МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе №7

по дисциплине «Вычислительная математика»

Тема: Составные формулы прямоугольников, трапеций, Симпсона

Студент гр. 8383 _____ Ларин А.

Преподаватель _____ Сучков А.И.

Санкт-Петербург

2019

Цель работы.

Изучение и сравнение различных методов численного интегрирования на примере составных формул прямоугольников, трапеций, Симпсона.

Основные теоретические положения.

Пусть требуется найти определенный интеграл $I = \int_a^b f(x)dx$, где функция $f(x)$ непрерывна на отрезке $[a, b]$. Для приближенного вычисления интегралов чаще всего подынтегральную функцию заменяют «близкой» ей вспомогательной функцией, интеграл от которой вычисляется аналитически. За приближенное значение интеграла принимают значение интеграла от вспомогательной функции. Заменим функцию на отрезке $[a, b]$ её значением в середине отрезка. Искомый интеграл, равный площади криволинейной фигуры, заменяется на площадь прямоугольника. Из геометрических соображений нетрудно записать формулу прямоугольников:

$$\int_a^b f(x)dx \approx f\left(\frac{a+b}{2}\right)(b-a).$$

Приблизив $f(x)$ линейной функцией и вычислив площадь соответствующей трапеции, получим формулу трапеций:

$$\int_a^b f(x)dx \approx \frac{1}{2}\big(f(a)+f(b)\big)(b-a).$$

Если же приблизить подынтегральную функцию параболой, проходящей через точки $\big(a, f(a)\big), \left(\frac{a+b}{2}, f\left(\frac{a+b}{2}\right)\right), \big(b, f(b)\big)$, то получим формулу Симпсона (парабол):

$$\int_a^b f(x)dx \approx \frac{1}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)(b-a).$$

Все три формулы хорошо иллюстрируются геометрически и представлены на рис. 1.

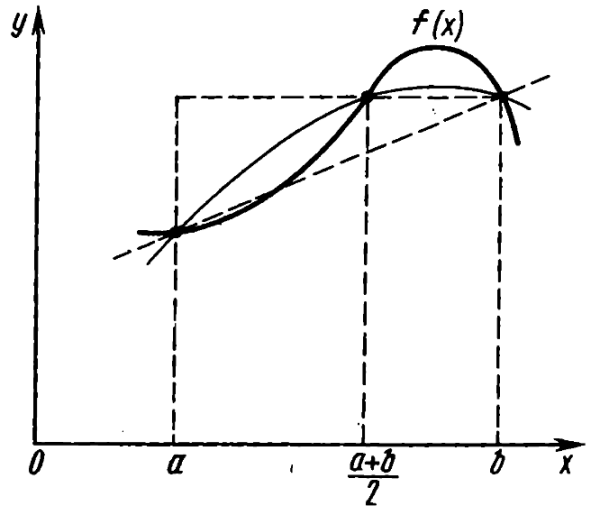

Рисунок 1 – Геометрическая интерпретация формул прямоугольников, трапеций и Симпсона

Для повышения точности интегрирования применяют составные формулы. Для этого разбивают отрезок $[a, b]$ на четное $n = 2m$ число отрезков длины $h = \frac{b-a}{n}$ и на каждом из отрезков длины $2h$ применяют соответствующую формулу. Таким образом получают составные формулы прямоугольников, трапеций и Симпсона.

На сетке $x_i = a + ih,\ y_i = f(x_i), i = \overline{0,2m}$ составные формулы имеют следующий вид:

- формула прямоугольников:

$$\int_a^b f(x)dx = h \sum_{i=0}^{n-1} f\left(x_i + \frac{h}{2}\right) + R_1$$

$R_1 \approx \frac{h^2}{24}\int_a^b f''(x)dx = o(h^2)$;

- формула трапеций:

$$\int_a^b f(x)dx = \frac{h}{2}\sum_{i=0}^{n-1}\left(f(x_i) + f(x_{i+1})\right) + R_2$$

$R_2 \approx -\frac{h^2}{12}\int_a^b f''(x)dx = o(h^2)$;

- формула Симпсона:

$$\int_a^b f(x)dx = \frac{h}{3}\sum_{i=0}^{m-1}\left(f(x_{2i}) + 4f(x_{2i+1}) + f(x_{2i+2})\right) + R_3$$

$$R_3 \approx -\frac{h^4}{180}\int_a^b f^{IV}(x)dx = o(h^4),$$

где R_1, R_2, R_3 – остаточные члены. Оценки остаточных членов получены в предположении, что соответствующие производные $f(x)$ непрерывны на $[a, b]$. При $n \to \infty$ приближенные значения интегралов для всех трех формул (в предположении отсутствия погрешностей округления) стремятся к точному значению интеграла. Любая попытка сравнить достоинства приведенных формул связана с вопросом типа «что больше $h^2 f''(x)$ или $h^4 f''''(x)$?» Ответ зависит от свойств интегрируемой функции. Можно лишь утверждать, что остаточный член формулы прямоугольников примерно вдвое меньше, чем формулы трапеций, и оба они имеют порядок h^2. А остаточный член формулы Симпсона убывает быстрее – со скоростью h^4.

Для практической оценки погрешности квадратурной можно использовать правило Рунге. Для этого проводят вычисления на сетках с шагом h и $h/2$, получают приближенные значения интеграла I_h и $I_{h/2}$ за окончательные значения интеграла принимают величины:

- $I_{h/2} + \frac{I_{h/2} - I_h}{3}$ – для формулы прямоугольников;
- $I_{h/2} - \frac{I_{h/2} - I_h}{3}$ – для формулы трапеций;
- $I_{h/2} - \frac{I_{h/2} - I_h}{15}$ – для формулы Симпсона.

При этом за погрешность приближённого значения интеграла принимается величина $\frac{|I_{h/2} - I_h|}{2^k - 1}$, причём для формул прямоугольников и трапеций $k = 2$, для формулы Симпсона $k = 4$. Такую оценку погрешностей применяют обычно для построения адаптивных алгоритмов, т.е. таких алгоритмов, которые автоматически так определяют величину шага h, что результат удовлетворяет требуемой точности.

Постановка задачи.

Используя квадратурные формулы прямоугольников, трапеций и Симпсона, вычислить значения заданного интеграла и, применив правило Рунге,

найти наименьшее значение n (наибольшее значение шага h), при котором каждая из указанных формул дает приближенное значение интеграла с погрешностью ε, не превышающей заданную. Порядок выполнения раобты следующий:

- Составить подпрограмму-функцию `F` для вычисления подынтегральной функции.
- Составить подпрограммы-функции `RECT`, `TRAP`, `SIMPS` для вычисления интегралов по формулам прямоугольников, трапеций и Симпсона соответственно.
- Составить головную программу, содержащую оценку по Рунге погрешности каждой из перечисленных ранее квадратурных формул, удваивающих n до тех пор, пока погрешность не станет меньше ε, и осуществляющих печать результатов значения интеграла и значения n для каждой формулы.
- Провести вычисления по программе, добиваясь, чтобы результат удовлетворял требуемой точности.

Выполнение работы.

Вычислим интеграл:

$$\int_{\pi/2}^{\pi} \sqrt{x} e^{-x^2} dx$$

Для нахождения значения определенного интеграла численными методами была составлена функция, вычисляющая подынтегральное выражение, а также три функции, вычисляющие интеграл методом численного интегрирования. График подынтегральной функции $f(x)$ с отмеченными пределами интегрирования представлен на рис. 2.

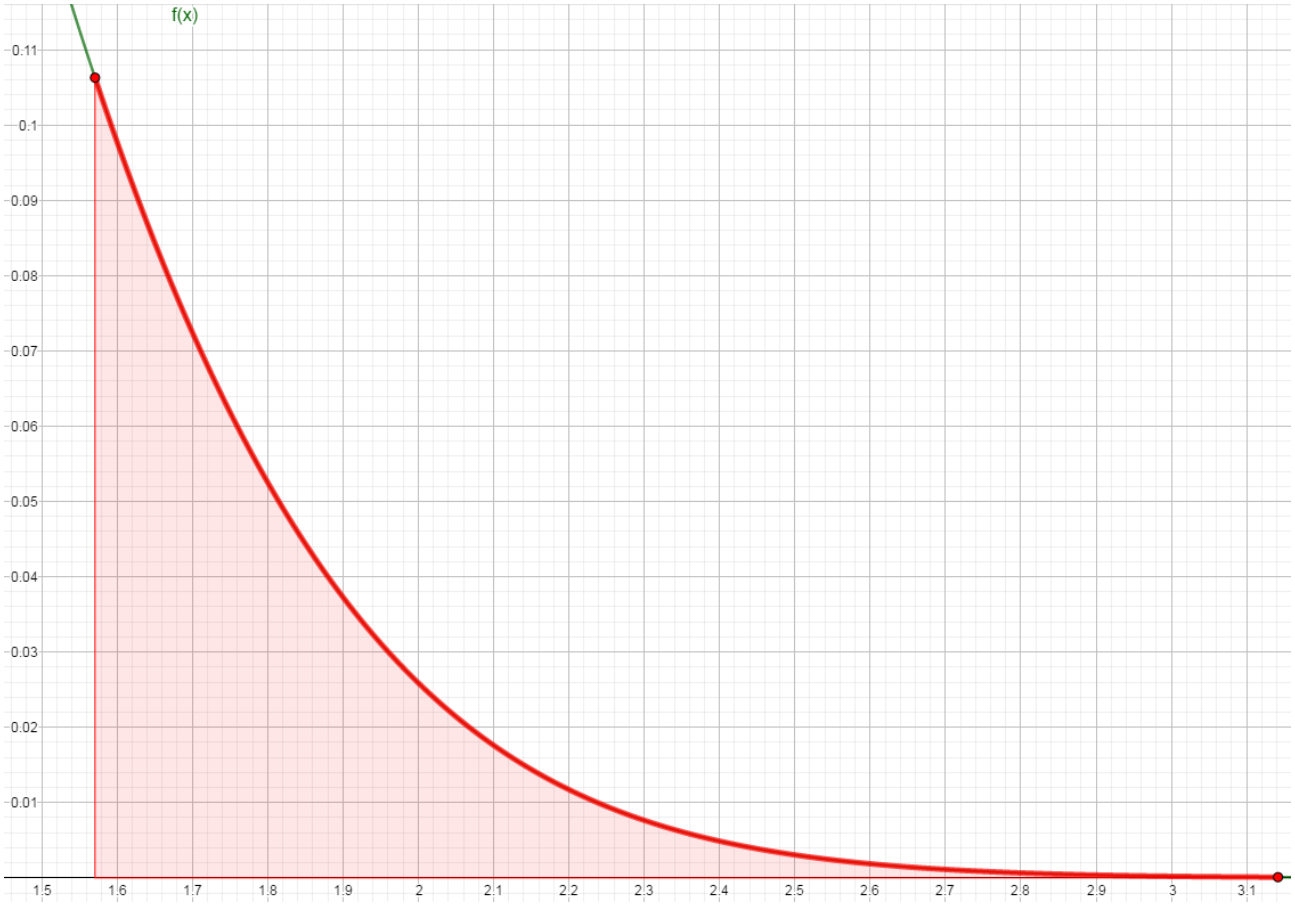

Рисунок 2 – График подынтегральной функции к отмеченными пределами интегрирования

Головная функция и функции для вычисления интеграла методом прямоугольников, трапеций и парабол предоставлены в приложении А. На вход каждая функция принимает следующие параметры: `a,b` – предел интегрирования $[a, b]$, `eps` – требуемая точность вычисления корня, `f(x)` – интегрируемая функция. Функции возвращают вычисленное значение определенного интеграла и количество отрезков, на которые понадобилось разбить интервал для достижения заданной точности. Вычисление количество отрезков интегрирования функции производится при помощи правила Рунге. Для этого в функциях производятся вычисления интегралов I_h и $I_{h/2}$, вычисляется погрешность по формуле $\frac{|I_{h/2}-I_h|}{2^k-1}$, где для формул прямоугольников и трапеций $k = 2$, для формулы Симпсона $k = 4$, после чего данная величина сравнивается с требуемым значением eps, и если превосходит его, то количество отрезков

разбиения n удваивается и происходит повторная оценка. В табл. 1 приведены расчеты значения интеграла s_* и количества отрезков n_*, необходимых для его расчета с варьируемой точностью `eps,` методом '*', где '*' принимает значения: R – метод прямоугольников, T – метод трапеций, S – метод Симпсона(парабол).

Таблица 1 – Вычисление значения интеграла различными методами с варьированием значения *eps*

eps	s_R	s_T	s_S	n_R	n_T	n_S
0.100000	0.028848	0.058819	0.031816	2	2	2
0.010000	0.028848	0.038961	0.031816	2	4	2
0.001000	0.031376	0.033299	0.031643	8	8	4
0.000100	0.031380	0.031500	0.031389	32	32	8
0.000010	0.031380	0.031387	0.031389	64	128	8
0.000001	0.031380	0.031380	0.031380	256	256	16

Выводы.

Проанализировав результаты расчетов, мы можем сделать вывод, что число итераций, необходимых для вычисления интеграла по составным формулам прямоугольника, трапеции и Симпсона возрастает с ростом требуемой точности результата.

По результатам оценки необходимого количества отрезков разбиения по правилу Рунге можем сделать вывод, что для данной функции наименьшее количество итераций для получения необходимой точности результата требуется при вычислении интеграла по формуле Симпсона(парабол), далее в порядке увеличения числа итераций идет формула прямоугольников, а затем формула трапеций. Следовательно, в данном случае метод парабол дает наилучшие результаты, так как дает наименьшую погрешность из исследованных методов.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define CSQR(CX) ((CX)*(CX))

#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif // !M_PI

#define A (M_PI/2)
#define B M_PI
#define E 0.000000000000001


double F(double x) {
	if (x < A)printf("Boo<%lf",x);
	if(x>B)printf("Foo>%lf",x);
	return sqrt(x)*exp(-CSQR(x));
}

double _RECT(double h, int n, double x0, double(*f)(double)) {
	double x = x0;
	double s = 0;
	for (int i = 0; i < n; i++) {
		s += f(x + h / 2);
		x += h;
	}
	s *= h;
	return s;
}

double RECT(double a, double b, double e, double(*f)(double), int &n) {
#define H ((b - a) / n)
	double h= NAN, h2=NAN, Ih=NAN, Ih2= NAN;
	int k = 2;
	double e0 = e * ((1 << k) - 1);
	double d = e0+1;
	n = 1;

	Ih2 = _RECT(H, n, a, f);
	while(d>=e0){
```

```
            n *= 2;
            Ih = Ih2;
            Ih2 = _RECT(H, n, a, f);
            d = fabsl(Ih2 - Ih);
      }
      return (Ih2 + (Ih2-Ih)/3);
#undef H
}

double _TRAP(double h, int n, double x0, double(*f)(double)) {
#define X(CI) (x0+(CI)*h)
      //double x = x0;
      double s = 0;
      for (int i = 0; i < n; i++) {
            s += f(X(i))+f(X(i+1));
            //x += h;
      }
      s *= h/2;
      return s;
}

double TRAP(double a, double b, double e, double(*f)(double), int &n) {
#define H ((b - a) / n)
      double h = NAN, h2 = NAN, Ih = NAN, Ih2 = NAN;
      int k = 2;
      double e0 = e * ((1 << k) - 1);
      double d = e0 + 1;
      n = 1;

      Ih2 = _TRAP(H, n, a, f);
      while (d >= e0) {
            n *= 2;
            Ih = Ih2;
            Ih2 = _TRAP(H, n, a, f);
            d = fabsl(Ih2 - Ih);
      }
      return (Ih2 - (Ih2 - Ih) / 3);
#undef H
}

double _SIMPS(double h, int n, double x0, double(*f)(double)) {
//#define X(CI) (x0+(CI)*h)
      int m = n / 2;
      double s = 0;
      for (int i = 0; i < m; i++) {
```

```
            s += f(X(2 * i)) + 4 * f(X(2 * i + 1)) + f(X(2 * i + 2));
      }
      s *= h / 3;
      return s;
}

double SIMPS(double a, double b, double e, double(*f)(double), int &n,
bool echo = false) {
#define H ((b - a) / n)
      double h = NAN, h2 = NAN, Ih = NAN, Ih2 = NAN;
      int k = 4;
      double e0 = e * ((1 << k) - 1);
      double d = e0 + 1;
      n = 1;

      Ih2 = _SIMPS(H, n, a, f);
      while (d >= e0) {
            n *= 2;
            Ih = Ih2;
            Ih2 = _SIMPS(H, n, a, f);
            d = fabsl(Ih2 - Ih);
      }
      return (Ih2 - (Ih2 - Ih) / 15);
#undef H
}




int main(){
      double s_R, s_T, s_S;
      int k_R, k_T, k_S;
      printf("Eps\t\ts_R\t\ts_T\t\ts_S\t\tk_R\tk_T\tk_S\n");
      for (double eps = 0.1; eps > 0.000001; eps /= 10) {
            s_R = RECT(A, B, eps, F, k_R);
            //s_T = TRAP_(eps, k_T);
            s_T = TRAP(A, B, eps, F, k_T);
            s_S = SIMPS(A, B, eps, F, k_S);
            printf("%lf\t%lf\t%lf\t%lf\t%d\t%d\t%d\n", eps, s_R, s_T, s_S,
k_R, k_T, k_S);
      }
      system("pause");
      return 0;
}
```