

Project Report
Software Engineering Process (SOEN 6011)

ETERNITY FUNCTIONS : F3

Mileshkumar Chandulal Kotadia (ID: 40156971)

August 04, 2022

Contents

1	Introduction	3
2	Code Repository	3
3	Problem 1	3
3.1	Function	3
3.2	Definition	3
3.3	Domain and Co-Domain	4
3.4	Characteristics	4
3.5	Context of Use Model	5
4	Problem 2	5
4.1	Functional Requirements	5
4.2	Non-Functional Requirements	6
4.3	Assumptions	6
5	Problem 3	7
5.1	Algorithm 1	7
5.2	Algorithm 2	9
6	Problem 4	10
6.1	Programming style	10
6.2	Debugger	11
6.3	Quality Attributes	12
6.4	CheckStyle	15
7	Problem 5	15
7.1	Junit standards	15
7.2	Traceability to Requirements	15
7.3	Traceability to Assumptions	16

1 Introduction

Hyperbolic functions are analogous to circular functions and trigonometric functions. It is usually possible to define hyperbolic functions by using algebraic expressions that include the exponential function (e^x) and its inverse exponential function (e^{-x}), where e represents Euler's constant. Here, we are going to discuss the sine hyperbolic functions, its properties, characteristics programming style and checkstyle used during development.

2 Code Repository

- URL : https://github.com/mileshkotadia987/SOEN_6011_Project.git
- Contributor : Mileshkumar Chandulal Kotadia (Student)
- Collaborator : Abdel Rahman Idrais Alsouqi (Teaching Assistant)
- Collaborator : Somayeh Davar (Teaching Assistant)
- Collaborator : Hamed Jafarpour (Teaching Assistant)

3 Problem 1

3.1 Function

- Mathematically function is defined as, $\sinh(x) = (e^x - e^{-x}) \div 2$.

3.2 Definition

- A hyperbolic sine function is an analogy to the circular Sin function that can be found throughout trigonometry[?]. When used with real numbers, it is defined by taking the area to be twice its axis, and tracing a line through the origin intersecting the unit hyperbola. This method is also used to solve ordinary differential equations of second order.

For the given function $\sinh(x) = (e^x - e^{-x}) \div 2$, e is the base of natural log . Approximate value of the e is 2.71828.

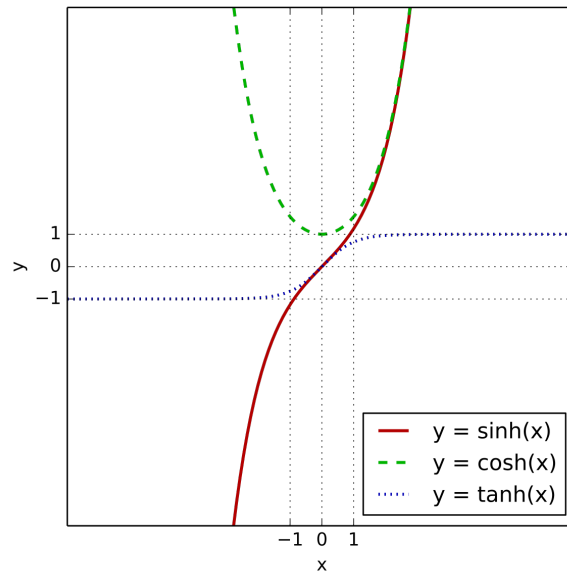


Figure 1: Graph of Hyperbolic Sine Function

3.3 Domain and Co-Domain

- Domain and Co-Domain of the hyperbolic sine function is \mathbb{R} .

$$\lim_{x \rightarrow -\infty} (\sinh(x)) = -\infty$$

$$\lim_{x \rightarrow \infty} (\sinh(x)) = \infty$$

3.4 Characteristics

- In terms of domain, the function is continuous, unbounded, and symmetric, namely odd, since $\sinh(-x) = -\sinh(x)$.
- $\sinh(x) \approx \cosh(x)$ for large x .
- $\sinh(x) \approx -\cosh(x)$ for large negative x .
- $\sinh(x)$ has period of $2\pi i$.
- The graph of $\sinh(x)$ is always between the graphs $e^x/2$ and $e^{-x}/2$.

3.5 Context of Use Model

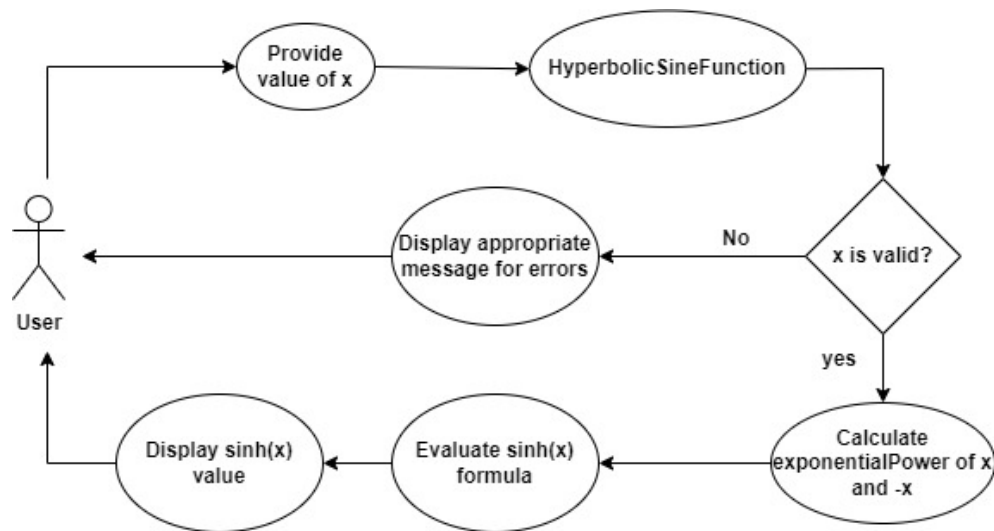


Figure 2: Context Diagram for Hyperbolic Sine Function

4 Problem 2

4.1 Functional Requirements

1. ID: FR1

Type: Functional

Owner: Miles

Difficulty: Easy

Description: When the user inputs a real number x , the system should calculate and display its hyperbolic sine value.

Rationale: To calculate $\sinh(x)$

2. ID: FR2

Type: Functional

Owner: Miles

Difficulty: Easy

Description: Whenever the user provides input other than a number, such as an alphabet or special character, then function should not return a value.

Rationale: when user provides invalid input

3. **ID:** FR3

Type: Functional

Owner: Miles

Difficulty: Easy

Description: Calculated exponential value e (Euler's number)[1] should be in a range of finite number.

Rationale: To calculate the exponential power of x .

4.2 Non-Functional Requirements

1. **ID:** NFR1

Type: Non-Functional

Owner: Miles

Difficulty: Easy

Description: The calculator should be easy for the user to use.

Rationale: Easiness of the application.

2. **ID:** NFR2

Type: Non-Functional

Owner: Miles

Difficulty: Easy

Description: System should display appropriate error messages for appropriate errors.

Rationale: The messages makes the work easier for the user.

3. **ID:** NFR2

Type: Non-Functional

Owner: Miles

Difficulty: Easy

Description: The calculator should provide accurate and precise answer.

Rationale: It helps the user to solve complex mathematical problems.

4.3 Assumptions

1. Assumption-1

Description: The domain of a hyperbolic sine function can be expressed mathematically as a set of all possible real numbers. In order to implement the function on the computer, Java will be used as a programming language,

so there are limitations on user input. The range of user input should be in between - 1.79769313486231570E+308 to + 1.79769313486231570E+308.

2. Assumption-2

Description: Due to the type of data used, there is a limit on the number of decimal points considered for the calculation if decimal numbers are input. The first 10 decimal points will be considered a significant decimal point.

3. Assumption-3

Description: Mathematically value of e is calculated as $e = 1/0! + 1/1! + 1/2! + 1/3! + \dots = 2.7182818284590452353602875\dots$. Since there is a limitation of datatype in java programming language, the infinite calculation is not possible[2]. Therefore, implementing the calculation of e will provide the value of $e = 2.7182815255731922$.

4. Assumption-4

Description: Because programming languages have restrictions on data types, there are also restrictions on output value. Output value cannot exceed what the datatype permits. The output will be constrained and less than the maximum permissible value of 1.79769313486231570E+308 and greater than the lowest allowed value even if the user enters the maximum allowed value (maximum that data type may carry) -1.79769313486231570E+308[3].

5 Problem 3

5.1 Algorithm 1

Description: For a given value of x , Algorithm 1 determines the hyperbolic sine function's value. There is no explicit calculation of the value of e in this algorithm. The below mentioned function EXPONENTIALPOWER() uses the Taylor series to determine the value of e power x [4].

We cannot calculate Taylor series that proceed indefinitely since the algorithm must end at some point. Due to this, the counter value is 10, which also meet above mentioned Assumption 3 and allows the algorithm to end with the highest precision possible up to 10 decimal places.

After computing the value of e^x and e^{-x} , value for the hyperbolic sine function is calculated.

Advantage:

- There is no recursion used in the algorithm which means that the execution stack will need less space.
- Calculation does not need any explicit function call. e

Disadvantage:

- The maximum accuracy of output is 10 decimal places.

Reason to select algorithm:

- No Recursion and Taylor Series Efficiency.

Pseudocode Algorithm 1:

Algorithm 1 Calculate $\sinh(x) = (e^x - e^{-x}) \div 2$

Require: $x \neq null$

$e^x \leftarrow \text{EXPONENTIALPOWER}(x)$
 $e^{-x} \leftarrow \text{EXPONENTIALPOWER}(-x)$
 $\sinh x \leftarrow (e^x - e^{-x}) \div 2$

function EXPONENTIALPOWER(x)
 $exponent \leftarrow x$
 $fractional \leftarrow exponent$
 $result \leftarrow 1 + exponent$
 $counter \leftarrow 1$
while $partial \neq result$ **do**
 $counter \leftarrow counter + 1$
 $fractional * \leftarrow exponent / counter$
 $partial \leftarrow result$
 $result + \leftarrow fractional$
end while
 return $result$
end function $= 0$

5.2 Algorithm 2

Description:

Algorithm 2 also determines the hyperbolic sine function's value. This algorithm directly calculates the value of e . With the use of the EXPPOWER function, the method utilises the value of e for computing e^x and e^{-x} . Since the algorithm must end at some point, the algorithm's counter value of 10 causes it to end with the greatest precision possible up to 10 decimal places. Hence, it helps in satisfying the above mentioned Assumption 3. Algorithm uses a Recursion functionality to calculate e^x and e^{-x} .

Advantage:

- Modularization becomes easy.
- The computation of e does not take x into account like Algorithm 1, making it simple to improve the approximation of value e as needed.

Disadvantage:

- Recursion is used to calculate exponential power, which takes up more execution space on the stack.
- To compute e , an explicit function call is needed.

Reason to select algorithm:

- adaptability in obtaining a greater approximation of e . This aids in getting precise results for $\sinh(x)$.

Pseudocode Algorithm 2:

Algorithm 2 Calculate $\sinh(x) = (e^x - e^{-x}) \div 2$

Require: $x \neq null$

$\sinh x \leftarrow (e^x - e^{-x}) \div 2$

$e \leftarrow \text{FINDE}()$

$e^x \leftarrow \text{EXPPOWER}(e, x)$

$e^{-x} \leftarrow \text{EXPPOWER}(e, -x)$

function FINDE()

$e \leftarrow 1$

$\text{decimalCount} \leftarrow 10$

while $\text{decimalCount} > 0$ **do**

$e = 1 + (e) \div \text{decimalCount};$

$\text{decimalCount} \leftarrow \text{decimalCount} - 1$

end while

return e

end function

function EXPPOWER(e, x)

if $x = 0$ **then**

return 1

end if

if $x \% 2 == 0$ **then**

return $\text{EXPPOWER}(e, x \div 2) * \text{EXPPOWER}(e, x \div 2)$

else

return $x * \text{EXPPOWER}(e, x \div 2) * \text{EXPPOWER}(e, x \div 2)$

end if

end function=0

6 Problem 4

6.1 Programming style

Style guides provide uniformity and predictability throughout your project[5]. We as a team had decided to use Google Programming style for java. Reasons to use Google Style guide :

- It makes reading code simpler.
- It improves predictability of file and variable names.
- Because of this, programmers may concentrate on logic rather than stylistic decisions.
- Using shortcuts, programmers may reformat code by importing the style guide into their IDE[6].

6.2 Debugger

When programming, a debugger aids in examining erratic behaviour. I used the IntelliJ IDE and IntelliJ Debugger to implement the $\sinh(x)$ function using Java.

Advantages of IntelliJ Debugger

- Analyze the use of memory
- support for remote Java application debugging
- assistance with conditional break points (stop debugger only when some condition occur)[7].
- Set up and customise breakpoints and watchpoints.
- Evaluate expressions
- Debugging sessions are interrupted and resumed

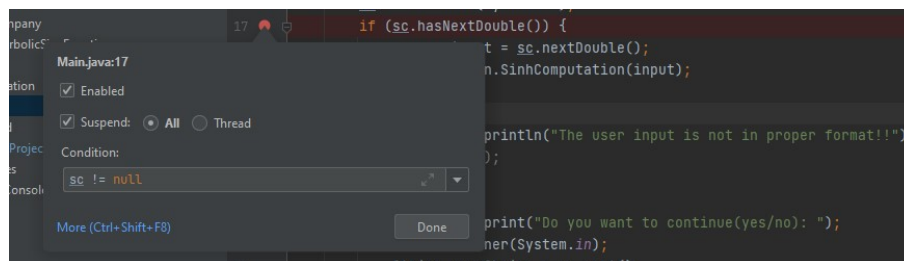


Figure 3: Support for conditional break points

Disadvantage of IntelliJ Debugger

- It is not feasible to use IntelliJ Debugger alone; it is a component of the IntelliJ IDE for Java.
- When compared to Eclipse IDE, IntelliJ takes greater system RAM to function[8]. High-end setups are required for the IntelliJ debugger.

6.3 Quality Attributes

1. Correctness

- Using Junit test cases, the accuracy of the implemented software is confirmed.
- In test cases, the expected result is calculated from <https://keisan.casio.com/exec/system/1223039747>, it is leading calculator manufacturing company [9].
- The computed value from the implemented function is compared to the expected value that was obtained and is asserted.

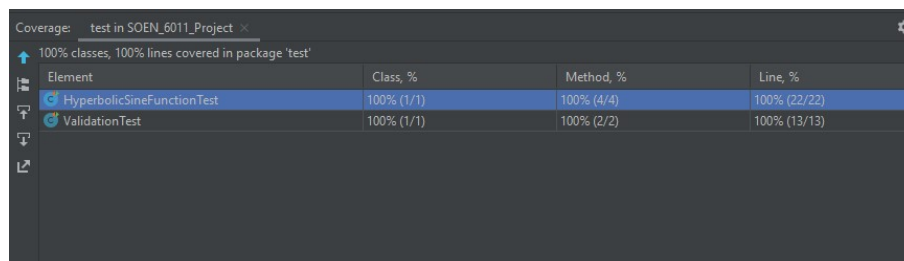


Figure 4: Code coverage for the Project

2. Efficiency

- No nested loops are used during implementation.
- There is no recursion in the implementation.
- It takes 74 ms (0.074 seconds) to complete a unit test suit (5 test cases) that runs all implemented functions.

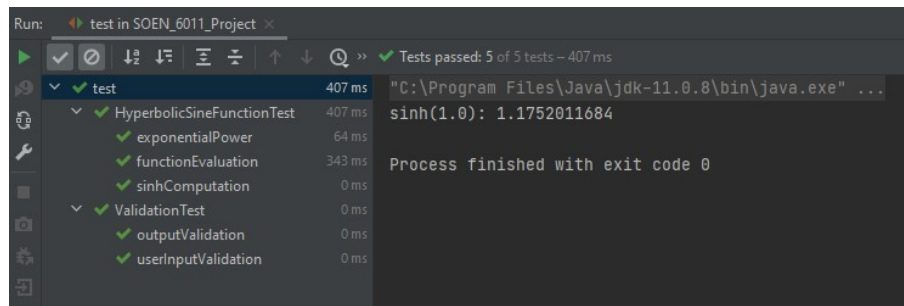


Figure 5: Junit test suit Execution time for all functions

3. Maintainable

- Modular architecture helps to ensure maintainability. For sine function computation and function validation, there is a separate class. For instance, all that has to be changed is the validation class if we wish to alter the criteria for validating user inputs and output.
- Comments are added with the proper details.

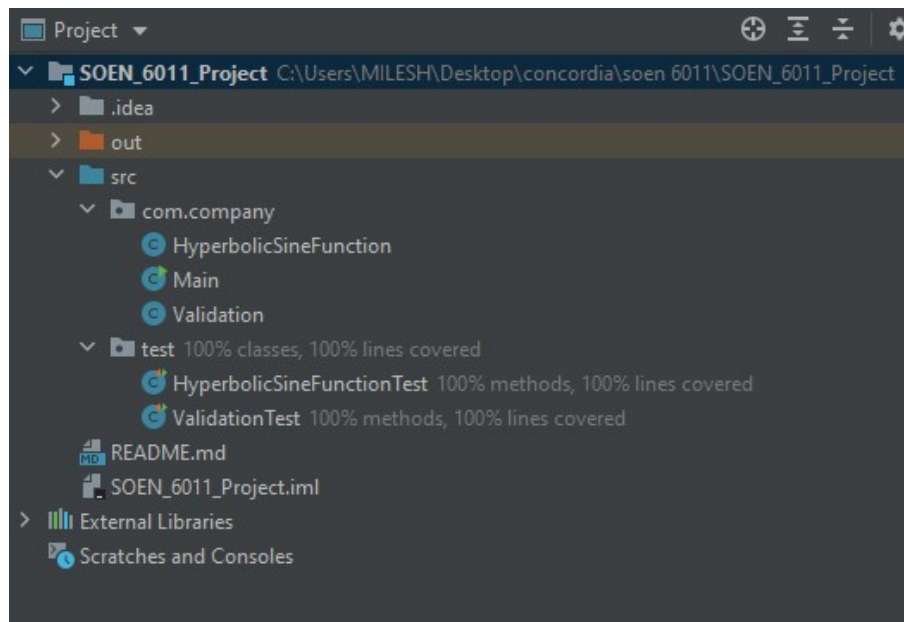


Figure 6: Project Modules

4. Robust

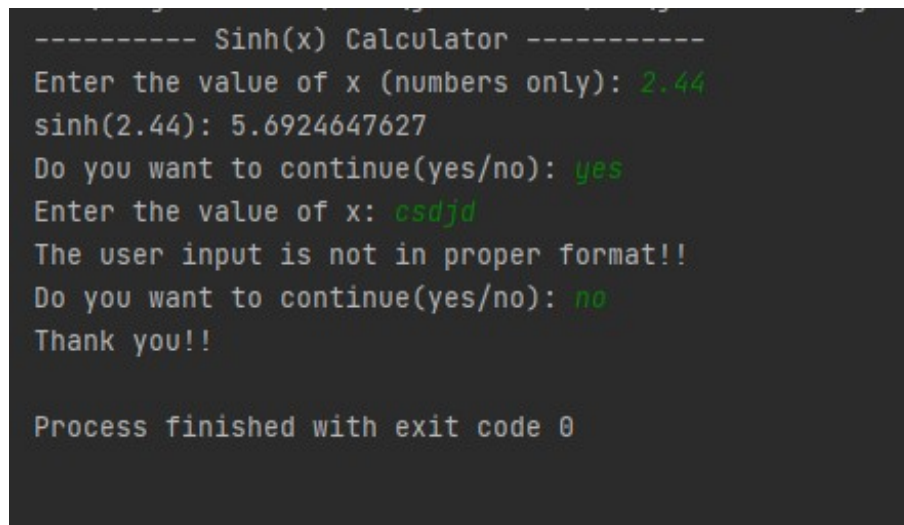
Robustness is achieved by ,

- Strict validation is performed on user input to ensure that only numeric values are entered.
- Prior to computation, the input is verified to ensure that the data type is supported.
- Verifying the output range, and displaying the appropriate error message to the user if the output exceeds what the data type can handle.

5. Usability

Usability is achieved with the help of,

- Providing a straightforward, user-friendly text UI.
- Displaying appropriate error messages as needed.
- Displaying appropriate directions for following up on the outcome.(Figure 7)
- Any system having a JVM may run an exported application as an executable jar file.



```
----- Sinh(x) Calculator -----
Enter the value of x (numbers only): 2.44
sinh(2.44): 5.6924647627
Do you want to continue(yes/no): yes
Enter the value of x: csdjd
The user input is not in proper format!!
Do you want to continue(yes/no): no
Thank you!!

Process finished with exit code 0
```

Figure 7: Screenshot of UI

6.4 CheckStyle

Programmers can use Checkstyle as a development tool to build Java code that follows a coding standard[10]. The sinh(x) function is implemented using Google Checkstyle.

Advantage

- Simple to set up for continuous integration. You may impose it on the project build process such that it fails if there is a checkstyle warning or error.
- Checkstyle makes code understandable by checking for layout and formatting errors.

Disadvantage

- There is no check that identifies superfluous type casts.
- No search for inactive public methods is performed.

7 Problem 5

7.1 Junit standards

One may better comprehend and appreciate the entire benefit of testing in software development with the aid of improved presentation and structure of the use of JUnit and testing. The Junit procedures listed in are applied wherever practical.

7.2 Traceability to Requirements

1. Requirement 1

JUnit test case(s) : userInputValidation() , functionEvaluation()

Description : These tests confirm that if the user enters a legitimate numeric value, the function will display the correct, accurate computed value.

2. Requirement 2

JUnit test case : userInputValidation()

Description : This test determines if the function displays the appropriate error message when the user enters an erroneous value (alphabet or special characters).

3. Requirement 3

JUnit test case : exponentialPower()

Description : This test determines if the computed value of e (Euler's number) is finite or not.

7.3 Traceability to Assumptions

1. Assumption 1

JUnit test case(s) : userInputValidation().

Description : when user inputs valid number value but more than what datatype can handle then function should show proper error message. This scenario is verified by given tests.

2. Assumption 2

JUnit test case(s) : sinhComputation().

Description : When a user enters a legitimate value but there are more than 10 decimal places, the function should only consider the first 10 places as meaningful decimal places.

3. Assumption 3

JUnit test case(s) : exponentialPower().

Description : By using the `epowern(double x)` function and supplying x as 1, this test validates the value of e .

4. Assumption 4

JUnit test case(s) : outputValidation().

Description : The function should return false if the user inputs a valid numeric number and the value is within the datatype's permitted range but the output is more or less than what the datatype can hold.

References

- [1] <https://reference.wolfram.com/language/ref/Sinh.html>.
- [2] <http://www.mathcentre.ac.uk/resources/workbooks/mathcentre/hyperbolicfunctions.pdf>.
- [3] https://www.analyzemath.com/DomainRange/domain_range_functions.html.
- [4] <http://functions.wolfram.com/ElementaryFunctions/Sinh/04/>.
- [5] <https://www.codereadability.com/why-use-a-style-guide/>.
- [6] <https://github.com/hpi-information-systems/metanome/wiki/installing-the-google-styleguide-settings-in-intellij-and-eclipse>.
- [7] <https://raygun.com/blog/java-debugging-tools/>.
- [8] <https://blog.gceasy.io/2018/11/05/memory-efficient-eclipse-or-intellij/>.
- [9] https://world.casio.com/media/company/files/casio2017_all_en.pdf.
- [10] <https://checkstyle.sourceforge.io/writingchecks.html>.