164942330 0.0110412136841314 0.0110372585360887 0.0110332512246791 0.0110291919264356 0.0110250808198291 0.0110209180852474 0.0110167039049743 0.0110124384631678 0.0110081219458391 0.0110037545408310 0.0109993364377958 0.0109948678281739 0.0109903489051715 0.0109857798637387 0.0109811609005469 0.0109764922139669 0.0109717740040461 0.0109670064724862 0.0109621898226203 0.0109573242593898 0.0109524099893227 0.0109474472205090 0.0109424361625790 0.0109373770266797 0.0109322700254514 0.0109271153730045 0.0109219132848968 0.0109166639781090 0.0109113676710221 0.0109060245833937 0.0109006349363342 0.0108951989522836 0.0108897168549874 0.0108841888694734 0.0108786152220278 0.0108729961401713 0.0108673318526358 0.0108616225893401 0.0108558685813666 0.0108500700609373 0.0108442272613901 0.0108383404171548 0.0108324097637299 0.0108264355376580 0.0108204179765028 0.0108143573188250 0.0108082538041586 0.0108021076729872 0.0107959191667214 0.0107896885276718 0.0107834159990310 0.0107771018248444 0.0107707462499902 0.0107643495201550 0.0107579118818100 0.0107514335821883 0.0107449148692611 0.0107383559917149 0.0107317571989282 0.0107251187409483 0.0107184408684683

Их Сумма равна 1.0976, что совпадает с результатом работы калькулятора выше.

**Абсолютная погрешность метода трапеций:**

По этой формуле погрешность составляет 0.00000040836

***Листинг программы***

***%adv\_input***

function adv\_input = adv\_input(str)

%проверка, что пользователь ввел только цифры

while 1

error = 0;

data = str2num(input(str, 's'));

res = isnan(data);

for i = 1:1:length(res)

if res(i)==1 & data(i)~="." & data(i)~=" "

fprintf("ОШИБКА, введено не число\n")

error = 1;

break;

end

end

if error ==0

adv\_input = data;

break

end

end

end

***%acc - погрешность***

function acc = acc(a, b,c,d, n)

x= a:(b-a)/n:b;

i=1;

while i <=n

diff(i)= -(c\*(d-2\*c.\*x.\*x))/(c.\*x.\*x+d).^(5/2);

i=i+1;

end

acc = max(abs(diff))\*((b-a)^3)/(12\*n^2);

end

***%main***

a = 0.3; %входные параметры

b = 1.3;

C = 0.042;

D = 0.8;

n=100; %количество трапеций

var = input("Ведите 1, если вы хотите задать свои значения, иначе на вход пойдут данные согласно варианту: ")

if var == 1

a = adv\_input("Введите нижнюю границу: ")

b = adv\_input("Введите верхнюю границу: ")

C = adv\_input("Введите С: ")

D = adv\_input("Введите D: ")

end

dx = (b-a) / n; %рассчет шага

x = a:dx:b;

func = (1./sqrt(C.\*x.\*x+D));% что будем считать

F = 0; %сумма

i = 1; %итератор

while i <= n

s(i) = abs(func(i)+func(i+1))/2\*(x(i+1)-x(i));

F = F + s(i) ;%сумма площади всех трапеций

i = i +1;

end

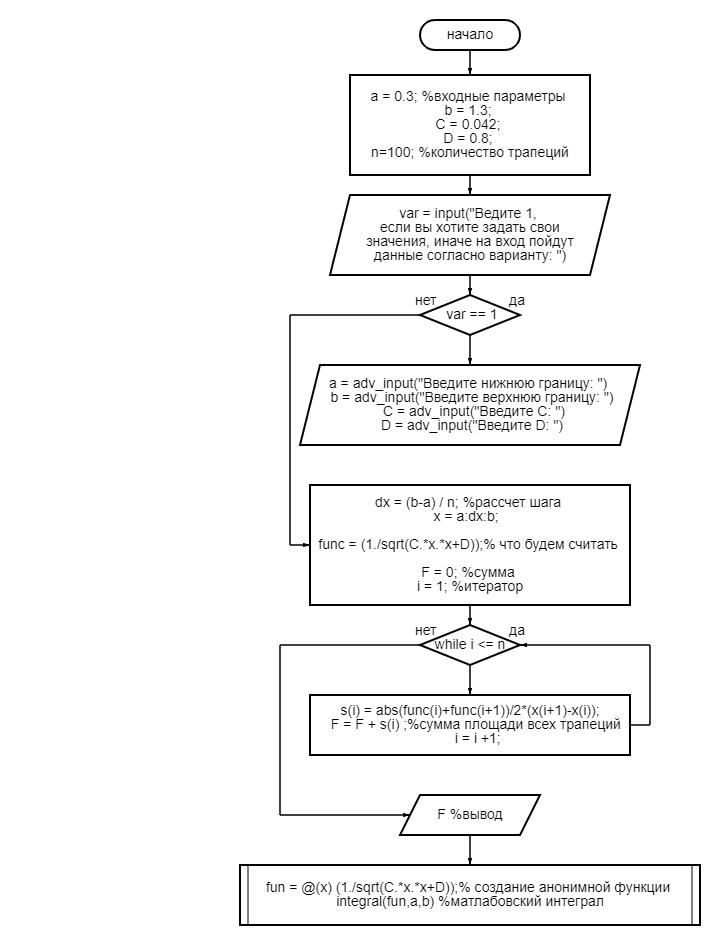
F %вывод

fun = @(x) (1./sqrt(C.\*x.\*x+D));% создание анонимной функции

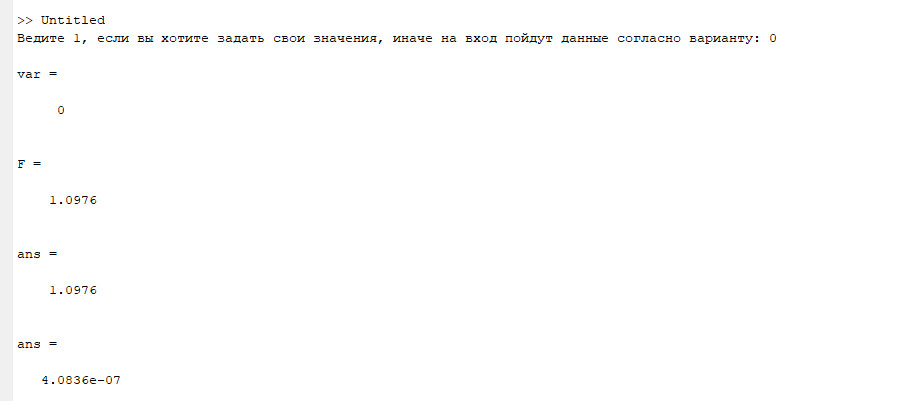
integral(fun,a,b) %матлабовский интеграл

acc(a, b,C,D, n)%Погрешность

***Блок-схема***

******

***Результат работы***



***Вывод***

Если сравнить полученные значения, они будут равны, что является доказательством того, что написанная мной функция работает исправно.

***Источники***

1. Метод Ньютона <https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%9D%D1%8C%D1%8E%D1%82%D0%BE%D0%BD%D0%B0>
2. Алгоритм решения методом Ньютона

<https://www.youtube.com/watch?v=OfRbIbeVYeg>

1. Лагранж

<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%B9_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D1%87%D0%BB%D0%B5%D0%BD_%D0%9B%D0%B0%D0%B3%D1%80%D0%B0%D0%BD%D0%B6%D0%B0>

1. Алгоритм <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%B9_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D1%87%D0%BB%D0%B5%D0%BD_%D0%9B%D0%B0%D0%B3%D1%80%D0%B0%D0%BD%D0%B6%D0%B0>
2. Онлайн калькулятор

<https://planetcalc.ru/8692/>

1. Калькулятор

<https://www.integral-calculator.ru/>

1. desmos

<https://www.desmos.com/calculator?lang=ru>

1. Метод трапеций

https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4\_%D1%82%D1%80%D0%B0%D0%BF%D0%B5%D1%86%D0%B8%D0%B9#:~:text=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%20%D1%82%D1%80%D0%B0%D0%BF%D0%B5%D1%86%D0%B8%D0%B9%20%E2%80%94%20%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%20%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F,%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%BE%D0%BC%20%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8%20%D0%B0%D0%BF%D0%BF%D1%80%D0%BE%D0%BA%D1%81%D0%B8%D0%BC%D0%B8%D1%80%D1%83%D0%B5%D1%82%D1%81%D1%8F%20%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D1%8B%D0%BC%D0%B8%20%D1%82%D1%80%D0%B0%D0%BF%D0%B5%D1%86%D0%B8%D1%8F%D0%BC%D0%B8.