Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

По дисциплине

“Тестирование программного обеспечения”

Вариант: 67839

Выполнил:
Стрельбицкий Илья Павлович

Группа: P33101

Преподаватель:
Машина Екатерина Алексеевна

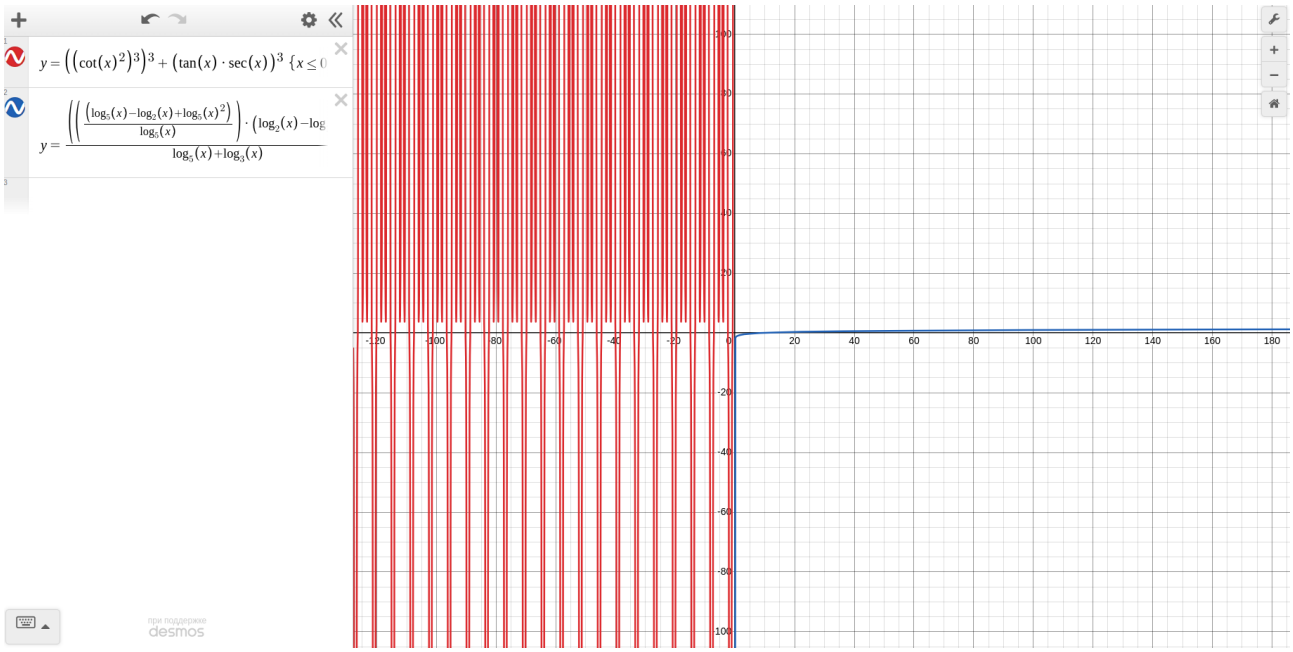
Санкт-Петербург, 2024г

Текст задания

Провести интеграционное тестирование программы, осуществляющей вычисление системы функций (в соответствии с вариантом).

$$\begin{cases} \left(\left(\left(\left(\cot(x)^2\right)^3\right)^3\right) + (\tan(x) \cdot \sec(x))\right)^3 & \text{if } x \leq 0 \\ \left(\frac{\left(\frac{(\log_5(x) - \log_2(x)) + \left(\log_5 (x)^2\right)}{\log_5(x)}\right) \cdot (\log_2(x) - \log_{10}(x))}{\log_5(x) + \log_3(x)}\right) & \text{if } x > 0 \end{cases}$$

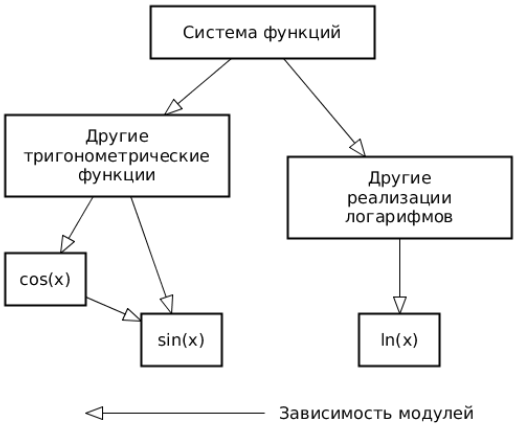
x <= 0 : (((((cot(x) ^ 2) ^ 3) ^ 3) + (tan(x) * sec(x))) ^ 3)
x > 0 : (((((log_5(x) - log_2(x)) + (log_5(x) ^ 2)) / log_5(x)) * (log_2(x) - log_10(x))) / (log_5(x) + log_3(x)))



Ссылка на график: https://www.desmos.com/calculator/q2j9jqb9si

Правила выполнения работы:

1. Все составляющие систему функции (как тригонометрические, так и логарифмические) должны быть выражены через базовые (тригонометрическая зависит от варианта; логарифмическая - натуральный логарифм).
2. Структура приложения, тестируемого в рамках лабораторной работы, должна выглядеть следующим образом (пример приведён для базовой тригонометрической функции sin(x)):



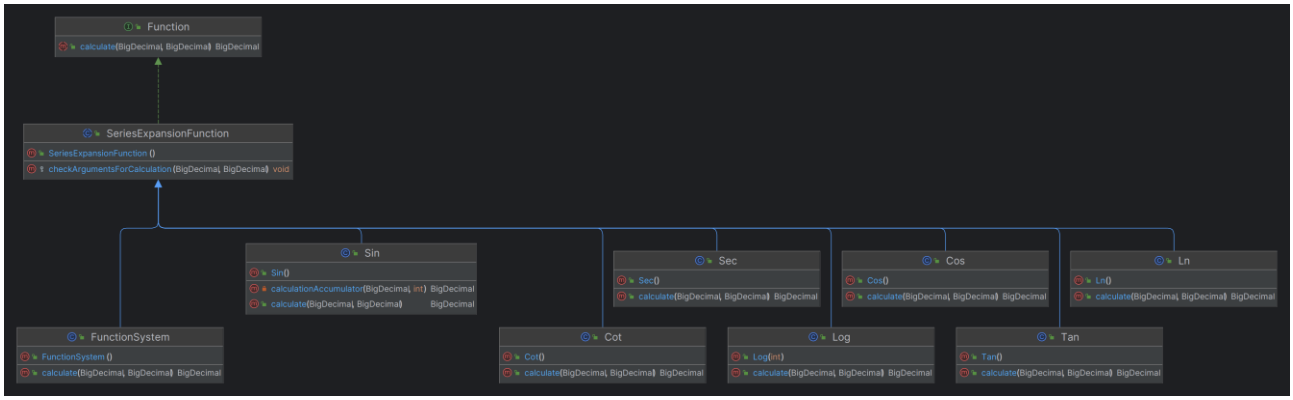
3. Обе «базовые» функции (в примере выше - sin(x) и ln(x)) должны быть реализованы при помощи разложения в ряд с задаваемой погрешностью. Использовать тригонометрические / логарифмические преобразования для упрощения функций ЗАПРЕЩЕНО.
4. Для КАЖДОГО модуля должны быть реализованы табличные заглушки. При этом, необходимо найти область допустимых значений функций, и, при необходимости, определить взаимозависимые точки в модулях.
5. Разработанное приложение должно позволять выводить значения, выдаваемое любым модулем системы, в csv файл вида «X, Результаты модуля (X)», позволяющее произвольно менять шаг наращивания X. Разделитель в файле csv можно использовать произвольный.

Порядок выполнения работы:

1. Разработать приложение, руководствуясь приведёнными выше правилами.
2. С помощью JUNIT4 разработать тестовое покрытие системы функций, проведя анализ эквивалентности и учитывая особенности системы функций. Для анализа особенностей системы функций и составляющих ее частей можно использовать сайт https://www.wolframalpha.com/.
3. Собрать приложение, состоящее из заглушек. Провести интеграцию приложения по 1 модулю, с обоснованием стратегии интеграции, проведением интеграционных тестов и контролем тестового покрытия системы функций.

Исходный код

Ссылка на github - https://github.com/stoneshik/tpo-second-lab



UML-диаграмма классов разработанного приложения.

Описание тестового покрытия с обоснованием его выбора

Для интеграционных тестов были выбраны крайние точки периодических функций, в которых функция меняет знак на противоположный, по точке с каждой стороны внутри периода функции и также точки в которых функция неопределенна.

Sin(x)

X	Y
-3	-0.1411
0	0
1	0.8415
π/2	1
-3+2*π	-0.1411

Cos(x)

X	Y
-3	-0.99
0	1
1	0.5403
π/2	0
-3+2*π	-0.99

Tan(x)

X	Y
-3	0.1425
0	0
-3+π	0.1425
1	1.5575
π/2	undefined

Cot(x)

X	Y
-3	7.0163
0	undefined
-3+π	7.0163
1	0.6421
π/2	0

Sec(x)

X	Y
-3	-2.4
0	1
1	1.9
π/2	undefined
-3+2*π	-2.4

Ln(x)

X	Y
0	undefined
0.1	-2.30248
1	0
2	0.69315

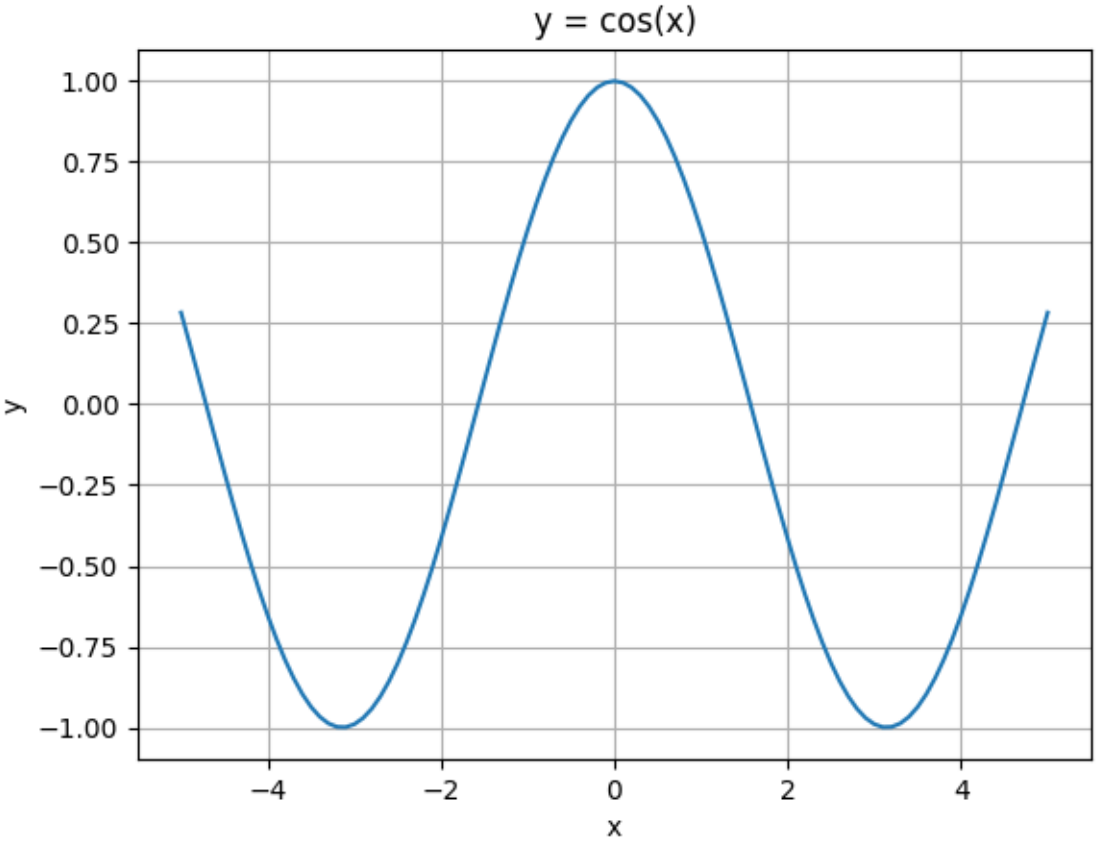
Log5(x)

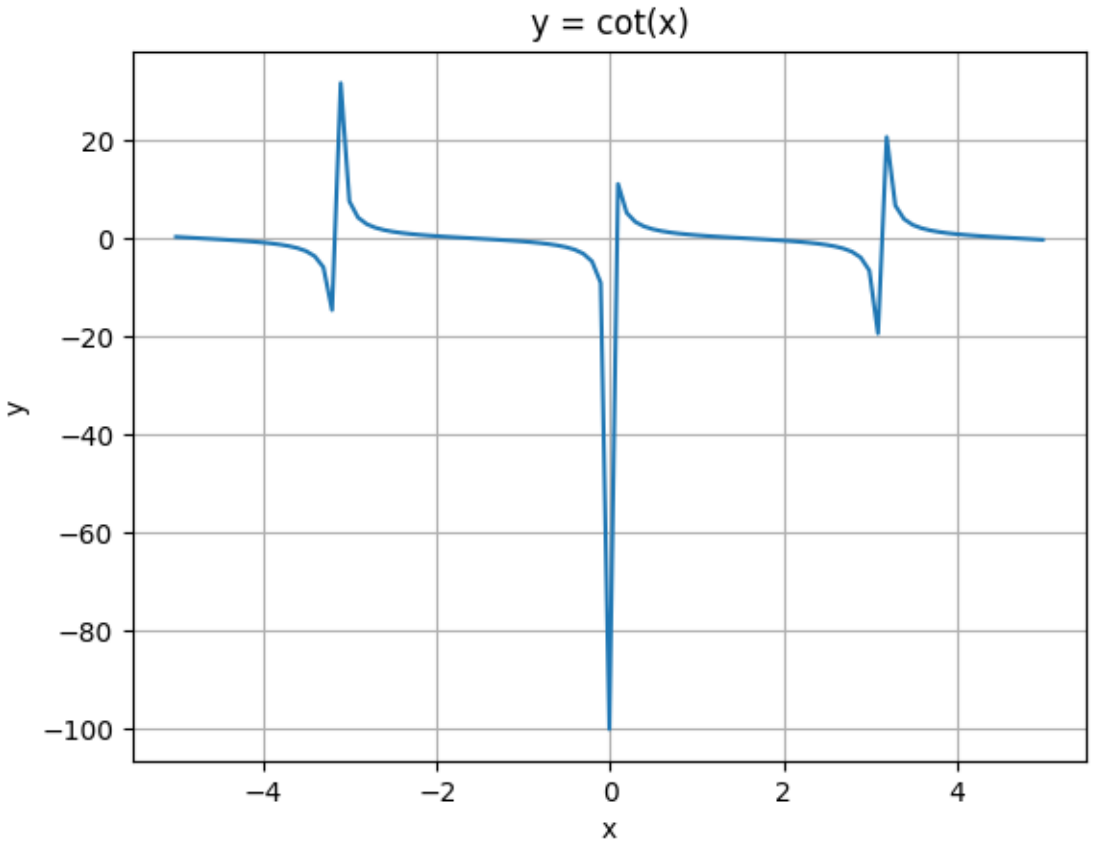
X	Y
0	undefined
0.4	-0.5694
1	0
1.2	0.1134

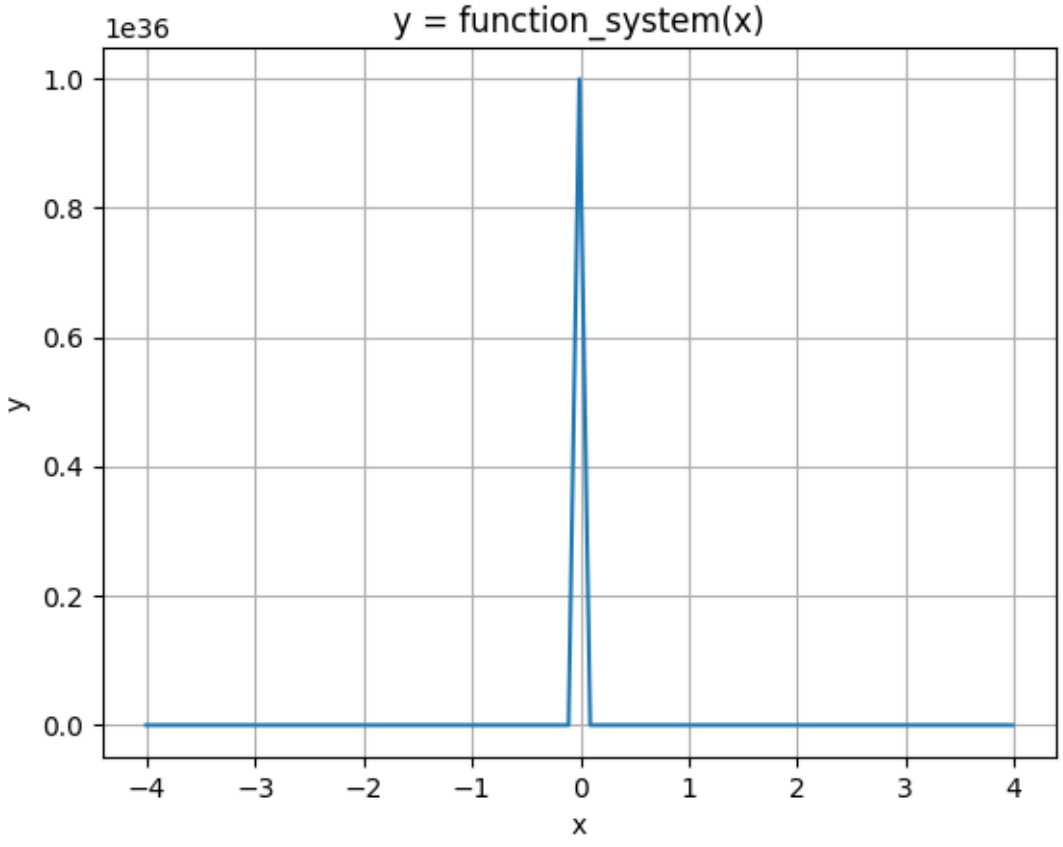
FunctionSystem(x)

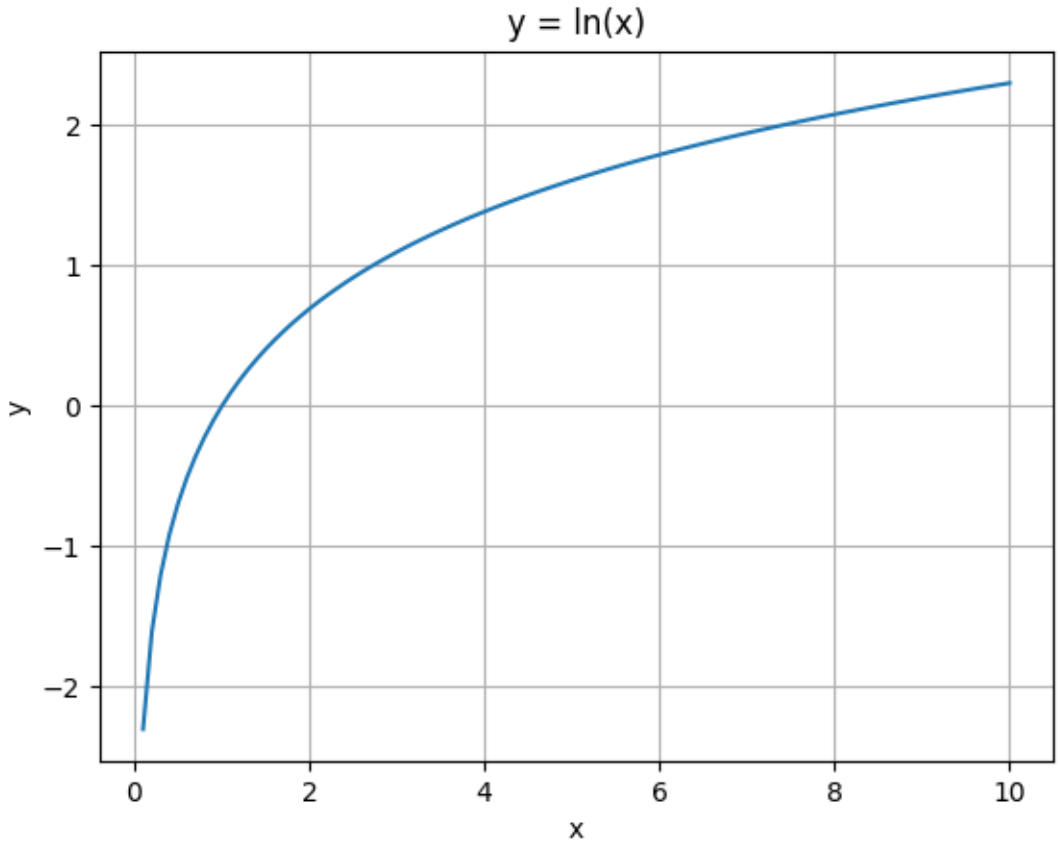
X	Y
-2	
0	undefined
1	undefined
2	-0.58680724

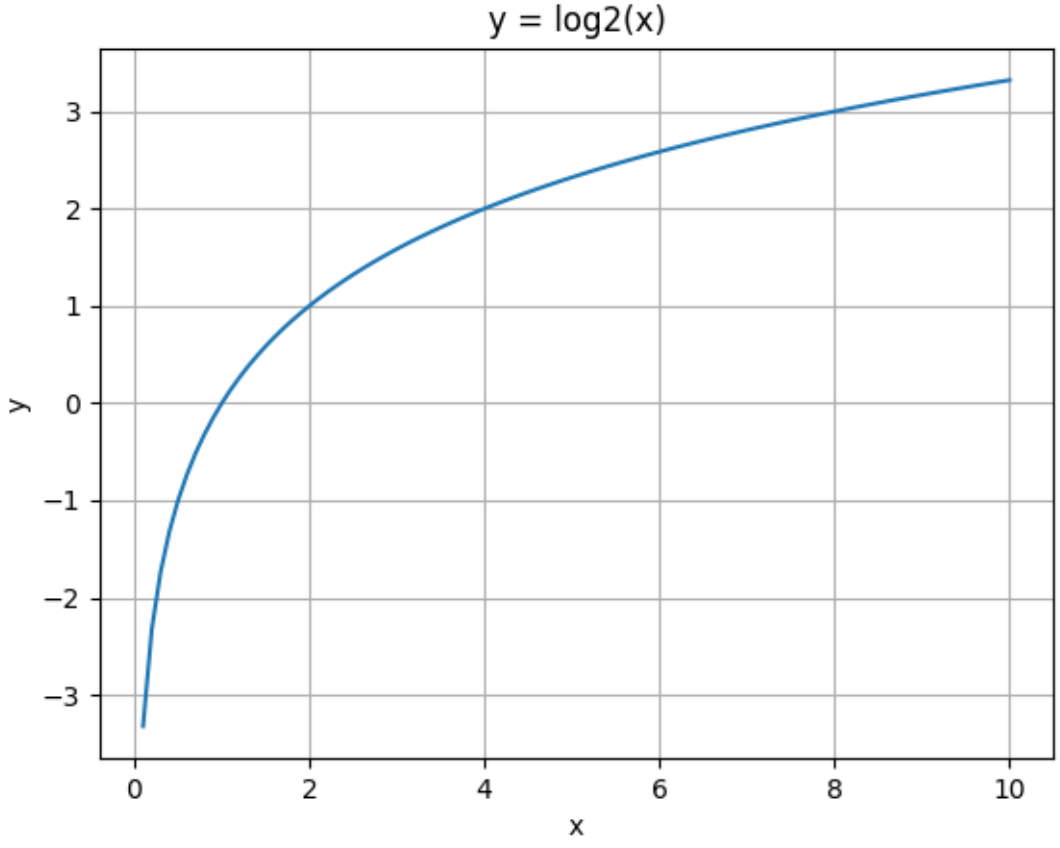
Графики построенные по csv-выгрузками, полученным в процессе интеграции приложения

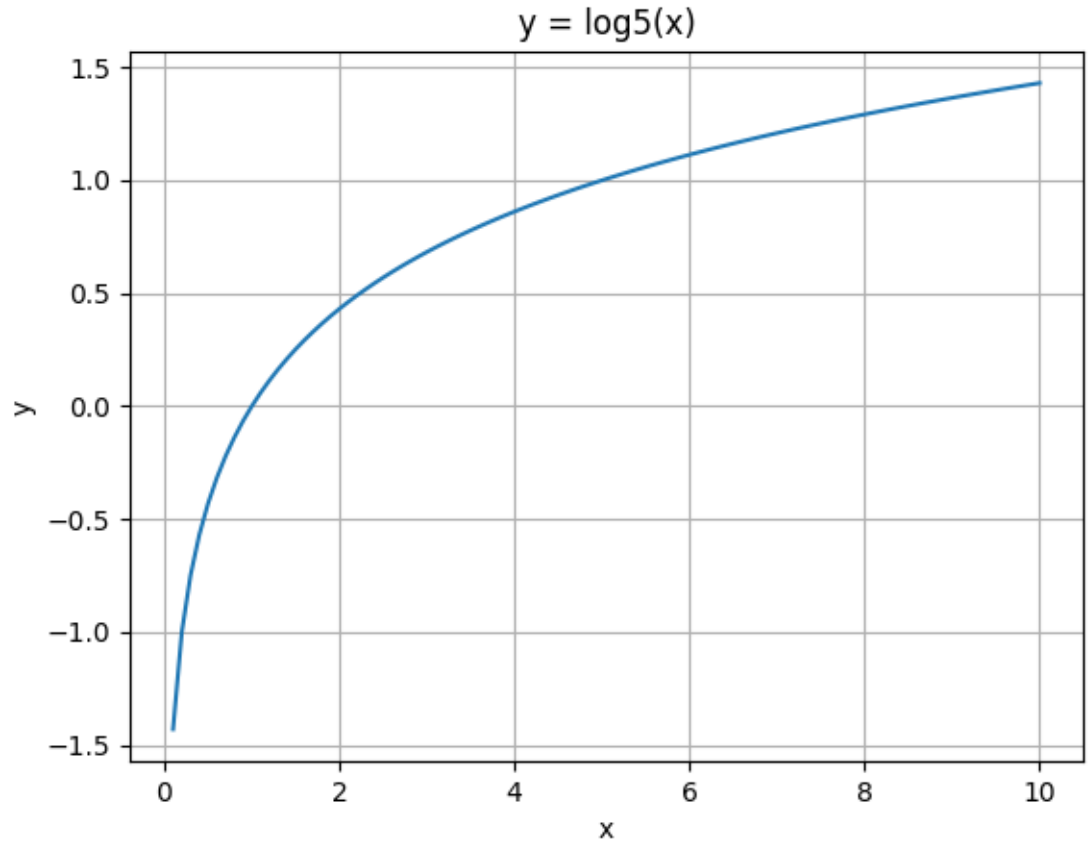

y = cos(x)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y
−4
−2
0
2
4
x

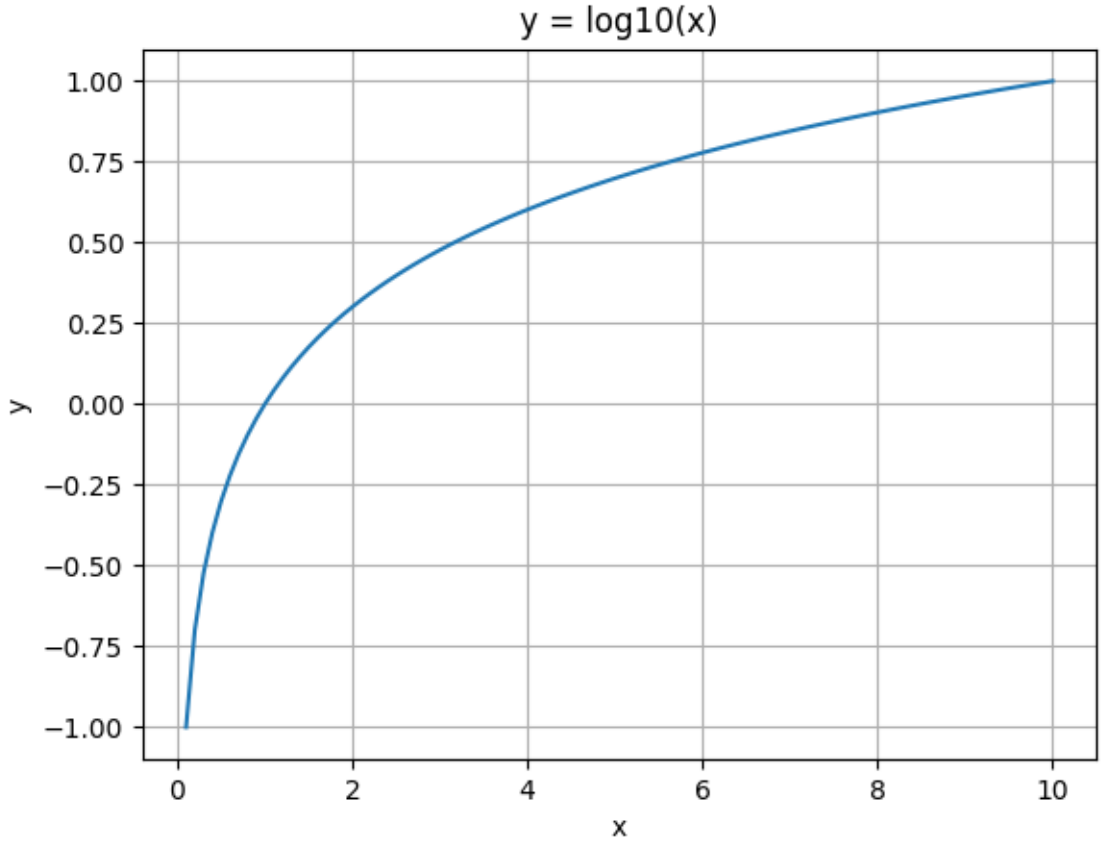

y = cot(x)
20
0
−20
−40
−60
−80
−100
y
−4
−2
0
2
4
x

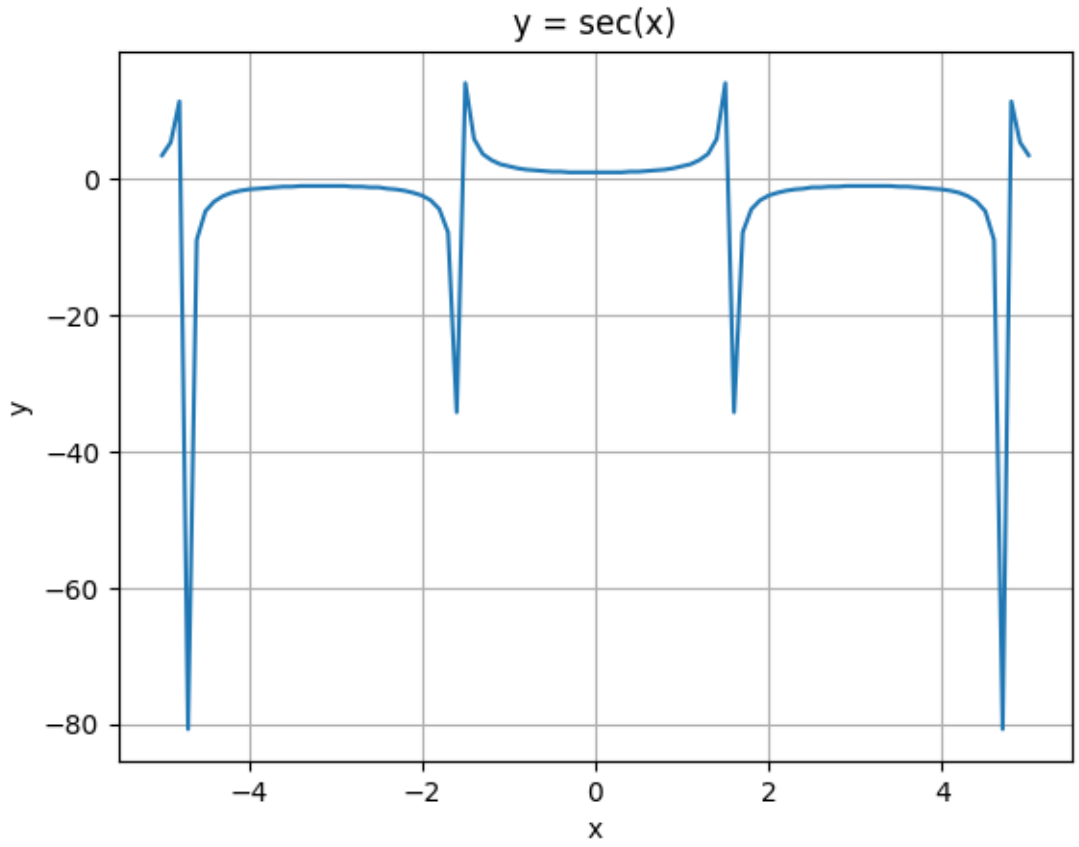

y = function_system(x)
1e36
1.0
0.8
0.6
0.4
0.2
0.0
y
−4
−3
−2
−1
0
1
2
3
4
x

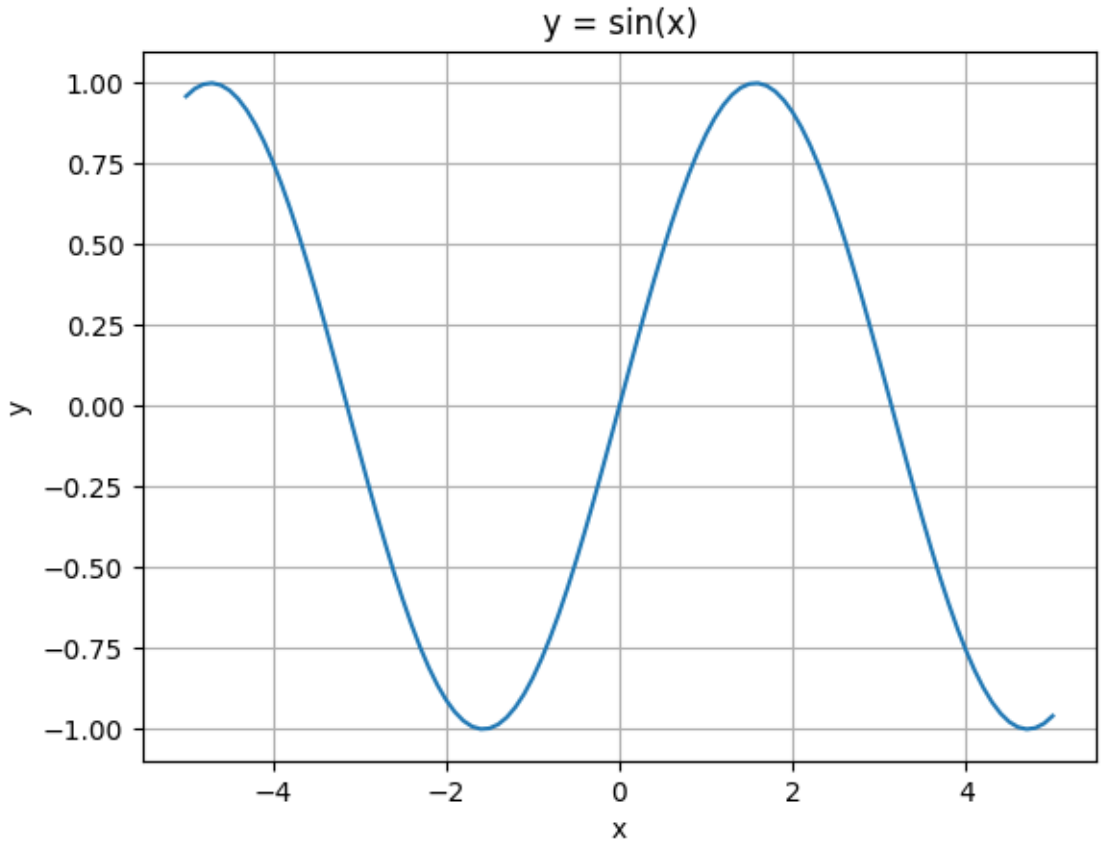

y = ln(x)
2
1
0
−1
−2
y
0
2
4
6
8
10
x

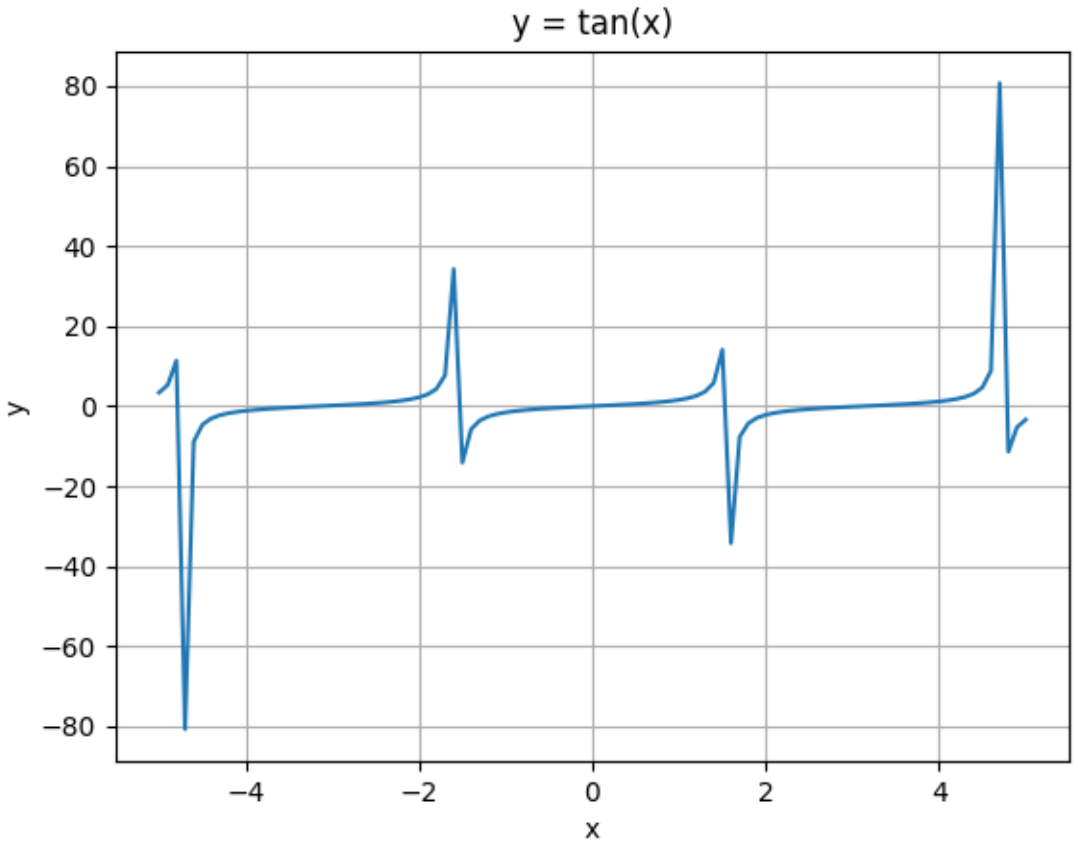

y = log2(x)
3
2
1
0
−1
−2
−3
y
0
2
4
6
8
10
x


y = log5(x)
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y
0
2
4
6
8
10
x


y = log10(x)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y
0
2
4
6
8
10
x


y = sec(x)
0
−20
−40
−60
−80
y
−4
−2
0
2
4
x


y = sin(x)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y
−4
−2
0
2
4
x



Вывод

В ходе выполнения данной лабораторной работы были изучены методы проведения интеграционного тестирования приложения написанного на Java при помощи Junit5. Основной сложность при проведении интеграционного тестирования является подбор возвращаемых заглушками вызываемых модулей значений, так как это требует глубокого анализа тестируемого кода и предметной области.