МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Численное интегрирование функций

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к лабораторной работе №4

по дисциплине

Вычислительная Математика

РУКОВОДИТЕЛЬ:

Суркова Анна Сергеевна

____

(подпись)

СТУДЕНТ:

Цветков Николай Максимович

____

(подпись)

19-ИВТ-3

Работа защищена «___» ____________

С оценкой ________________________

Нижний Новгород 2021

Оглавление

Цель

Закрепление знаний и умений по численному интегрированию функций.

Постановка задачи

Вычислить интеграл по формулам центральных (средних) прямоугольников, трапеций и формуле Симпсона, при n=8 и n=20; оценить погрешность результата.

$$4.\ \int_{0.6}^{1.4} \frac{cosx}{x+1}dx$$

Теоретические сведения

Метод средних(центральных) прямоугольников

Пусть функция *y = f(x)* непрерывна на отрезке *[a; b]*. Нам требуется вычислить определенный интеграл $\int_a^b f(x)dx$.

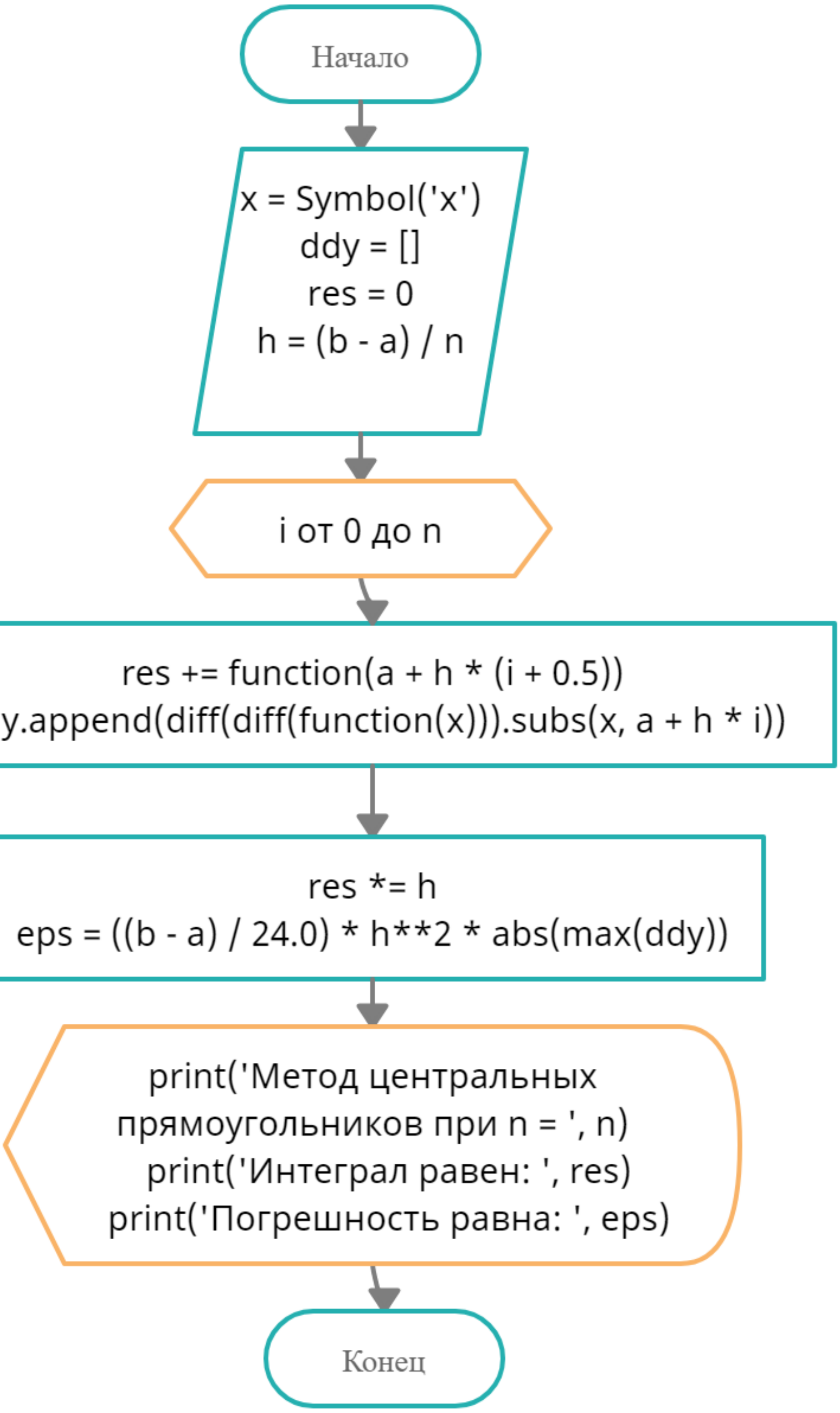
Обратимся к понятию определенного интеграла. Разобьем отрезок *[a;b]* на *n* частей $[x_{i-1}; x_i],\quad i=1,2,\ldots,n$ точками $a = x_0 < x_1 < x_2 < \ldots < x_{n-1} < x_n = b$. Внутри каждого отрезка $[x_{i-1}; x_i],\quad i=1,2,\ldots,n$ выберем точку ξ_i. Так как по определению определенный интеграл есть предел интегральных сумм при бесконечном уменьшении длины элементарного отрезка разбиения $\lambda = \max\limits_{i=1,2,\ldots,n}(x_i - x_{i-1}) \to 0$, то любая из интегральных сумм является приближенным значением интеграла $\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(\xi_i)\cdot(x_i - x_{i-1})$.

Суть метода прямоугольников заключается в том, что в качестве приближенного значения определенного интеграла берут интегральную сумму

Формула метода средних прямоугольников.

Если отрезок интегрирования *[a;b]* разбить на РАВНЫЕ части длины *h* точками $a = x_0, x_1 = x_0 + h, x_2 = x_0 + 2h, \ldots, x_{n-1} = x_0 + (n-1)h, x_n = x_0 + nh = b$ (то есть $h = x_i - x_{i-1} = \frac{b-a}{n},\quad i=1,2,\ldots,n$) и в качестве точек ξ_i выбрать СЕРЕДИНЫ элементарных отрезков $[x_{i-1}; x_i],\quad i=1,2,\ldots,n$ (то есть $\xi_i = x_{i-1} + \frac{h}{2},\quad i=1,2,\ldots,n$), то приближенное равенство $\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(\xi_i)\cdot(x_i - x_{i-1})$ можно записать в виде $\boxed{\int_a^b f(x)dx \approx h\cdot\sum_{i=1}^{n} f\left(x_{i-1} + \frac{h}{2}\right)}$. Это и есть формула метода прямоугольников. Ее еще называют формулой средних прямоугольников из-за способа выбора точек ξ_i.

$h = \dfrac{b-a}{n}$ называют **шагом разбиения отрезка** *[a;b]*.



Метод трапеций

Метод трапеций состоит в представлении определенного $\int\limits_a^b f(x)dx$ интеграла в виде суммы интегралов вида $\int\limits_{x_{i-1}}^{x_i} f(x)dx$ на каждом элементарном отрезке и в последующей приближенной замене

$$\int\limits_{x_{i-1}}^{x_i} f(x)dx \approx \frac{f(x_{i-1})+f(x_i)}{2}\cdot h$$

.

В силу пятого свойства определенного интеграла $\int\limits_a^b f(x)dx = \sum\limits_{i=1}^{n}\int\limits_{x_{i-1}}^{x_i} f(x)dx$.

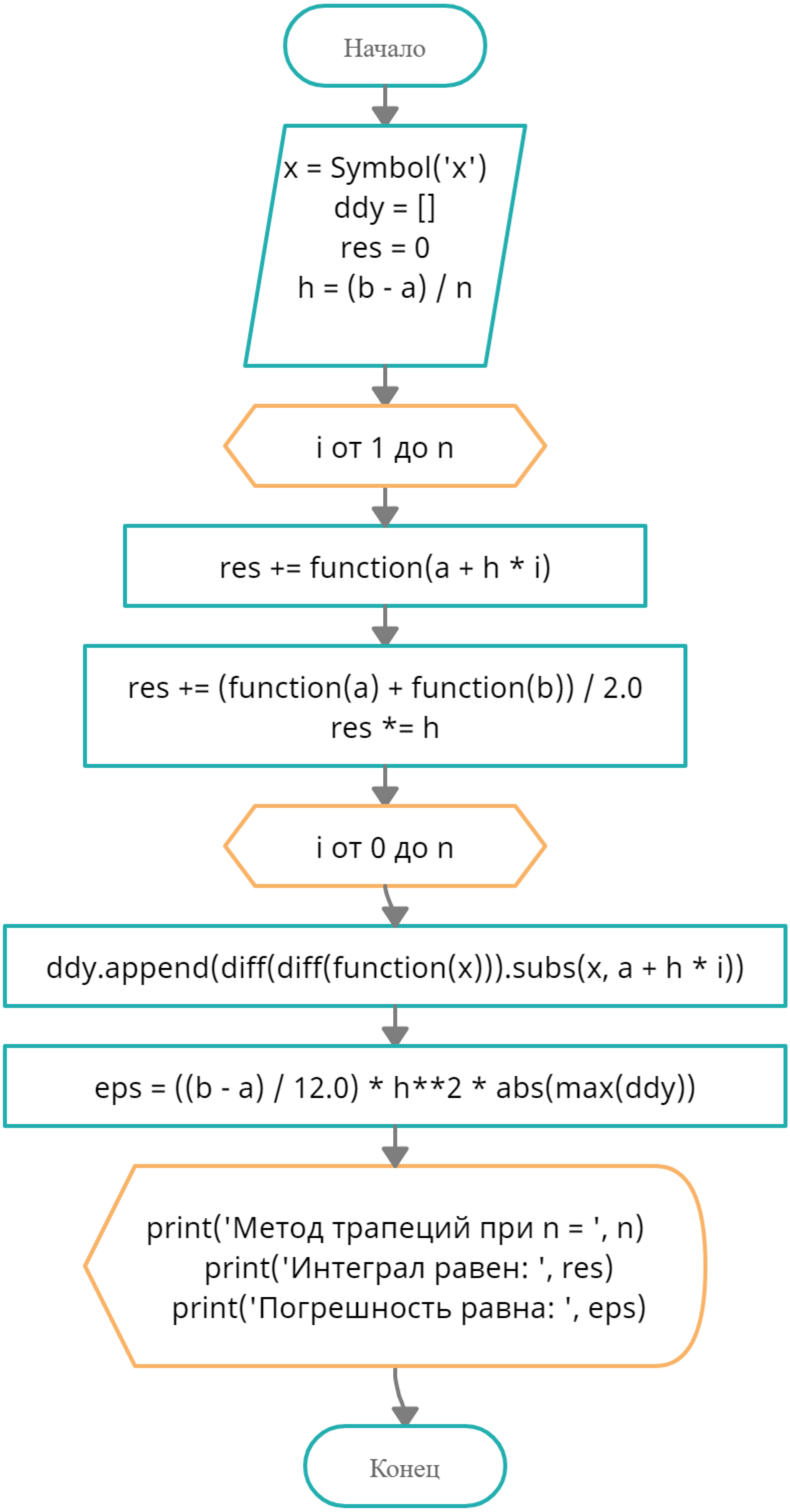
Если вместо интегралов $\int\limits_{x_{i-1}}^{x_i} f(x)dx$ подставить их приближенные значения, то получится **формула метода трапеций**:

$$\int\limits_a^b f(x)dx = \sum_{i=1}^{n}\int\limits_{x_{i-1}}^{x_i} f(x)dx \approx \sum_{i=1}^{n}\frac{f(x_{i-1})+f(x_i)}{2}\cdot h =$$

$$= \frac{h}{2}\cdot\left(f(x_0)+f(x_1)+f(x_1)+f(x_2)+f(x_2)+f(x_3)+\ldots+f(x_n)\right) =$$

$$= \frac{h}{2}\cdot\left(f(x_0)+2\sum_{i=1}^{n-1} f(x_i)+f(x_n)\right) \Rightarrow$$

$$\boxed{\int\limits_a^b f(x)dx \approx \frac{h}{2}\cdot\left(f(x_0)+2\sum_{i=1}^{n-1} f(x_i)+f(x_n)\right)}$$


Начало
x = Symbol('x')
ddy = []
res = 0
h = (b - a) / n
i от 1 до n
res += function(a + h * i)
res += (function(a) + function(b)) / 2.0
res *= h
i от 0 до n
ddy.append(diff(diff(function(x))).subs(x, a + h * i))
eps = ((b - a) / 12.0) * h**2 * abs(max(ddy))
print('Метод трапеций при n = ', n)
print('Интеграл равен: ', res)
print('Погрешность равна: ', eps)
Конец

Метод Симпсона

Этот метод более точный по сравнению с методами прямоугольников и трапеций.

В основе формулы Симпсона лежит квадратичная интерполяция подынтегральной функции на отрезке *[а,* 6] по трем равноотстоящим узлам.

Разобьем интервал интегрирования *[а, b]* на четное число *п* равных отрезков с шагом *h.*

Примем: $x_0 = a,\ x_1 = x_0 + h,\ \ldots,\ x_n = x_0 + nh = b.$

Значения функций в точках обозначим соответственно:

$$y_0 = f(a);\ y_1 = f(x_1);\ y_2 = f(x_2);\ \ldots;\ y_n = f(b).$$

На каждом отрезке [x_0, x_2], [x_2, x_4], ..., [x,_i, x,+ |] подынтегральную функцию Дх) заменим интерполяционным многочленом второй степени:

$$f(x) \approx P_i(x) = a_i x^2 + b_i x + c_i, \qquad (7.13)$$

где $x_{i-1} \le x \le x_{i+1}$

В качестве P,(x) можно принять интерполяционный многочлен Лагранжа второй степени, проходящий через концы каждых трех ординат: $y_0, y_1, y_2;\ y_2, y_3, y_4;\ y_4, y_5, y_6;\ \ldots;\ y_{n-2}, y_{n-1}, y_n.$

Формула Лагранжа для интервала [x,_i, ,t,+i] имеет вид:

$$P_i(x) = \frac{(x - x_i)(x - x_{i+1})}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})} \cdot y_{i-1} + \frac{(x - x_{i-1})(x - x_{i+1})}{(x_i - x_{i-1})(x_i - x_{i+1})} \cdot y_i + \\ + \frac{(x - x_{i-1})(x - x_i)}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)} \cdot y_{i+1}.$$

Элементарная площадь s, (см. рис. 7.7) может быть вычислена с помощью определенного интеграла. Учитывая, что x, - x,_i=x,+i - x, = Л, проведем вычисления и получим для каждого элементарного участка:

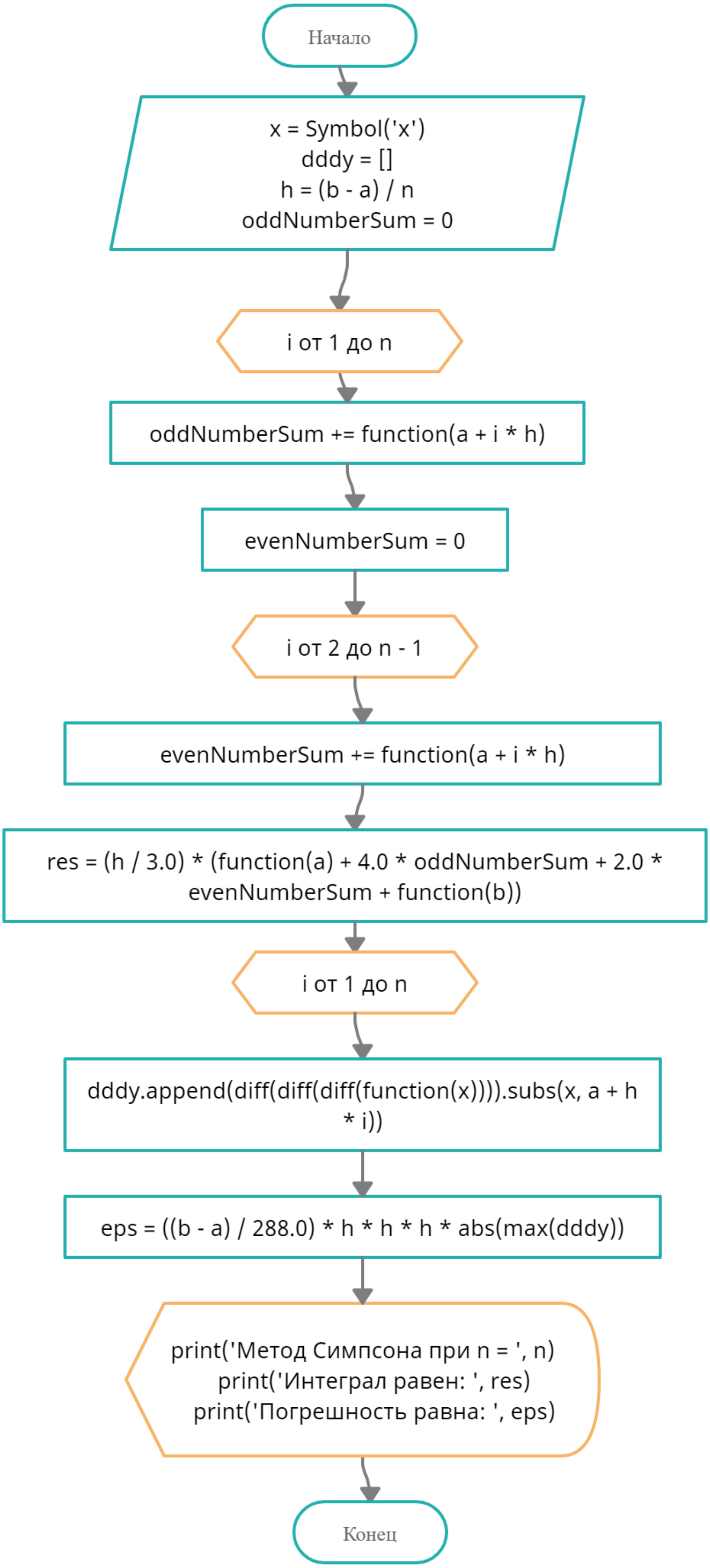
$$s_i = \int_{X_{i-1}}^{X_{i+1}} P_i(x)dx = \frac{h}{3} \cdot (y_{i-1} + 4y_i + y_{i+1}). \qquad (7.14)$$

После суммирования интегралов по всем отрезкам, получим составную формулу Симпсона:

$$\int_a^b f(x)dx \approx \frac{h}{3}\cdot\left[y_0 + 4(y_1 + y_3 + ... + y_{n-1}) + 2(y_2 + y_4 + ... + y_{n-2}) + y_n\right]. \quad (7.15)$$

Часто пользуются простой формулой Симпсона

$$\int_a^b f(x)dx = \frac{h}{3}\cdot\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right). \qquad (7.16)$$


Начало
x = Symbol('x')
dddy = []
h = (b - a) / n
oddNumberSum = 0
i от 1 до n
oddNumberSum += function(a + i * h)
evenNumberSum = 0
i от 2 до n - 1
evenNumberSum += function(a + i * h)
res = (h / 3.0) * (function(a) + 4.0 * oddNumberSum + 2.0 * evenNumberSum + function(b))
i от 1 до n
dddy.append(diff(diff(diff(function(x)))).subs(x, a + h * i))
eps = ((b - a) / 288.0) * h * h * h * abs(max(dddy))
print('Метод Симпсона при n = ', n)
print('Интеграл равен: ', res)
print('Погрешность равна: ', eps)
Конец

Расчетные данные

Исходная функция:

$$4.\ \int_{0.6}^{1.4} \frac{cosx}{x+1}dx$$

n = 8

	Значение интеграла	Значение погрешности
Метод средних прямоугольников	0.222167897115371	0.000109429359579646
Метод трапеций	0.222461800899640	0.000218858719159292
Метод Симпсона	0.222266020718306	1.05885814696600e-7

n = 20

	Значение интеграла	Значение погрешности
Метод средних прямоугольников	0.222250176423056	1.75086975327434e-5
Метод трапеций	0.222297211608253	3.50173950654868e-5
Метод Симпсона	0.222265858852776	7.14903395037209e-9

Листинг разработанной программы

Main.py

```
from solutionMethods import *


a = 0.6 # Наши границы интегрирования
b = 1.4
n1 = 8 # Наши границы разбиения
n2 = 20

centerRectangleMethod(a, b, n1)
trapeziumMethod(a, b, n1)
simpsonMethod(a, b, n1)
centerRectangleMethod(a, b, n2)
trapeziumMethod(a, b, n2)
simpsonMethod(a, b, n2)
```

solutionMethod.py

```
import math
from sympy import *

def function(x): # Наша исходная функция
    return cos(x) / (x + 1)

def centerRectangleMethod(a, b, n):
    x = Symbol('x')
    ddy = [] # Здесь мы храним значения второй производной в точках от a до b
    res = 0 # Переменная, в которой будет храниться наш результат
    h = (b - a) / n  # Шаг h
    for i in range(n): # Считаем значения второй производной и сумму значений фун
кции в промежутке
        res += function(a + h * (i + 0.5)) # Начиная с точки a, прибавляем в резу
льтату значение y(xi - 0.5), которое вычисляется как h * (i + 0.5)
        ddy.append(diff(diff(function(x))).subs(x, a + h * i)) # Заносим наши зна
чения второй производной от a до b с шагом h в список
    res *= h # Результат домножаем на h, так как шаг является раномерным
    eps = ((b - a) / 24.0) * h**2 * abs(max(ddy)) # Считаем погрешность по формул
е Rn = (b - a) / 24 * h^2 * max(y''(x))
    print('Метод центральных прямоугольников при n = ', n)
    print('Интеграл равен: ', res)
    print('Погрешность равна: ', eps)



def trapeziumMethod(a, b, n):
    x = Symbol('x')
    ddy = [] # Здесь мы храним значения второй производной в точках от a до b
    res = 0 # Переменная, в которой будет храниться наш результат
    h = (b - a) / n  # Шаг h
```

```python
    for i in range(1, n): # Считаем значения второй производной и сумму значений функции в промежутке
        res += function(a + h * i) # Начиная с точки a, прибавляем в результату значение y(xi), которое вычисляется как h * i
    res += (function(a) + function(b)) / 2.0 # К результату добавим (y0 + yn) / 2
    res *= h # Результат домножаем на h, так как шаг является раномерным
    for i in range(n):
        ddy.append(diff(diff(function(x))).subs(x, a + h * i)) # Заносим наши значения второй производной от a до b с шагом h в список
    eps = ((b - a) / 12.0) * h**2 * abs(max(ddy)) # Считаем погрешность по формуле Rn = (b - a) / 12 * h^2 * max(y''(x))
    print('Метод трапеций при n = ', n)
    print('Интеграл равен: ', res)
    print('Погрешность равна: ', eps)

def simpsonMethod(a, b, n):
    x = Symbol('x')
    dddy = [] # Здесь мы храним значения третьей производной в точках от a до b
    h = (b - a) / n  # Шаг h
    oddNumberSum = 0 # Здесь мы храним сумму нечетных значений
    for i in range(1, n, 2): # Считаем сумму y1 + y3 + ... + yn - 1
        oddNumberSum += function(a + i * h)
    evenNumberSum = 0 # Здесь мы храним сумму четных значений
    for i in range(2, n - 1, 2): # Считаем сумму y2 + y4 + ... + yn - 2
        evenNumberSum += function(a + i * h)
    res = (h / 3.0) * (function(a) + 4.0 * oddNumberSum + 2.0 * evenNumberSum + function(b)) # Вычисляем интеграл = h / 3(y0 + 4 * (y1 + y3 + ... + yn-1) + 2 * (y2 + y4 + ... + yn - 2) + yn)
    for i in range(n):
        dddy.append(diff(diff(diff(function(x)))).subs(x, a + h * i)) # Заносим наши значения третьей производной от a до b с шагом h в список
    eps = ((b - a) / 288.0) * h * h * h * abs(max(dddy)) # Считаем погрешность по формуле Rn = (b - a) / 288 * h^3 * max(y'''(x))
    print('Метод Симпсона при n = ', n)
    print('Интеграл равен: ', res)
    print('Погрешность равна: ', eps)
```

Результаты работы программы

```
Метод цетральных прямоугольников при n =  8
Интеграл равен:  0.222167897115371
Погрешность равна:  0.000109429359579646
Метод трапеций при n =  8
Интеграл равен:  0.222461800899640
Погрешность равна:  0.000218858719159292
Метод Симпсона при n =  8
Интеграл равен:  0.222266020718306
Погрешность равна:  1.05885814696600e-7
Метод цетральных прямоугольников при n =  20
Интеграл равен:  0.222250176423056
Погрешность равна:  1.75086975327434e-5
Метод трапеций при n =  20
Интеграл равен:  0.222297211608253
Погрешность равна:  3.50173950654868e-5
Метод Симпсона при n =  20
Интеграл равен:  0.222265858852776
Погрешность равна:  7.14903395037209e-9
```

Вывод

В ходе данной работы были закреплены знания и умения по вычислению интеграла при помощи методов средних прямоугольников, трапеция и методу Симпсона.

Самым точным является метод Симпсона, а самым не точным метод трапеций. При увеличении кол-во промежутков разбиения n точность результатов возрастает.