



Лабораторная работа №2

по дисциплине: Тестирование программного обеспечения

Вариант: 75123

Выполнил: Неграш Андрей, Р33301

Преподаватель: Гаврилов Антон Валерьевич

Санкт-Петербург, 2023

Задание

Лабораторная работа #2

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

Please, enter your variant:

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

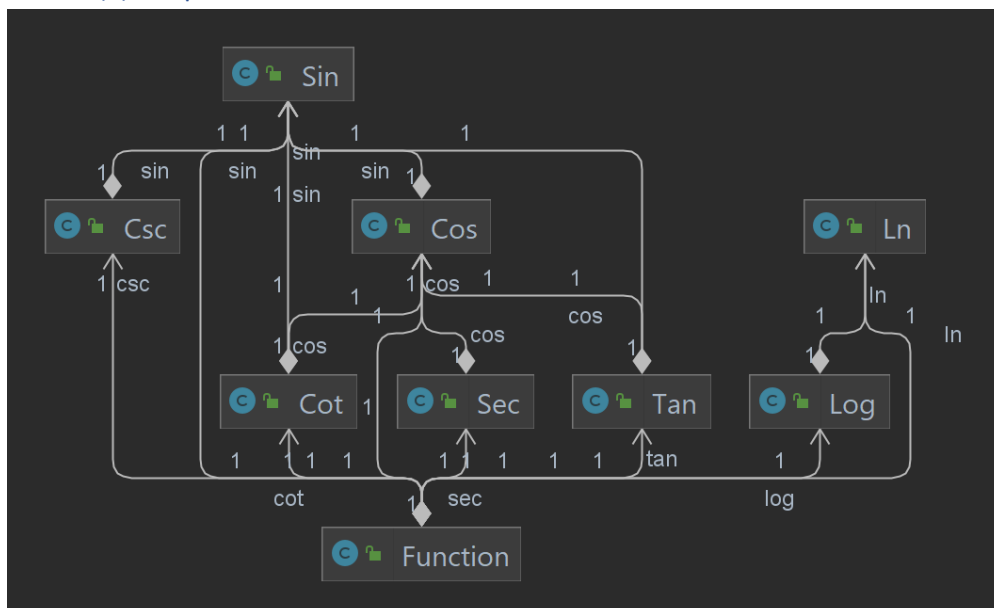
Правила выполнения работы:

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 $\sin(x)$):

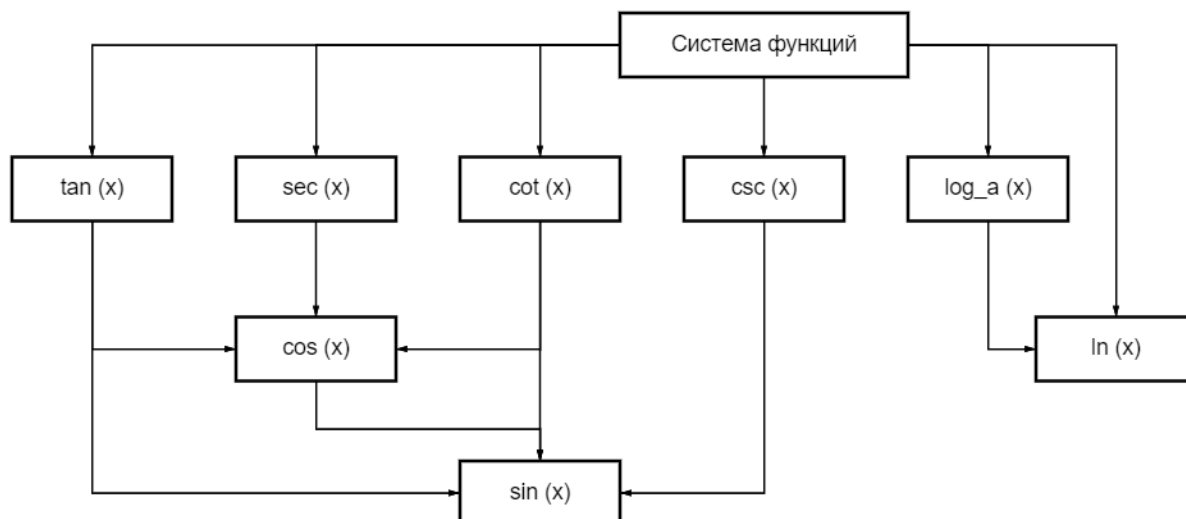


3. Обе "базовые" функции (в примере выше - $\sin(x)$ и $\ln(x)$) должны быть реализованы при помощи разложения в ряд с задаваемой погрешностью. Использовать тригонометрические / логарифмические преобразования для упрощения функций ЗАПРЕЩЕНО.
4. Для КАЖДОГО модуля должны быть реализованы табличные заглушки. 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», позволяющее произвольно менять шаг наращивания X. Разделитель в файле csv можно использовать произвольный.

UML-диаграмма



Описание тестового покрытия



Интеграция приложения проводилась сверху вниз, так как этот метод позволяет использовать заглушки, которые гораздо проще реализовать и в целом данный вид интеграции более прост в реализации.

Тестовое покрытие состоит из 2 частей:

- 1) Тесты для базовых функций и сравнение с эталоном из библиотеки Math

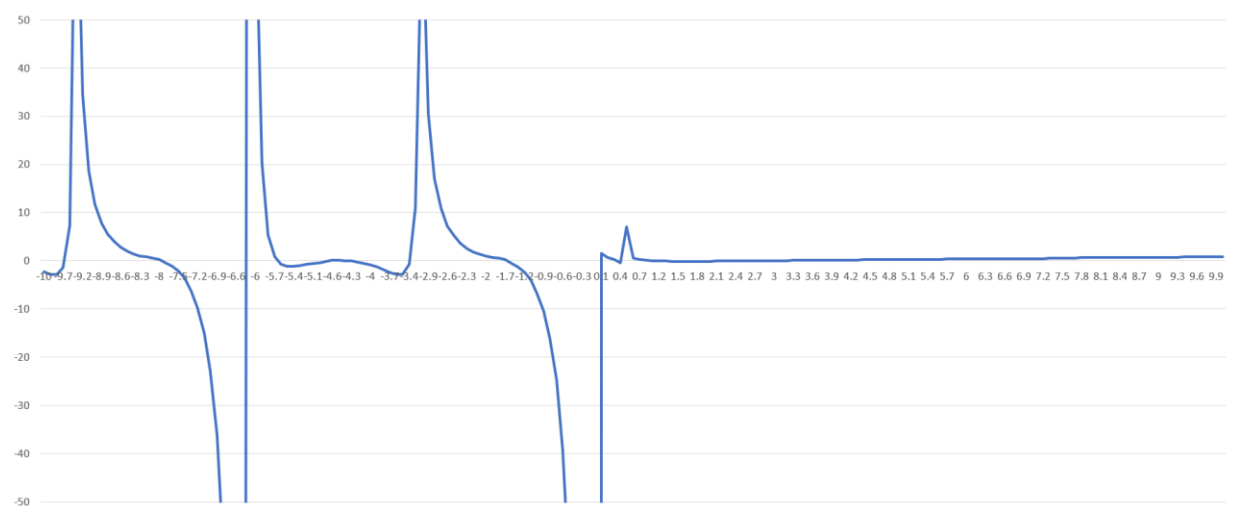
✓	✓	cosTest(double)	12 ms
	✓	[1] -10.0	1 ms
	✓	[2] -1.01	1 ms
	✓	[3] -1.0	1 ms
	✓	[4] -0.99	1 ms
	✓	[5] -0.01	1 ms
	✓	[6] 0.0	1 ms
	✓	[7] 10.0	
	✓	[8] 1.01	1 ms
	✓	[9] 1.0	1 ms
	✓	[10] 0.99	1 ms
	✓	[11] 0.01	
	✓	[12] NaN	1 ms
	✓	[13] Infinity	1 ms
	✓	[14] -Infinity	1 ms
>	✓	cotTest(double)	6 ms
>	✓	cscTest(double)	5 ms
>	✓	lnTest(double)	11 ms
>	✓	logTest(double)	14 ms
>	✓	secTest(double)	3 ms
>	✓	sinTest(double)	3 ms
>	✓	tanTest(double)	89 ms

2) Тесты для системы функций согласно заданию

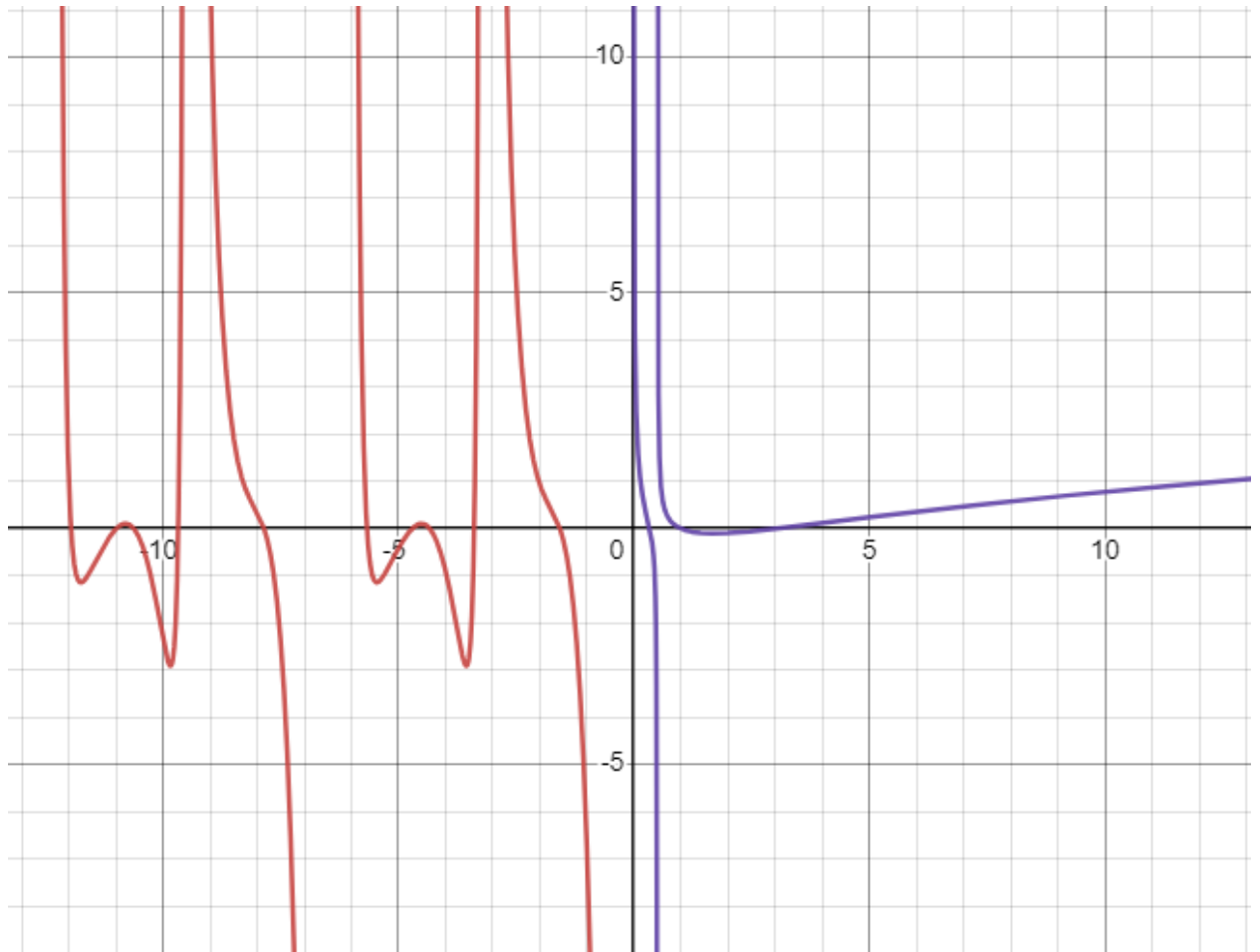
✓ testSystemWithMocks(double, double)	52 ms
✓ [1] 0, NaN	3 ms
✓ [2] -0.5235987, -56.04295	3 ms
✓ [3] -1.047198, -5.47606	3 ms
✓ [4] -1.570796, NaN	4 ms
✓ [5] -3.1415925, NaN	5 ms
✓ [6] -6.28318530718, NaN	3 ms
✓ [7] -Infinity, NaN	3 ms
✓ [8] 0.25, 0.357790	4 ms
✓ [9] 0.33333, 0	3 ms
✓ [10] 1, NaN	3 ms
✓ [11] 2, -0.09016	2 ms
✓ [12] 2.718281828, -0.030396	3 ms
✓ [13] 4, 0.11835	3 ms
✓ [14] 10, 0.78225	4 ms
✓ [15] 100, 4.32147	3 ms
✓ [16] Infinity, NaN	3 ms
> ✓ testWithCos(double, double)	60 ms
> ✓ testWithCosDeeper(double, double)	27 ms
> ✓ testWithCot(double, double)	40 ms
> ✓ testWithCotDeeper(double, double)	43 ms
> ✓ testWithCotDeeperAndCosDeeper(double, double)	62 ms
> ✓ testWithCsc(double, double)	38 ms
> ✓ testWithCscDeeper(double, double)	123 ms
> ✓ testWithLn(double, double)	27 ms
> ✓ testWithLog(double, double)	31 ms
> ✓ testWithLogDeeper(double, double)	41 ms
> ✓ testWithSec(double, double)	42 ms
> ✓ testWithSecDeeper(double, double)	54 ms
> ✓ testWithSin(double, double)	37 ms
> ✓ testWithSinAndLn(double, double)	33 ms
> ✓ testWithTan(double, double)	43 ms
> ✓ testWithTanDeeper(double, double)	29 ms
> ✓ testWithTanDeeperAndCosDeeper(double, double)	33 ms

Графики

График, полученный через csv-выгрузки



Эталонный график функции (<https://www.desmos.com/calculator/b2umm0us86>):



Исходный код

Можно посмотреть на моём GitHub: https://github.com/ANegrash/TPO_lab2

Вывод

Итак, в процессе выполнения данной лабораторной работы мною было выполнено интеграционное тестирование для разработанной программы, а также изучена работа классов заглушек на примере Mockito.