**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. 8381 |  | Киреев К.А. |
| Преподаватель |  | Щеголева Н.Л. |

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

, (1)

где функция  непрерывна на отрезке .

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  её значением в середине отрезка. Искомый интеграл, равный площади криволинейной фигуры, заменяется на площадь прямоугольника. Из геометрических соображений нетрудно записать *формулу прямоугольников*:

 (2)

Приблизив  линейной функцией и вычислив площадь соответствующей трапеции, получим формулу трапеций:

 (3)

Если же приблизить подынтегральную функцию параболой, проходящей через точки , то получим формулу Симпсона (парабол)

 (4)

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

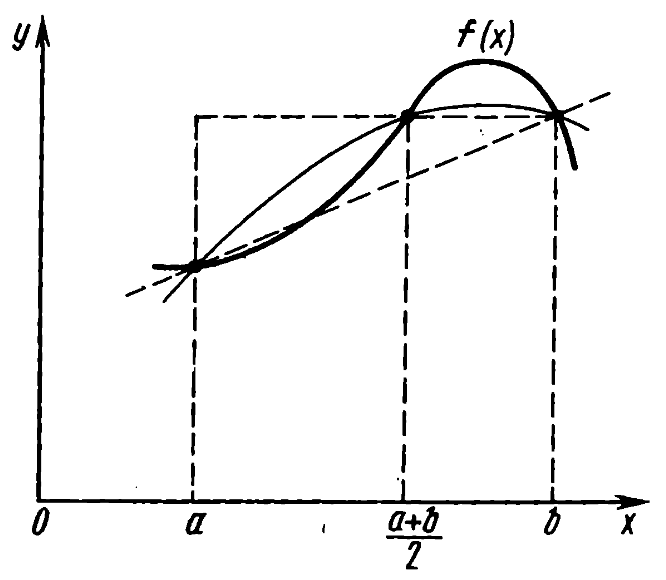


Рисунок 1 – Формулы прямоугольников, трапеций и Симпсона

Для повышения точности интегрирования применяют составные формулы. Для этого разбивают отрезок  на четное число отрезков длины  и на каждом из отрезков длины 2h применяют соответствующую формулу. Таким образом получают *составные формулы прямоугольников, трапеций и Симпсона*.

На сетке  составные формулы имеют следующий вид:

* Формула прямоугольников:

 (5)

* Формула трапеций:

 (6)

* Формула Симпсона:

 (7)

где R1, R2, R3 – остаточные члены. Оценки остаточных членов получены в предположении, что соответствующие производные  непрерывны на При n→∞ 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 или ?» 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 h2. А остаточный член формулы Симпсона убывает быстрее – со скоростью h4.

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 Ih и Ih/2 за окончательные значения интеграла принимают величины:

*  - для формулы прямоугольников (8)
*  - для формулы трапеций (9)
*  - для формулы Симпсона (10)

При этом за погрешность приближенного значения интеграла принимается величина , причем 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.**

Найти наименьшее значение , при котором каждая из указанных квадратурных формул даёт приближённое значение интеграла с погрешностью , не превышающей заданную.

Поиск осуществить для погрешностей 0.01, 0.001, 0.0001

**Выполнение работы.**

1. Построен график функции

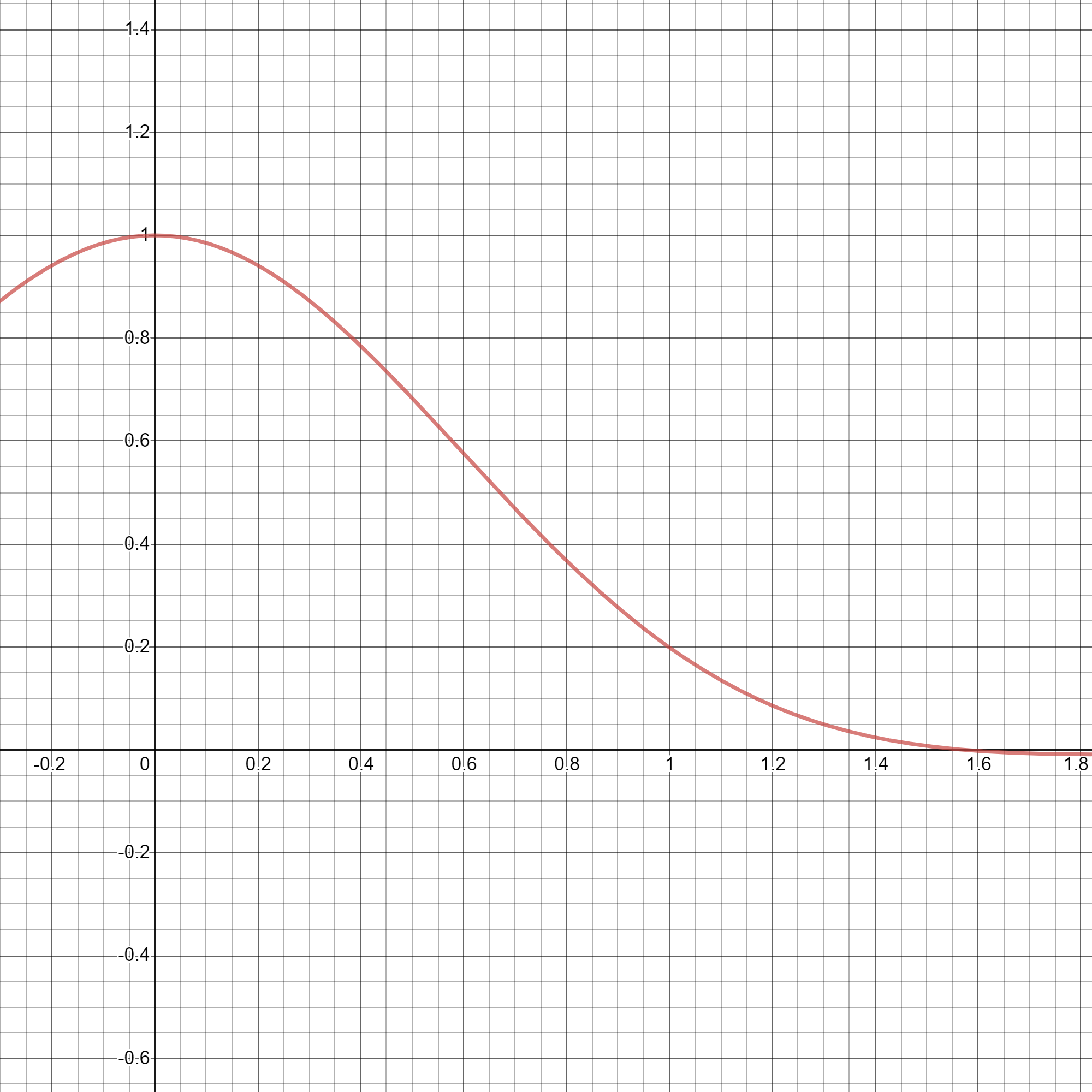


Рисунок 2 – График функции

1. Составлена формула для вычисления подынтегральной функции
2. Составлены подпрограммы-функции для вычисления интегралов по формулам прямоугольников, трапеций и Симпсона.
3. Составлена головная программа, содержащая 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.
4. Посчитаны значения интегралов по первому приближению:

Таблица 1 – Первое приближение

|  |  |  |
| --- | --- | --- |
| RECT | TRAP | SIMPS |
| 0.683462 | 0.599383 | 0.655436 |

1. Посчитаны значения интегралов по второму приближению (при h = 0.02):

Таблица 2 – Второе приближение

|  |  |  |
| --- | --- | --- |
| RECT | TRAP | SIMPS |
| 0.66419 | 0.687206 | 0.701764 |

1. Сделаны расчёты при разных значениях  и составлена таблица с подсчитанными значениями интегралов на отрезке :

Таблица 3 – Формула прямоугольников

|  |  |  |  |
| --- | --- | --- | --- |
|  | Значение | n | h |
| 0.01 | 0.657913 | 8 | 0.125 |
| 0.001 | 0.656283 | 32 | 0.0078125 |
| 0.0001 | 0.656188 | 128 | 0.000976562 |

Таблица 4 – Формула трапеций

|  |  |  |  |
| --- | --- | --- | --- |
|  | Значение | n | h |
| 0.01 | 0.673925 | 16 | 0.03125 |
| 0.001 | 0.657303 | 32 | 0.00195312 |
| 0.0001 | 0.656316 | 128 | 0.000244141 |

Таблица 3 – Формула Симпсона

|  |  |  |  |
| --- | --- | --- | --- |
|  | Значение | n | h |
| 0.01 | 0.676177 | 8 | 0.02537 |
| 0.001 | 0.657451 | 16 | 0.0078125 |
| 0.0001 | 0.656334 | 32 | 0.000976562 |

**Выводы.**

В ходе выполнения лабораторной работы, используя квадратурные формулы прямоугольников, трапеций и Симпсона, были вычислено значение интеграла . Изучены методы нахождения значения интеграла тремя способами, а именно по формулам прямоугольников, трапеций и Симпсона. Написана программа для получения приближенных значений интеграла, для каждого метода было найдено по три значения интеграла, каждый раз стремясь достичь более близкого к точному решению значения. Были найдены погрешности близкие к значениям для каждого метода, используя правило Рунге. Зная погрешность, были найдены наибольшие длины (h) отрезков, на которые можно разделить отрезок [a,b]. Из полученных данных видно, что при расчёте интеграла по формуле Симпсона необходимо разделить отрезок [a,b] наименьшее количество отрезков, то есть наибольшее значение шага. Сравнивая данные, полученные при расчёте значений методами прямоугольников и трапеций, видно, что при более низкой точности метод прямоугольников имеет преимущество, так как имеет больший шаг.Приложение А

исходный код программы

#include <iostream>

#include <fstream>

#include <cmath>

#include <cstdlib>

using namespace std;

ofstream ff("output.txt");

double F(double x)//Функция

{

    return cos(x)\*exp(-pow(x, 2));

}

double F2(double x)//Вторая производная

{

    double result = ((4\*x\*x-3)\*cos(x) + 4\*x\*sin(x)) \* exp(-pow(x, 2));

    return result;

}

double F4(double x)//Четвертая производная

{

    double result = ((16\*x\*x\*x\*x-72\*x\*x+25)\*cos(x) + 8\*x\*(4\*x\*x-7)\*sin(x)) \* exp(-pow(x, 2));

    return result;

}

double RECT(double a, double b)

{

    return F((a+b)/2) \* (b-a);

}

double TRAP(double a, double b)

{

    return (F(a)+F(b)) \* (b-a)/2;

}

double SIMPS(double a, double b)

{

    return (F(a) + 4\*F((a+b)/2) + F(b)) \* (b-a)/6;

}

double RECT\_S(double a, double b, double h)

{

    double R1 = pow(h, 2) / 24 \* F2((a + b) / 2) \* (b - a);

    double xi;

    double result = 0;

    for (double i = 0; i < (b - a) / h - 1; i += 2)

    {

        xi = a + i \* h;

        result += F(xi + h / 2);

    }

    result = 2 \* h \* result + R1;

    return result;

}

double TRAP\_S(double a, double b, double h)

{

    double R2 = -pow(h, 2) / 12 \* F2((a + b) / 2) \* (b - a);

    double xi;

    double result = 0;

    for (double i = h; i < (b - a) / h - 1; i += 2)

    {

        xi = a + i \* h;

        result += F(xi) + F(xi - 2 \* h);

    }

    result = h \* result + R2;

    return result;

}

double SIMPS\_S(double a, double b, double h)

{

    double R3 = -pow(h, 4) / 180 \* F4((a + b) / 2) \* (b - a);

    double xi;

    double result = 0;

    for (double i = 2 \* h; i < (b - a) / h - 1; i += 2)

    {

        xi = a + i \* h;

        result += F(xi) + F(xi - 2 \* h) + F(xi - 4 \* h);

    }

    result = 2\*h / 3 \* result + R3;

    return result;

}

//Окончательное приближение

double RECT\_E(double a, double b, double h)

{

    return RECT\_S(a, b, h / 2) + (RECT\_S(a, b, h / 3) - RECT\_S(a, b, h)) / 3;

}

double TRAP\_E(double a, double b, double h)

{

    return TRAP\_S(a, b, h / 2) - (TRAP\_S(a, b, h / 3) - TRAP\_S(a, b, h)) / 3;

}

double SIMPS\_E(double a, double b, double h)

{

    return SIMPS\_S(a, b, h / 2) - (SIMPS\_S(a, b, h / 3) - SIMPS\_S(a, b, h)) / 15;

}

//Высчитывание погрешности

bool RUNGE\_RECT(double a, double b, double h, double Eps)

{

    int k = 2;

    double Delta = (fabs(RECT\_S(a, b, h / 2) - RECT\_S(a, b, h)) / (pow(2, k) - 1));

    if (Eps > Delta)

        return true;

    return false;

}

bool RUNGE\_TRAP(double a, double b, double h, double Eps)

{

    int k = 2;

    double Delta = (fabs(TRAP\_S(a, b, h / 2) - TRAP\_S(a, b, h)) / (pow(2, k) - 1));

    if (Eps > Delta)

        return true;

    return false;

}

bool RUNGE\_SIMPS(double a, double b, double h, double Eps)

{

    int k = 4;

    double Delta = (fabs(SIMPS\_S(a, b, h / 2) - SIMPS\_S(a, b, h)) / (pow(2, k) - 1));

    if (Eps > Delta)

        return true;

    return false;

}

int main()

{

    double a = 0, b = 1, h = 0.02, n = 0;

    bool RECT\_end = false, TRAP\_end = false, SIMPS\_end = false;

    double RECT\_n = 0, TRAP\_n = 0, SIMPS\_n = 0;

    ff << "Первое приближение:" << endl;

    ff << RECT(a, b) << endl;

    ff << TRAP(a, b) << endl;

    ff << SIMPS(a, b) << endl<< endl;

    ff << "Второе приближение:" << endl;

    ff << RECT\_S(a, b,h) << endl;

    ff << TRAP\_S(a, b,h) << endl;

    ff << SIMPS\_S(a, b,h) << endl<< endl;

    for (double Eps = 0.01; Eps >= 0.0001; Eps /= 10){

        bool RECT\_end = false, TRAP\_end = false, SIMPS\_end = false;

        for (n = 1; !(RECT\_end & TRAP\_end & SIMPS\_end); n \*= 2){

            h = (b - a) / n;

            if (!RECT\_end){

                RECT\_n = n;

                RECT\_end = RUNGE\_RECT(a, b, h, Eps);

            }

            if (!TRAP\_end){

                TRAP\_n = n;

                TRAP\_end = RUNGE\_TRAP(a, b, h, Eps);

            }

            if (!SIMPS\_end){

                SIMPS\_n = n;

                SIMPS\_end = RUNGE\_SIMPS(a, b, h, Eps);

            }

        }

        h = (b - a) / n;

        ff << endl << "H:" << h << endl;

        ff << endl << "Eps:" << Eps << endl;

        ff << "RECT: " << RECT\_E(a, b, h) << "\nn: " << RECT\_n << "\nh: " << (b-a)/RECT\_n << endl;

ff << "TRAP: " << TRAP\_E(a, b, h) << "\nn: " << TRAP\_n << "\nh: " << (b-a)/TRAP\_n << endl;

        ff << "SIMPS: " << SIMPS\_E(a, b, h) << "\nn: " << SIMPS\_n << "\nh: " << (b-a)/SIMPS\_n << endl;

    }

    return 0;

}