Московский государственный университет
имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчет по заданию №6

«Сборка многомодульных программ. Вычисление корней уравнений и определенных интегралов.»

Вариант 7 / 4 / 1

Выполнил:
студент 102 группы
Мартьянов А. О.

Преподаватель:
Кулагин А. В.

Москва
2022

Содержание

Постановка задачи

Требуется с помощью квадратурной формулы вычислить с точностью $\varepsilon = 0.001$ площадь плоской фигуры, ограниченной кривыми:

- $f_1 = \ln x$
- $f_2 = -2x + 14$
- $f_3 = \frac{1}{2-x} + 6$

Сначала требуется с помощью комбинированного метода(хорд и касательных) найти абсциссы точек пересечения кривых, предварительно определив подходящие под условия применимости методов отрезки. Далее произвести приближеным вычисления интрегралов, откуда найти площадь плоской фигуры.

Под каждую задачу требуется реализовать соответствующие Си-функции `root(f, g, df, dg, a, b ,eps1)` и `integral(f, a, b, eps2)`, вычисляющие нужные значения с заданной точностью, которую нужно подобрать таким образом, чтобы итоговая точность вычислений площади состовляла ε.

Математическое обоснование

Графики заданных кривых (рис. 1).

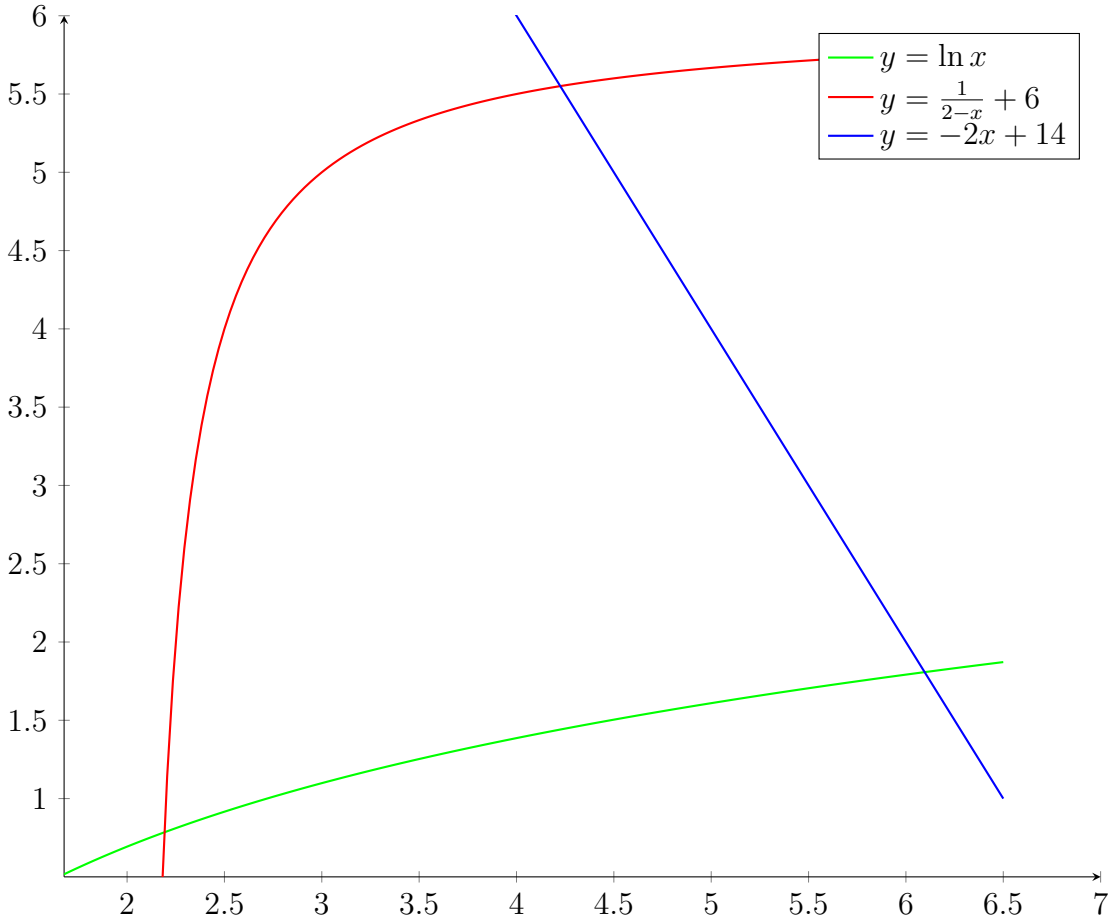

Рис. 1: Плоская фигура, ограниченная графиками заданных уравнений

Выбор отрезков для нахождения точек пересечения[1]

Для корректного применения методов приближенного уравнения $F(x) = 0$(где $F(x) = f(x) - g(x)$) необходимо найти отрезок [a,b], на концах которого функция $F(x)$ имеет разные знаки и на всем отрезке производная функции не меняет знак. Кроме того требуется, чтобы на этом отрезке первая и вторая производные функции не меняли свой знак. Находим приближенные решения уравнений для соответсвующих функций с точностью $\varepsilon_1 = 0.0001$:
[2.0001, 3.0] для f_1 и f_3
[4.0, 5.0] для f_2 и f_3
[5.5, 6.5] для f_1 и f_2

Вычисление приближенных значений интеграла и выбор ε_1 и ε_2:[2] Вычисление интеграла с точностью ε_2 происходит с помощью квадратурной формулы прямоугольников. Требуемая точность достигается с помощью *правила Рунге:*

$$|I - I_n| \cong p|I_n - I_{2n}|$$

Для нашего метода $p = 1/3$. Точность вычислений ε_2 достигается за счет разбиения отрезка на более мелкие части, а количество этих частей динамически увеличивается до момента пока соседние значения интегралов не станут ближе, чем ε_2.

Результаты экспериментов

Результаты вычисления пересечений кривых приведены в таблице(таблица 1):

Кривые	x	y
1 и 2	6.0962	5.7559
2 и 3	4.2247	5.5506
1 и 3	2.1917	0.7847

Таблица 1: Координаты точек пересечения

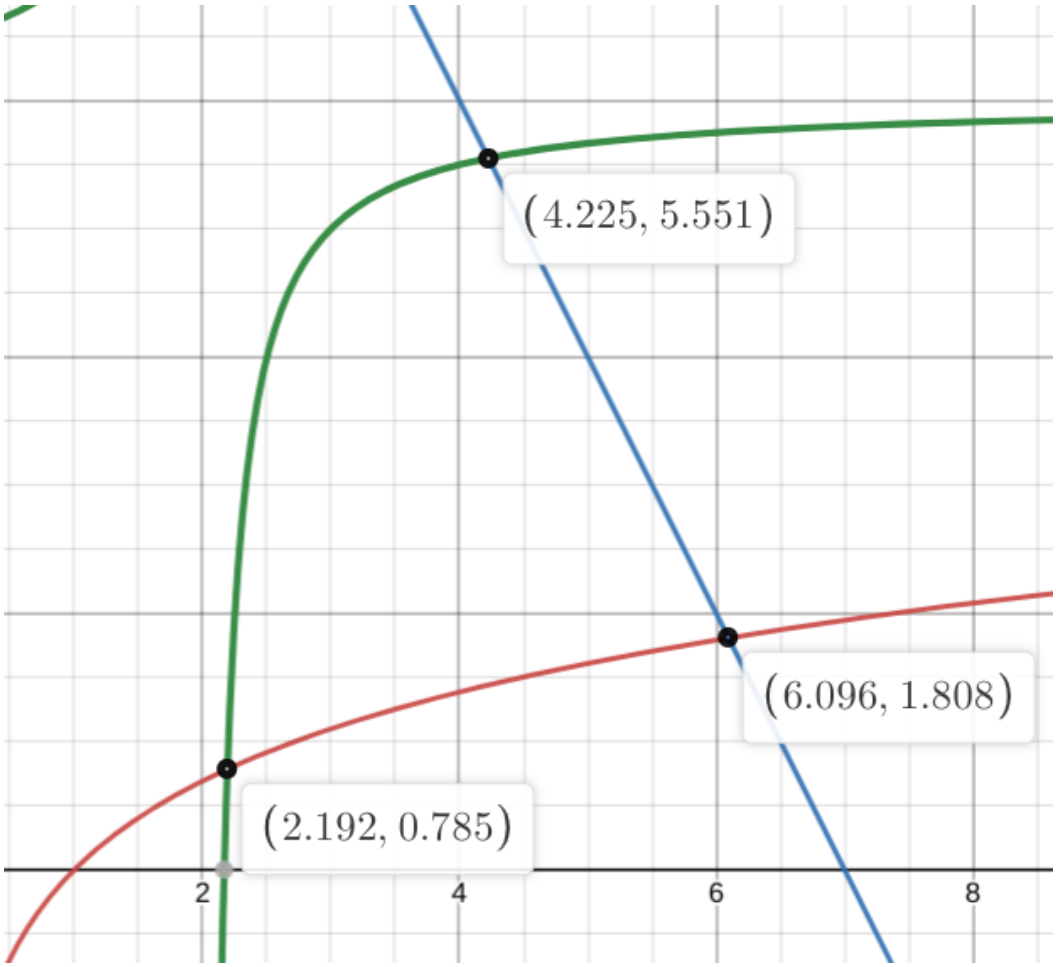

Рис. 2: Отображение точек пересечения с их координатами

Структура программы и спецификация функций

Программа состоит из модуля main.c на языке C и модуля funcs.asm на языке Ассемблера.

Модуль main.c:
Для удобства определим собственный тип: `typedef double(*func)(double x);`

1. `double rounding(double x, double eps);`
   Функция, округляющая переданное в качестве аргумента число x до eps знаков после запятой.

2. `int main(int argc, char *argv[])`
   Функция main, принимающая на вход аргументы argc, argv из командной строки для управления ходом исполнения программы.

3. `double root_case_1(func f1, func f2, func df1, func df2, double a, double b, double eps);`
   Функция приближенного нахождения корня с помощью смешанного метода в случае 1: если функция возрастает и ее график расположен ниже хорды или если функция убывает и ее график расположен выше хорды.

4. `double root_case_2(func f1, func f2, func df1, func df2, double a, double b, double eps);`
   Функциональность аналогична функции root_case_1, только для случая 2: если функция возрастает и ее график расположен выше хорды или если функция убывает и ее график расположен ниже хорды.

5. `double root(func f1, func f2, func df1, func df2, double a, double b, double eps);`
   Функция, анализирующая переданные в качестве входных параметров функции f1 и f2 и определяющая, какую из функций root_case_1, root_case_2 запускать. Возвращает результат, округленный до точности eps.

6. `double integral_n(func F, int n, double a, double b);`
   Функция, вычисляющая приближенное значение интеграла на заданном промежутке [a, b] с помошью метода прямоугольников с количеством отрезков разбиения, равным переданному аргументу n.

7. `double integral(func f, double a, double b, double eps);`
   Функция, вычислиющая приближенное значение интеграла на отрезке [a, b] с заданной точностью eps.

8. `void test_root(func f1, func f2, func f3, double eps);`
   Функция, запускающая набор заранее заданных тестов для функции root. Проверяет совпадение результата работы функции с правильным ответом с точностью eps.

9. `void test_integral(func f1, func f2, func f3, double eps)`
   Функция, запускающая набор заданных заранее тестов для функции integral и проверяющая совпадения рассчитанного значения с истинным с точностью eps.

Модуль funcs.asm:

1. `f1`
   Принимает значение абсциссы и рассчитывает значение f_1 в этой точке.
2. `f2`
   Принимает значение абсциссы и рассчитывает значение f_2 в этой точке.
3. `f3`
   Принимает значение абсциссы и рассчитывает значение f_3 в этой точке.
4. `df1`
   Принимает значение абсциссы и рассчитывает значение производной функции f_1 в этой точке.
5. `df2`
   Принимает значение абсциссы и рассчитывает значение производной функции f_2 в этой точке.
6. `df3`
   Принимает значение абсциссы и рассчитывает значение производной функции f_3 в этой точке.

Сборка программы (Make-файл)

```
LIBS = -lm
C_NAME = main
EXEC_NAME = exec
ASM_NAME = funcs

clean: all
	rm -rf *.o
all:
	gcc -m32 -c ${C_NAME}.c -o ${C_NAME}.o
	nasm -f elf -o ${ASM_NAME}.o ${ASM_NAME}.asm
	gcc -m32 ${C_NAME}.o ${ASM_NAME}.o ${LIBS} -o ${EXEC_NAME}

start: clean
	./${EXEC_NAME}

test_integral: clean
	./${EXEC_NAME} -i

test_root: clean
	./${EXEC_NAME} -r

test_root_manual: clean
	./${EXEC_NAME} --test-root-manual

test_integral_manual: clean
	./${EXEC_NAME} --test-integral-manual
```

Рис. 3: Makefile

Созданы переменные LIBS, C_NAME, EXEC_NAME, ASM_NAME для удосбтва работы. Первой стоит цель `clean`, зависящая от цели `all`, за счет чего при любом сценарии использования make все .o файлы удаляются.
Также опрделены цели:

1. `start` для одновременной сборки и запуска без флагов
2. `test_integral` для сборки и запуска с флагами для проверки заданного набора тестов для функции `integral`
3. `test_root` для сборки и запуска с флагами для проверки заданного набора тестов для функции `root`
4. `test_root_manual` для сборки и запуска с флагами для ручного тестирования функции `root`
5. `test_integral_manual` для сборки и запуска с флагами для ручного тестирования функции `integral`

Отладка программы, тестирование функций

Тестирование функций запускается флагами `--test-root[-r]`, `--test-integral[-i]` для проверки стандартных тестов и флагами `--test-root-manual`, `--test-integral-manual` для ввода тестов самим пользователем.

Уравнение	Левая граница	Правая граница	Значение
f1	0.015	2.0	-0.534910
f1	5.7	14.2	19.255388
f1	3.6	9.6	11.101604
f2	-14.6542	-5.435740	314.256748
f2	-3.5654	2.44645	90.892860
f2	17.4634	25.46453	-231.456129
f3	-14.543634	-1.435320	80.221685
f3	-2.342650	1.453260	24.847418
f3	5.325426	14.432643	53.324620

Таблица 2: Тестирование приближенного вычисления интеграла

Проверка точности проивзеденных рассчетов производилась при помощи сервисов `wolframalpha.com и desmos.com`.

Программа на Си и на Ассемблере

Исходные тексты программ на языках `C и Assembler` находятся в архиве, приложенном к данному отчету.

Анализ допущенных ошибок

В процессе работы ошибок допущено не было.

Список литературы

[1] Ильин В. А., Садовничий В. А., Сендов Бл. Х. Математический анализ. Т. 1 — Москва: Наука, 1985.

[2] «Задания практикума на ЭВМ» Трифонов Н.П., Пильщиков В.Н.