Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №2**

по «Тестированию программного обеспечения»

Выполнил:

Студент группы P33092

Абдурасул кызы Мээрим

Преподаватель:

Клименков Сергей Викторович

Санкт-Петербург

2024

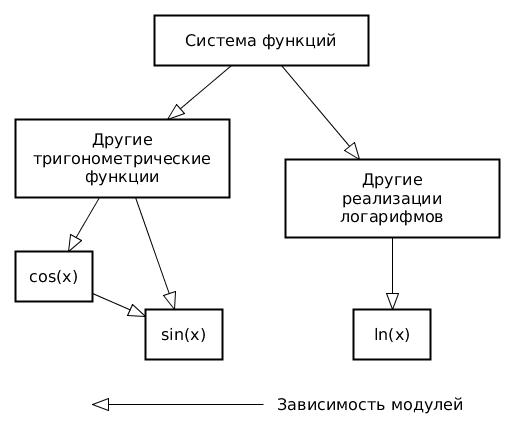
Вариант: 9222

Текст Задания:

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



**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 сsv файл вида «X, Результаты модуля (X)», позволяющее произвольно менять шаг наращивания Х. Разделитель в файле csv можно использовать произвольный.

Система функций:

A graph of a function

Description automatically generated

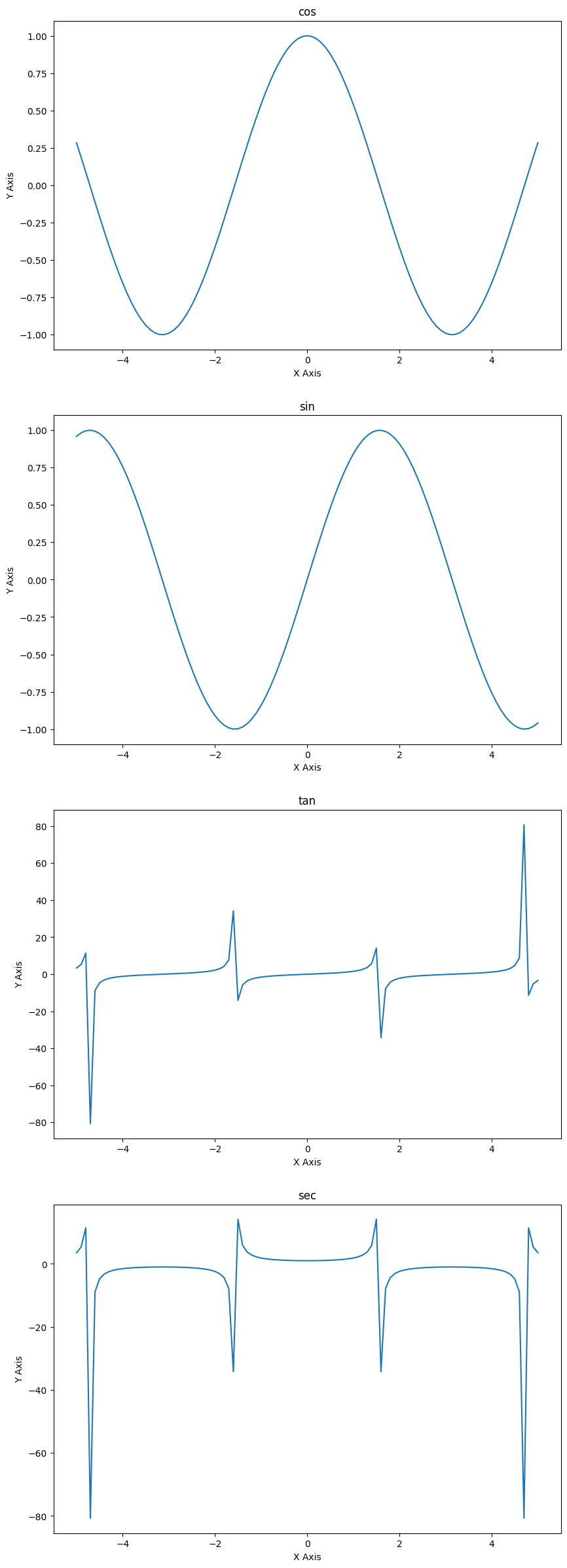
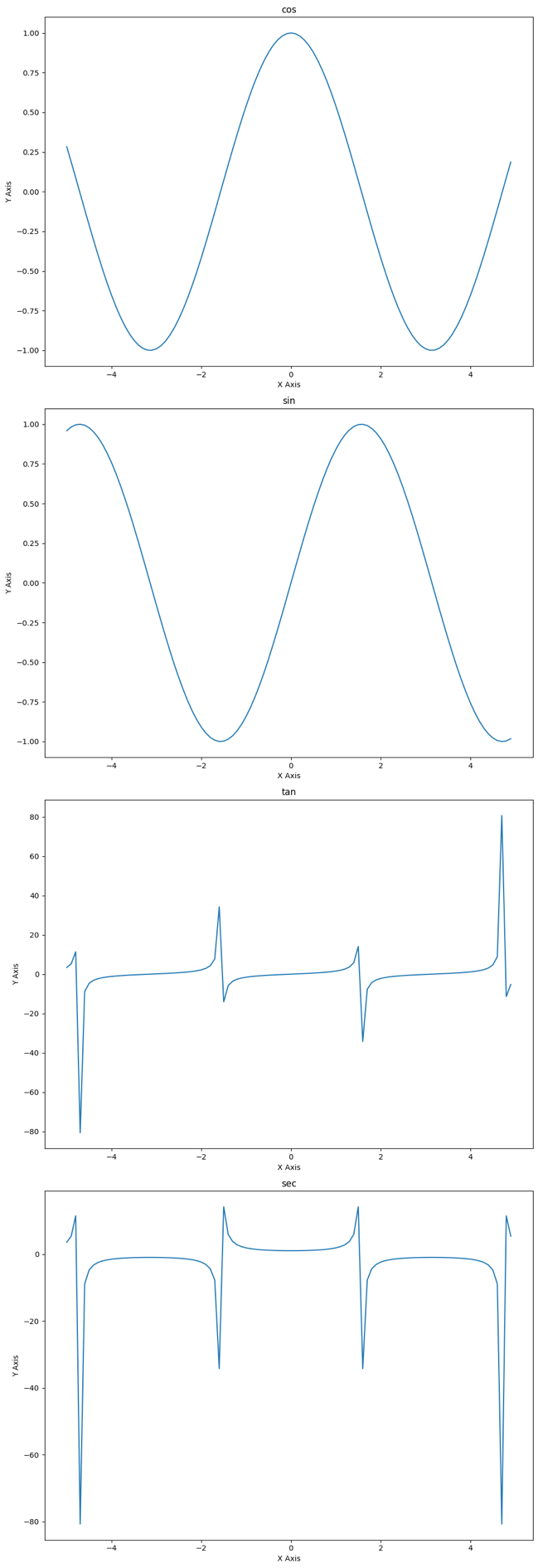
UML-диаграмма:

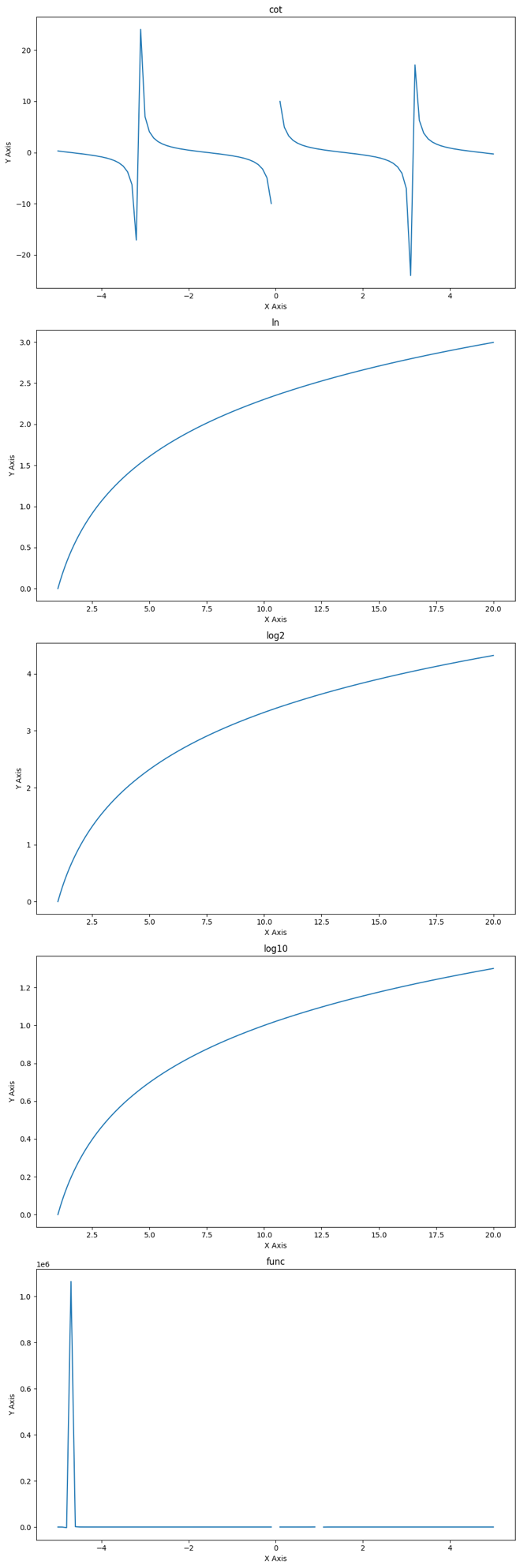
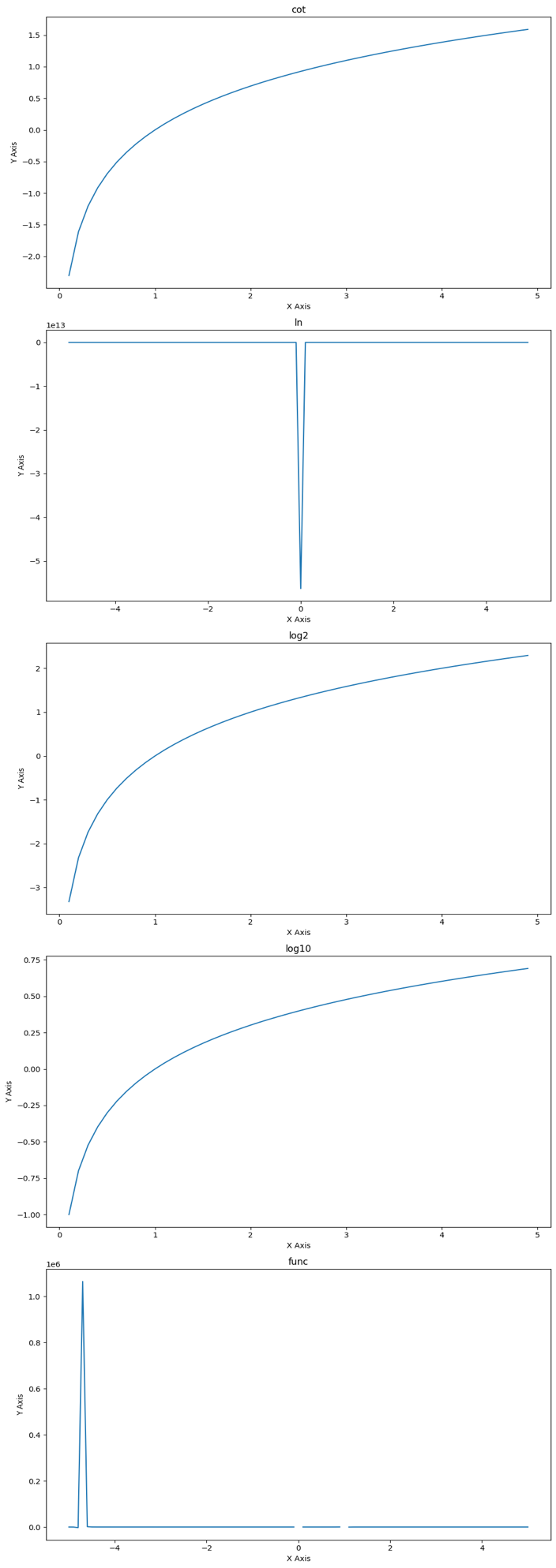
A diagram of function and function

Description automatically generated

Графики:

CSV-выгрузка: Функция NumPy:

Вывод:

Работа с Mockito и интеграционное тестирование представляют собой ключевые аспекты разработки программного обеспечения, позволяющие обеспечить высокое качество кода и надежную работу системы. Mockito обеспечивает управление зависимостями и создание заглушек для тестирования с внешними ресурсами, в то время как интеграционное тестирование проверяет правильность взаимодействия системы с внешними сервисами. Оба подхода, используемые вместе или по отдельности, помогают достичь максимального покрытия кода тестами и обеспечить его стабильность и надежность.