Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление программная инженерия
Образовательная программа системное и прикладное программное обеспечение
Специализация системное программное обеспечение

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

курса «Тестирование программного обеспечения»

по теме: «Интеграционное тестирование»

Вариант № 133232

Выполнил студент:
Тюрин Иван Николаевич
группа: P33102

Преподаватель:
Клименков С.В.,
Харитонова А.Е.

Санкт-Петербург, 2024 г.

Содержание

Лабораторная работа № 2 Интеграционное тестирование

1. Задание варианта № 133232

, , ,

Лабораторная работа #2

$$y = \begin{cases} (((\cos(x) - \csc(x))^2) + \cos(x)), & \text{при } x \leq 0 \\ (\dfrac{\dfrac{((\log_3(x) - \log_3(x)) \cdot \log_{10}(x))^3}{\ln(x)}}{\dfrac{(\log_2(x) - (\ln(x) + \log_{10}(x))) - \log_{10}(x)}{\log_5(x) + \log_5(x)}}), & \text{при } x > 0 \end{cases}$$

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 $\sin(x)$):

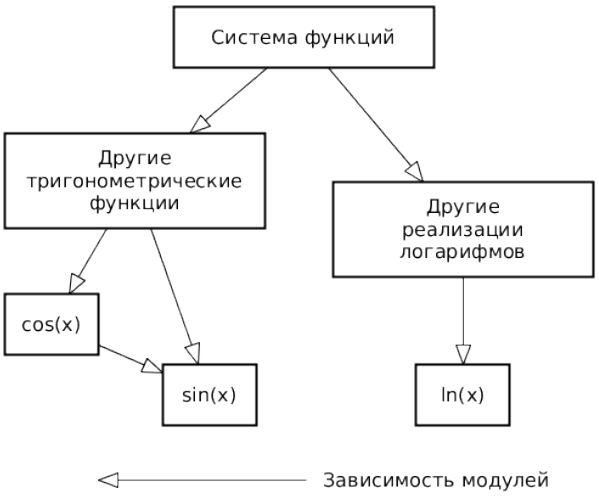

Рис. 1.1: Структура модулей в ЛР

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.

, , ,

2. Выполнение задания

В соответствии с заданием были реализованы базовые функции $\sin(x)$ и $\ln(x)$, которые использовались для реализации других используемых тригонометрических и логарифмических функций в соответствии с математическими формулами.

2. 1. Детали реализации модели

Для сохранения значений системы в csv-файл используется отдельная функция, а в качестве «заглушек» модулей используются библиотечные реализации функций.

Код программы для выполнения заданий был выполнен на языке программирования Kotlin с использованием библиотеки Kotest, которая использует внутри себя Junit 5 для JVM-таргетов. В реализации модели использовался один класс результирующей функции, унаследованный от типа функции с аргументом и возвращаемым значением типа Double, представленной на рис. 1.2. Этот класс в качестве аргументов своего конструктора принимает реализации математических функций.

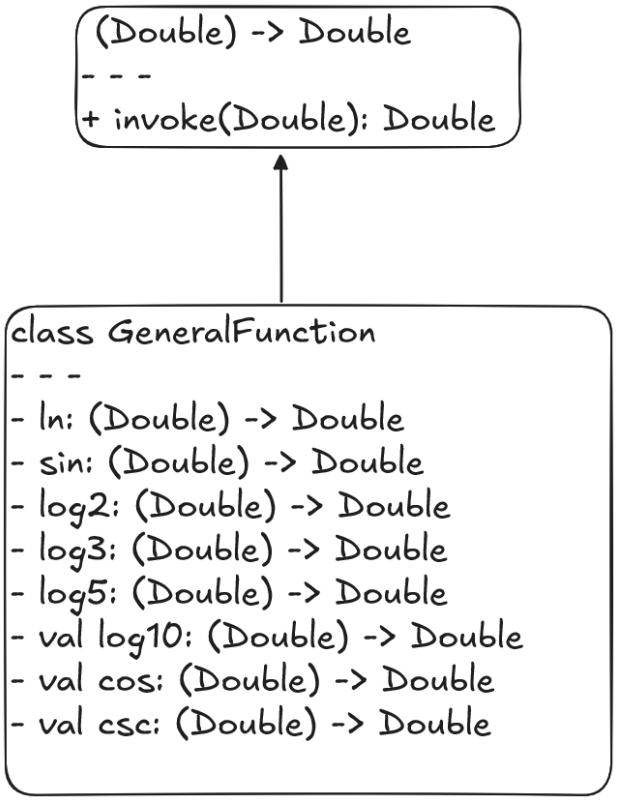

Рис. 1.2: UML-диаграмма программы

2. 2. Анализ эквивалентности

С помощью приложения для графического исследования математических выражений Desmos было выяснено, что на промежутке $x > 0$ функция тождественно равна 0, а на промежутке $x < 0$ она периодична, см. рисунок 1.3.

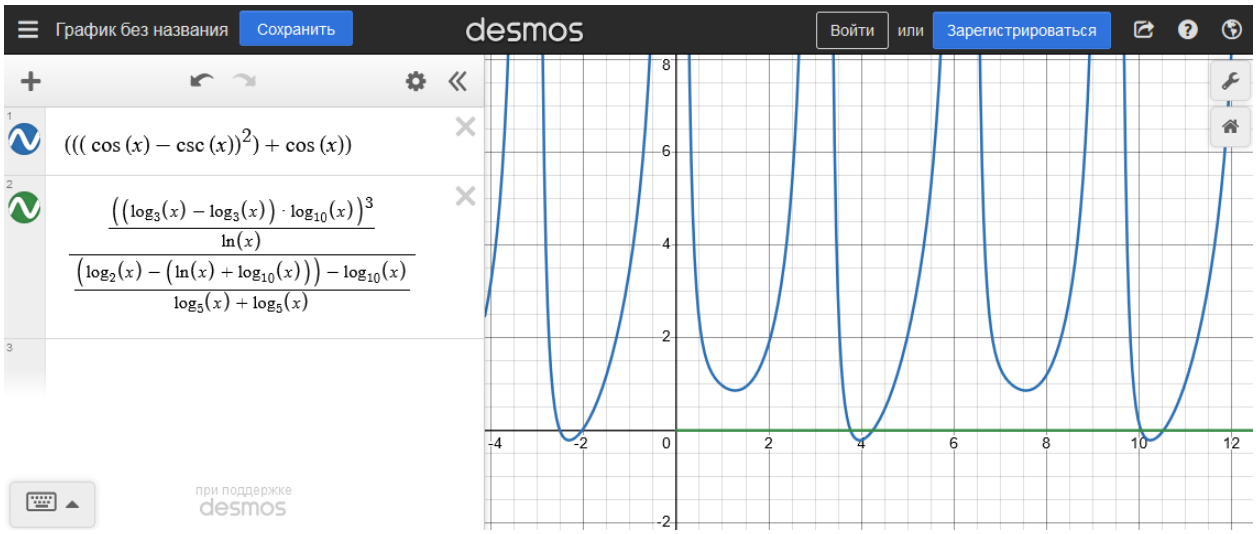

Рис. 1.3: Область допустимых значений системы

В качестве области тестирования было принято решение взять следующие промежутки: $(-2\pi; -\pi) \cup (-\pi; 0] \cup (0; 1) \cup (1; 5)$, визуальное представление которых можно видеть на рисунке 1.4. Промежутки отражают области эквивалентности с учетом области допустимых значений системы.

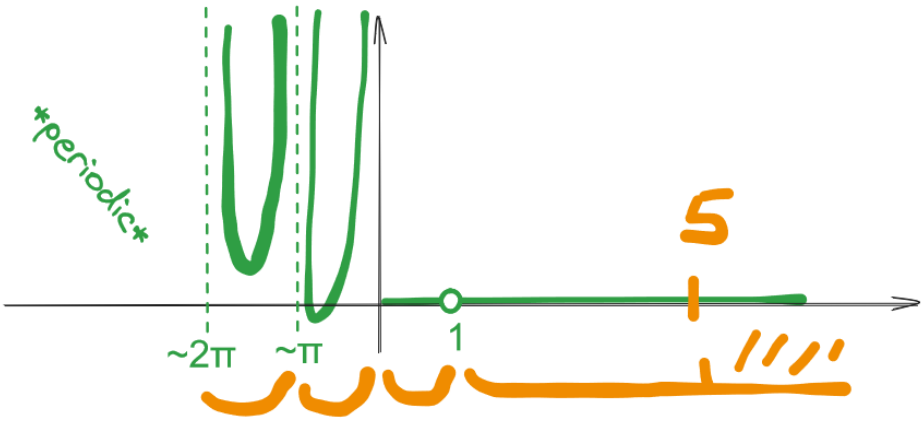

Рис. 1.4: Тестируемая область значений

2. 3. Тестирование функции

Тестирование производилось по схеме представленной на рисунке 1.5, где изображено как модули интегрируются в систему.

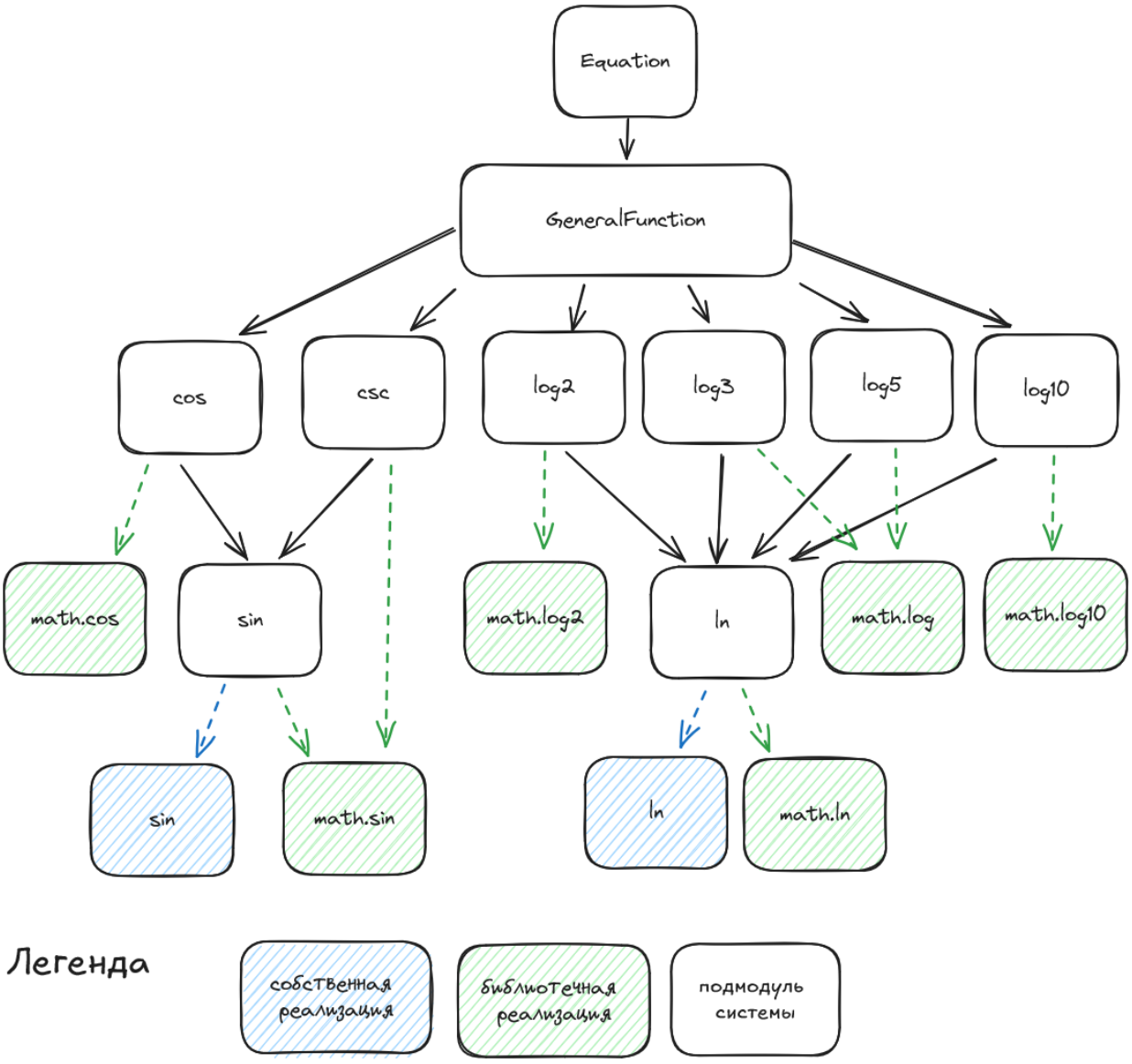

Рис. 1.5: Схема тестирования системы

В тестирующей программе производилась внедрение модулей с чередованием способов их реализации. Так, было создано 256 экземпляров класса GeneralFunction, значения которого на исследуемых промежутках были сохранены в файлы для дальнейшего анализа.

Исходные коды алгоритма представлены в репозитории, модуле `eqauation`: https://github.com/e1turin/itmo-sw-testing/blob/main/lab-2-integration-testing/equation/src/test/kotlin/io/github/e1turin/GeneralFunctionTest.kt

3. Анализ результатов

Анализ полученных данных производился с помощью kotlin-ноутбука со встроенными средствами визуализации, его можно найти по ссылке https://github.com/e1turin/itmo-sw-testing/blob/main/lab-2-integration-testing/equation/src/test/kotlin/io/github/e1turin/function-analysis.ipynb.

В качестве анализа строились графики отклонения значений разных реализаций системы от эталонной (со всеми библиотечными реализациями функций). Как можно было видеть, функции в целом имеют похожее значение отклонения: сильнее всего отклонение у тех функций, которые используют собственную реализацию $csc(x)$ (маска оканчивается на 0).

График можно видеть на рисунке 1.6.

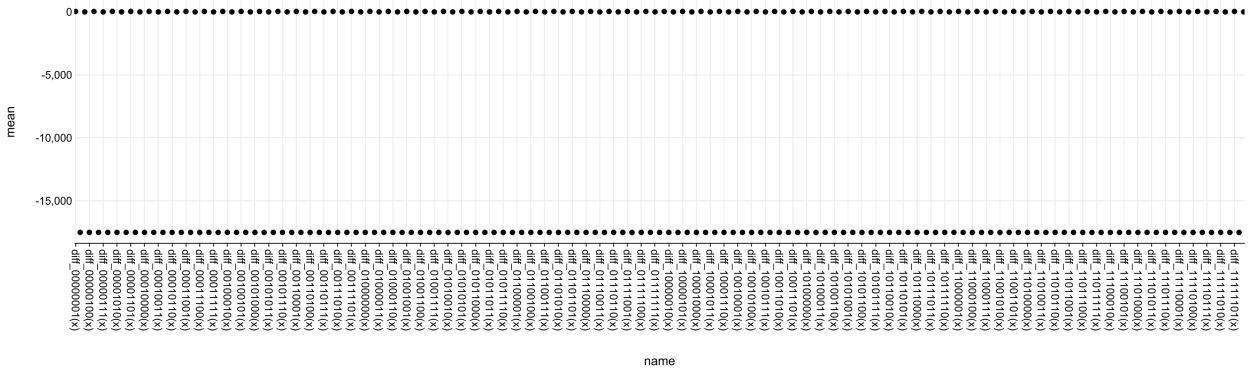

Рис. 1.6: График погрешности при интеграционном тестировании

4. Вывод

В соответствии с заданием был разработан программный код для интеграционного тестирования разработанных программных компонентов. Все исходные коды можно найти в репозитории: https://github.com/e1turin/itmo-sw-testing/. Во время выполнения лабораторной работы были укреплены навыки программирования, выработаны навыки написания интеграционных тестов и получен опыт работы с данными Kotlin-ноутбуке (он оказался сыроват для серьезного анализа).

Задание на мой взгляд плохо выражает суть интеграционного тестирования, более иллюстративный пример такого тестирования можно найти в программировании цифровой схемотехники: например, для задания этой лабораторной работы можно было бы взять разработанные модули на ЯП Chisel и провести интеграционное тестирование для них.