КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

# Семестровая работа по предмету: «Вычислительные методы» на тему: « Вычисление интеграла с помощью квадратурных формул»

Вариант №6

Работу выполнил:
Студент 3-го курса
Группы 09-409
Набиев М.А.

Работу проверила:
профессор КВМ Павлова М.Ф.
11 мая 2017 г.

Казань-2017

1. Постановка задачи: на стеке узлов

$\{x_i\}_{i=1}^{n}$ , $где$ $x_i=a+ih$, $h=\frac{b-a}{n}$, $где$ $a=0$, $b=1.5$, $n=10$ построить таблицу приближенных значений

$$C(x)=\int_0^x \cos\left(\frac{\pi t^2}{2}\right)dt \quad (1)$$

Цель задания: изучить и сравнить эффективность различных квадратурных формул для приближенного вычисления интеграла, используя составные квадратурные формулы:

- Левых прямоугольников,
- Трапеции,
- Симпсона

Вычисление квадратурных формул проводится для 11 точек x на отрезке от 0 до 1.5 с шагом $h=\frac{(b-a)}{10}$ . Для каждой точки вычисления проводятся на отрезке от 0 до x, где t меняется с шагом $H_N=\frac{x}{N}$ . Интеграл вычисляется с точностью $\varepsilon=10^{-4}$ . Точность вычисления определяется сравнением результатов вычисления при различном числе разбиения отрезка интегрирования. Точность считается достигнутой, если выполняется следующее неравенство: $|C_N-C_{2N}|<\varepsilon$ , где $C_N$ и $C_{2N}$ - приближенные значения интеграла, вычисленные с помощью составной квадратурной формулы при разбиении отрезка на N и 2N частей соответственно.

2.1 Квадратурная формула левых прямоугольников.

Квадратурная формула левых прямоугольников для определенного интеграла имеет следующий вид:

$J=\int_a^b f(x)dx=(b-a)f(a)+R(f)$ , где погрешность R(f) вычисляется по формуле:

$$R(f)\le-\frac{f'(\xi)(b-a)^2}{2},\ \xi\subseteq[a,b]$$

Составная квадратурная формула левых прямоугольников имеет следующий вид:

$J_N=\int_a^b f(x)dx=\sum_{i=0}^{N-1}\int_{x_i}^{x_{i+1}} f(x)dx\approx\sum h*f(x_i)=S_N$ , где погрешность составной квадратурной формулы вычисляется следующим образом:

$$R_N(f)\le\sum_{i=0}^{N-1}\frac{h^2}{2}f'(\xi_i),\quad где\quad h=\frac{b-a}{n},\ \xi_i\subseteq(x_i,\ x_i+1),\ i=0,..,N-1$$

В результате были получены следующие данные, где N – количество разбиений, необходимых для достижения заданной точности:

| x | C(x) | N |
|---|---|---|
| 0 | 0 | 1 |
| 0.15 | 0.1499970724 | 2 |
| 0.3 | 0.2999063256 | 2 |
| 0.45 | 0.4455559528 | 128 |
| 0.6 | 0.5811865455 | 512 |
| 0.75 | 0.693592915 | 2048 |
| 0.9 | 0.7649005694 | 4096 |
| 1.05 | 0.7759838779 | 8192 |
| 1.2 | 0.7154976859 | 16384 |
| 1.35 | 0.5923475549 | 16384 |
| 1.5 | 0.4453492455 | 16384 |

таблица 1. Значения интеграла, вычисленные методом левых прямоугольников

2.2 Квадратурная формула трапеций.

Квадратурная формула трапеций имеет следующий вид:

$J=\int_a^b f(x)=(b-a)\frac{f(a)+f(b)}{2}+R(f)$ , где погрешность квадратурной формулы R(f) составляет:

$R(f)\le-\frac{f''(\xi)}{12}(b-a)^{3,}\ \ \xi\subseteq[a,b]$

Составная квадратурная формула трапеций вычисляется следующим образом

$J_N=\int_a^b f(x)\,dx\approx\sum_{i=0}^{N-1}\frac{f(x_i)+f(x_{i+1})}{2}h$

где погрешность квадратурной формулы составляет:

$R_N(f)\le\sum_{i=0}^{N=1}\frac{h^3}{12}f''(\xi_i),\ \ где\ \ h=\frac{b-a}{n},\ \ \xi_i\subseteq(x_i,\ \ x_i+1),\ \ i=0,..,N-1$

Результаты вычислений, где N – количество разбиений, необходимых для достижения заданной точности:

| x | C(x) | N |
|---|---|---|
| 0 | 0 | 1 |
| 0.15 | 0.1499736538 | 2 |
| 0.3 | 0.2993854378 | 8 |
| 0.45 | 0.445439094 | 16 |
| 0.6 | 0.5810658588 | 32 |
| 0.75 | 0.693505147 | 64 |
| 0.9 | 0.7648118875 | 128 |
| 1.05 | 0.7758912629 | 128 |
| 1.2 | 0.7154164471 | 128 |
| 1.35 | 0.5922559281 | 128 |
| 1.5 | 0.4452818138 | 128 |

таблица 2. Значения интеграла, вычисленные методом трапеций

**2.3 Квадратурная формула Симпсона**

Квадратурная формула Симпсона для определенного интеграла имеет следующий вид:

$J=\int_a^b f(x)\,dx=\frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)+R(f)$ , где погрешность квадратурной формулы R(f) составляет

$$R(f) \le \frac{-(b-a)^5}{2880} f^{(4)}(\xi), \ \xi \subseteq [a,b]$$

Составная формула Симпсона вычисляется следующим образом:

$$J_N = \int_a^b f(x)\,dx \approx \sum_{i=0}^{N-1} \frac{h}{6}\left( f(x_i) + 4f\left(\frac{x_i + x_{i+1}}{2}\right) + f(x_{i+1})\right)$$

Где погрешность квадратурной формулы вычисляется следующим образом:

$$R(f) \le \sum_{i=0}^{N-1} \frac{h^4}{2880} f^{(4)}(\xi_i) \quad \text{где} \quad h = \frac{b-a}{n}, \ \xi_i \subseteq (x_i, \ x_i + 1), \ i = 0,..,N-1$$

Результаты вычислений, где N – количество разбиений, необходимых для достижения заданной точности:

| x | C(x) | N |
|---|---|---|
| 0 | 0 | 1 |
| 0.15 | 0.1499812155 | 2 |
| 0.3 | 0.2993994458 | 2 |
| 0.45 | 0.4454675709 | 4 |
| 0.6 | 0.5810934041 | 4 |
| 0.75 | 0.6935240681 | 4 |
| 0.9 | 0.7648237778 | 8 |
| 1.05 | 0.7759136894 | 8 |
| 1.2 | 0.7154384276 | 16 |
| 1.35 | 0.5922678042 | 16 |
| 1.5 | 0.445262747 | 16 |

таблица 3. Значения интеграла, вычисленные методом Симпсона

**Вывод**

По результатам таблиц 1, 2, 3 можно заметить, что для достижения заданной точности для разных методов необходимо разное количество разбиений, при чем точнее формула, тем меньше разбиений требуется.

**Приложение**

Листинг программы

```
import math
import numpy as np
from matplotlib import pyplot as plt

eps = 1e-06
int_eps = 1e-4

def f(x, eps):
  sum = x
  a = x
  n = 0
  while (math.fabs(a) > eps):
    q = -(((math.pi / 2) ** 2) * (x ** 4) * (4 * n + 1)) / ((2 * n + 1) * (2 *
n + 2) * (4 * n + 5))
    a *= q
    sum += a
    n += 1
  return sum

def c(x):
  return math.cos(math.pi*x*x/2)


def integral_left_rect(a, b, x):
  N = 1
  s = 0
  s_prev = 0
  while True:
    s_prev = s
    s=0
    h = float(b - a) / N
    for i in range(N):
      s += c( a + i * h)*h
    #s *= h
    if (math.fabs(s - s_prev) < int_eps):
      break
    else:
      N *=2
  return s, N

def integral_trapezoid(a, b, x):
  N = 1
  s = 0
  s_prev = 0
  while True:
    s_prev = s
    s=0
    h = float(b - a) / N
    for i in range(N):
      s += (c(a + i * h) + c(a + (i + 1) * h))
    s *= (h / 2.)
```

```
    if (math.fabs(s - s_prev) < int_eps):
      break
    else:
      N *=2
  return s, N

def integral_simpson(a, b, x):
  N = 1
  s = 0
  s_prev = 0
  while True:
    s_prev = s
    s=0
    h = float(b - a) / N
    for i in range(N):
      s += (c(a + i * h) + 4*c(a + (i + 0.5) * h) + c(a + (i + 1) * h))
    s *= (h / 6.)
    if (math.fabs(s - s_prev) < int_eps):
      break
    else:
      N *=2
  return s, N


def main():
  a = 0
  b = 1.5
  n=10
  h=float(b-a)/n
  for i in range(n+1):
    x=a+i*h
    left=integral_simpson(0,x,x)
    trap=integral_trapezoid(0,x,x)
    simp=integral_simpson(0,x,x)
    f_r=f(x, eps)
    print 'Left rect: ', left
    print 'Trapezoid: ', trap
    print 'Simpson:   ',simp
    print 'real val:  ',f_r

if __name__ == "__main__":
  main()
```