МИНИСТЕРСТВО НАУКИ и высшего образования РОССИЙСКОЙ ФЕДЕРАЦИИ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**

**ОТЧЁТ**

**по лабораторной работе №5**

**по дисциплине: «** *Вычислительная математика* **»**

**Вариант 2**

Выполнил(а):Проверил:

Студенты гр. *АП-227* *Ландовский В.В.*

*Бузмаков А.И.*

*Шестаков К.Д.*

*Федотов И.В.*

«\_\_\_» \_\_\_\_\_\_ 20\_\_г.«\_\_\_» \_\_\_\_\_\_ 20\_\_ г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск

2024

**Цель работы**

Знакомство с некоторыми методами численного интегрирования функций. Получение практических навыков разработки алгоритмов и программной реализации данных методов.

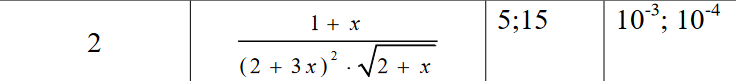
**Постановка задачи**

1. Проанализировать поведение подынтегральной функции, заданной согласно варианту задания в таблице 5.2. Вычислить оценки шага интегрирования для заданных преподавателем методов и значений точности из таблицы 5.2 используя формулы (5.2), (5.5), (5.7), (5.9). Расчеты выполнить вручную или с использованием любых доступных математических программных пакетов.

2. Разработать программную реализацию вычисления интеграла заданными методами с контролем точности основанном на правиле Рунге с учетом порядка точности метода. Входной информацией для разработанной программы должны быть: пределы интегрирования и требуемая точность Подынтегральная функция жестко задается в программе. Выходная информация: значение интеграла и шаг, при котором оно вычислено. При разработке алгоритма стараться по возможности минимизировать вычислительные затраты. Ограничений на среду разработки не накладывается.

3. Сравнить оценки шага со значениями шага, которые получены в результате работы программ, а значения интеграла полученные программно с результатом, полученным в стороннем математическом программном пакете с высокой точностью.

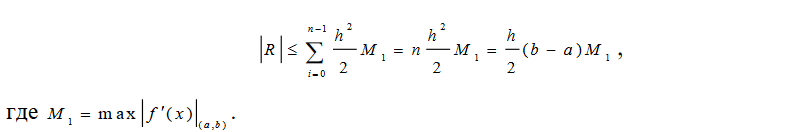
**Исходные данные**

****

1. Метод правых прямоугольников,
2. Метод средних прямоугольников,
3. 3 степени.

**Ход работы**

Для оценки шага метода прямоугольников используем формулу



Найдем значения первой, второй и четвертой производных функции :

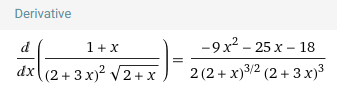


Рис. 1 - Первая производная функции

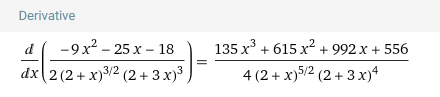


Рис. 2 - Вторая производная функции

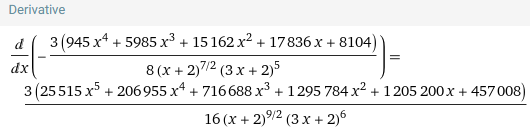


Рис. 3 - Четвертая производная функции

Построим график . Видим, что максимальное по модулю значение на отрезке [5; 15] находится в точке и равно .

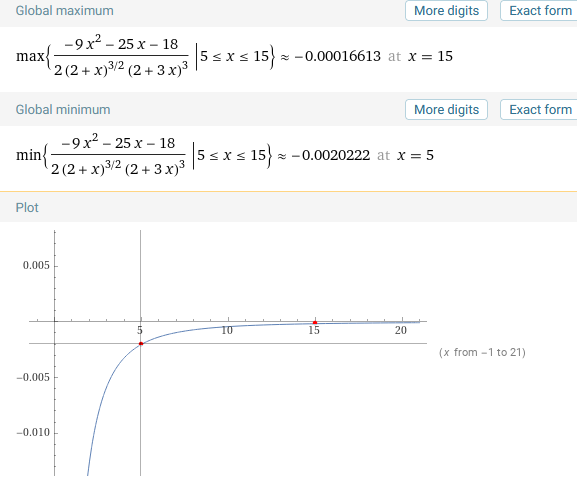


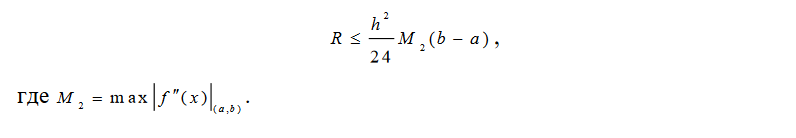
Рис. 4 - График первой производной функции

Найдем значение шага для метода правых прямоугольников, преобразовав формулу к виду

h ; для ε =

h для ε =

Для оценки шага метода средних прямоугольников используем формулу



Построим график . Видим, что максимальное по модулю значение на отрезке [5; 15] находится в точке и равно .

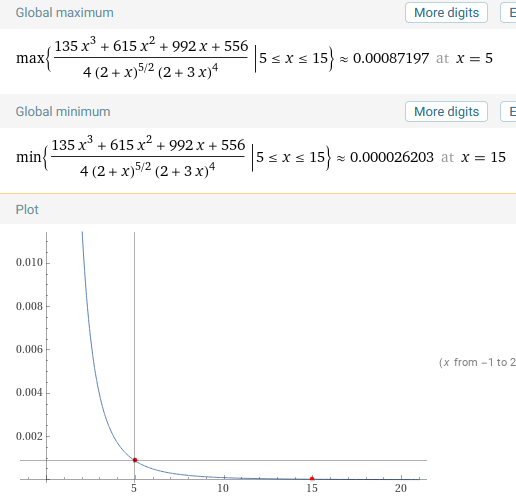


Рис. 5 - График второй производной

Найдем значение шага для метода средних прямоугольников, преобразовав формулу к виду

h ; для ε =

h для ε =

Для оценки шага метода Ньютона-Котеса 3 степени используем формулу

Построим график . Видим, что максимальное по модулю значение на отрезке [5; 15] находится в точке и равно .

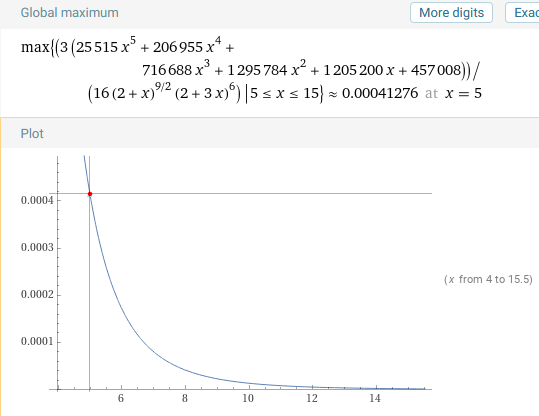


Рис. 6 - График четвертой производной функции

Найдем значение шага для метода Ньютона-Котеса 3 степени, преобразовав формулу к виду

h ; для ε =

h для ε =

Рассчитаем значение интеграла соответствующими методами с помощью математических пакетов:

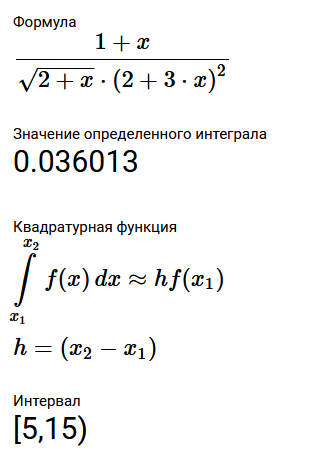


Рис. 7 - Решение интеграла методом правых прямоугольников

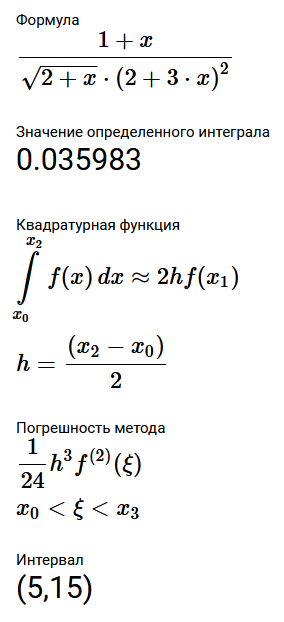
****

Рис. 8 - Решение интеграла методом центральных прямоугольников

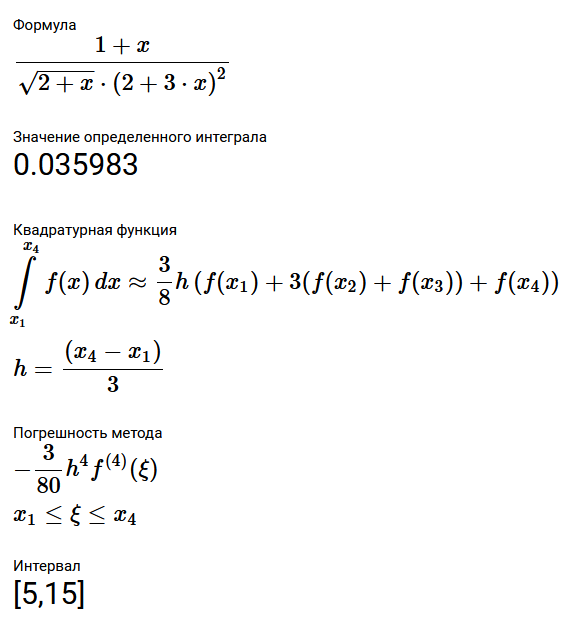
****

Рис. 9 - Решение интеграла методом Ньютона-Котеса 3 порядка

После выполненных расчетов приступим к программной реализации соответствующих методов и их сравнению.

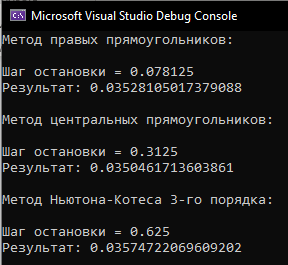


Рис. 10 - Результат работы программы при Epsilon = 10^-3

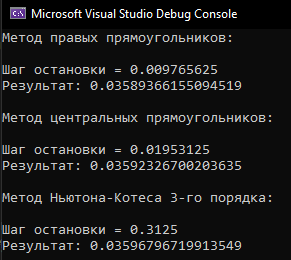


Рис. 11 - Результат работы программы при Epsilon = 10^-4

В таблице 1 представлено сравнение полученных результатов с теоретически вычисленными.

Таблица 1- Результаты вычислений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Точность | Метод | Теоретическое значение шага | Экспериментальное значение шага | Значение интеграла |
| 0,001 | Правых прямоугольников | 0,099 | 0,078125 | 0,3528105017379088 |
| Средних прямоугольников | 1,73 | 0,625 | 0,0350461713603861 |
| Ньютона-Котеса 3 степени | 1,85 | 0,8333 | 0,3574722069609202 |
| 0,0001 | Правых прямоугольников | 0,009 | 0,009765625 | 0,0358936615509451 |
| Средних прямоугольников | 0,548 | 0,078125 | 0,0359232670020363 |
| Ньютона-Котеса 3 степени | 1,17 | 0,4166 | 0,0359679671991354 |

**Объяснение полученных результатов**

Если сравнить полученные теоретические и экспериментальные значения шагов интегрирования, то можно сделать вывод о том, что программная реализация заданных методов удовлетворяет требованию точности вычислений. Значения интеграла, вычисленные программно, совпадают со значениями ручных вычислений c заданной точностью.

Для оценки погрешности интегрирования, например, составной формулы Симпсона (метод Ньютона-Котеса 3 степени) необходимо вычислить максимум модуля четвертой производной подынтегральной функции, что не всегда удобно. В нашей программной реализации мы использовали способ оценки погрешности по правилу Рунге. Его преимущество заключается в том, что метод не требует вычисления и (или) оценки производных и может применяться для любого из заданных методов интегрирования, что значительно упрощает вычисления в программе.

Плюсом численного интегрирование являются случаи, где подынтегральная функция задана не аналитически, а в виде таблицы или экспериментальной зависимости. В таких случаях численное интегрирование – это единственный способ расчета интеграла.

**Код разработанной программы**

|  |
| --- |
| Main.cs |
| namespace YourNamespace  {      class Program      {          // Функция          static double Func(double x)          {              return (1 + x) / (Math.Pow((2 + 3 \* x), 2) \* Math.Sqrt(2 + x));          }          // Правило Рунге          static bool RungeRule(double Sh, double Shr, double epsilon, double r, double p)          {              return (Math.Abs(Sh - Shr) / (Math.Pow(r, p) - 1)) < epsilon;          }          // Метод правых прямоугольников          static double RightRec(double epsilon, double r, double h, double a, double b)          {              double sum = 0;              double sumResult = 0;              double x = a + h;              // Первичный подсчет              while (x <= b + h)              {                  sum += Func(x) \* h;                  x += h;              }              // Процесс уменьшения шага для достижения заданной точности              while (!RungeRule(sum, sumResult, epsilon, r, 1)) // Порядок точности равен 1              {                  x = a + h;                  sum = sumResult;                  sumResult = 0;                  h /= r;                  while (x <= b + h)                  {                      sumResult += Func(x) \* h;                      x += h;                  }              }              Console.Write("Шаг остановки = " + h);              return sumResult;          }          // Метод средних прямоугольников          static double MiddleRec(double epsilon, double r, double h, double a, double b)          {              double sum = 0;              double sumResult = 0;              double x = (a + b) / 2;              // Первичный подсчет              while (x <= b)              {                  sum += Func(x) \* h;                  x += h;              }              //Процесс уменьшения шага для достижения заданной точности              while (!RungeRule(sum, sumResult, epsilon, r, 2)) // Порядок точности равен 1              {                  x = (a + a + h) / 2;                  sum = sumResult;                  sumResult = 0;                  h /= r;                  while (x <= b)                  {                      sumResult += Func(x) \* h;                      x += h;                  }              }              Console.Write("Шаг остановки = " + h);              return sumResult;          }          // Метод Ньютона-Котеса          static double Simpson(double a, double b)          {              double h = (b - a) / 3;              // Две дополнительные точки (n+1), n = 3              double mid = a + h;              double mid2 = mid + h;              // Результат минус погрешность (3/80\*h^5)              double res = (3 \* h / 8) \* (Func(a) + 3 \* Func(mid) + 3 \* Func(mid2) + Func(b)) - 3 \* Math.Pow(h, 5) / 80;              return res;          }          static double NewtonCotes3(double a, double b, double r, double eps)          {              double result = 0, sum = 0;              double intervalCount = r;              int k = 1;              double h = (b - a) / 3;              // Первичный подсчет              for (int i = 0; i < intervalCount; i++)              {                  double low = a + i \* h;                  double up = low + h;                  result += Simpson(low, up);                  //Console.WriteLine(low + " " + up);              }              while (!RungeRule(sum, result, eps, 2, 4) && k < 20)              {                  sum = result;                  result = 0;                  intervalCount \*= 2;                  h /= 2;                  k++;                  for (int i = 0; i < intervalCount; i++)                  {                      double low = a + i \* h;                      double up = low + h;                      result += Simpson(low, up);                  }              }              Console.Write("Шаг остановки = " + h);              return result;          }          static void Main(string[] args)          {              Console.Clear();              double a = 5;              double b = 15;              double eps = 1e-4;              double h = (b - a);              const double r = 2;              double result;              Console.WriteLine("Метод правых прямоугольников:\n");              result = RightRec(eps, r, h, a, b);              Console.WriteLine("\nРезультат: " + result + "\n");              Console.WriteLine("Метод центральных прямоугольников:\n");              result = MiddleRec(eps, r, h, a, b);              Console.WriteLine("\nРезультат: " + result + "\n");              Console.WriteLine("Метод Ньютона-Котеса 3-го порядка:\n");              result = NewtonCotes3(a, b, r, eps);              Console.WriteLine("\nРезультат: " + result);          }      }  } |