Concordia University

# TRIGONOMETRIC FUNCTION: tan(x)

August 13, 2021

Deliverable 1

**Arnab Roy**

Student Id: 40184043

Team: E

# Contents


**1 Introduction** **1**
- 1.1 Domain and co-domain . . . 1
- 1.2 Interesting properties of the graph of the tangent function . . . 2

**2 Requirements and assumptions for Tangent function** **2**
- 2.1 Assumptions . . . 2
- 2.2 Requirements . . . 3

**3 Algorithms** **5**
- 3.1 Pseudocode . . . 5
  - 3.1.1 The algorithm for sin(x)/cos(x) . . . 5
  - 3.1.2 The algorithm for the Taylor series of tangent function . . . 7
- 3.2 Description of the implemented algorithm . . . 7
  - 3.2.1 Time complexity . . . 7
  - 3.2.2 Space complexity . . . 7
- 3.3 The advantages and disadvantages . . . 7
  - 3.3.1 Disadvantage of Taylor series of tangent . . . 7
  - 3.3.2 Advantage of sin(x)/cos(x) . . . 8

**4 Information about the debugger used** **8**
- 4.1 Advantages . . . 8
- 4.2 Disadvantages . . . 9

**5 Effort to make the program better** **9**
- 5.1 Correctness . . . 9
- 5.2 Efficiency . . . 9
- 5.3 Maintainability . . . 10
- 5.4 Robustness . . . 10
- 5.5 Usability . . . 10

**6 Unit Testing Guidelines** **10**

**7 Description of Checkstyle** **10**

**8 Code review of $\Gamma(x)$** **11**
- 8.1 Purpose . . . 11
- 8.2 Guidelines . . . 11
- 8.3 Approach . . . 11
- 8.4 Results of review . . . 12
- 8.5 Conclusion . . . 12

**9 Testing the ab$^x$ function** **12**

# 1 Introduction

The tangent function is one of the six trigonometric functions. It has many important uses in real life calculations such as calculating the slope of straight lines, angles of elevation and depression [1], rate of altitude change of an aircraft [2] etc.
The tangent function can be understood from an unit circle(a circle whose radius = 1).

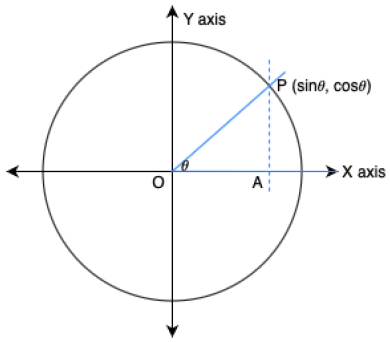


**Figure 1:** A unit circle

In a unit circle we take two lines originating from the center, one along the positive x axis and the other line intersecting the circumference of the circle. From the definition of unit circle, the coordinate of the point at which the circumference is intersected is $(\sin\theta, \cos\theta)$ [3] and the tangent of the angle $\theta$ is:

$$\tan(\theta) = \sin(\theta) \div \cos(\theta) \tag{1}$$

## 1.1 Domain and co-domain

The domain of $\tan\theta$ is x $\in \mathbb{R}$, x $\neq (\pi/2) + n^*\pi$ and its codomain is (-$\infty$, $\infty$) [4].

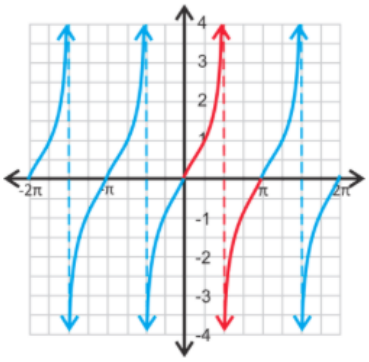


**Figure 2:** Graph of tangent function

### 1.2 Interesting properties of the graph of the tangent function

The graph of the tangent function has some interesting properties. As $\cos\theta = 0$ when $\theta = \pm(n\pi/2)$ where n is odd, the graph approaches an asymptote along y-axis as $\cos\theta$ approaches 0. Moreover, the graph has no amplitude and the period is calculated as the distance between any two high points or low points at the same height and the period is $\pi$. The graph intersects the y-axis only at (0,0) and the x-axis for $\theta = n\pi$.

## 2 Requirements and assumptions for Tangent function

The requirements and assumptions of the function tan(x) follow ISO/IEC/IEEE 29148 standards. A priority of 1 means most important and a priority of 5 means least important.

### 2.1 Assumptions

| **Assumption ID: F2-ASSUMP1** | |
|---|---|
| **Version:** | 1.0 |
| **Type:** | Functional |
| **Owner:** | Arnab Roy |
| **Priority:** | 1 |
| **Difficulty:** | Easy |
| **Description:** | The user shall provide input in integers |
| **Rationale:** | To keep the implementation simple and generate accurate outputs with a simple algorithm |

| **Assumption ID: F2-ASSUMP2** | |
|---|---|
| **Version:** | 1.0 |
| **Type:** | Functional |
| **Owner:** | Arnab Roy |
| **Priority:** | 1 |
| **Difficulty:** | Easy |
| **Description:** | The user shall provide input in range $-2^{31}$ to $2^{31}-1$ |
| **Rationale:** | The range for the `int` primitive type is from $-2^{31}$ to $2^{31}-1$ and to keep inputs small for a simple implementation |

| Assumption ID: F2-ASSUMP3 | |
|---|---|
| **Version:** | 1.0 |
| **Type:** | Non-functional |
| **Owner:** | Arnab Roy |
| **Priority:** | 3 |
| **Difficulty:** | Easy |
| **Description:** | The factorials for the calculation may be pre-stored in an array |
| **Rationale:** | To speed up the calculation of the output and execution of the algorithm |

## 2.2 Requirements

| Requirement ID: F2-REQ1 | |
|---|---|
| **Version:** | 1.0 |
| **Type:** | Functional |
| **Owner:** | Arnab Roy |
| **Priority:** | 1 |
| **Difficulty:** | Easy |
| **Description:** | When the user enters a value for which cos(x)=0, an exception shall be thrown with an useful error message |
| **Rationale:** | Tangent function does not have a valid output for angle=0 and the user shall be let known about this |

| Requirement ID: F2-REQ2 | |
|---|---|
| **Version:** | 1.0 |
| **Type:** | Functional |
| **Owner:** | Arnab Roy |
| **Priority:** | 1 |
| **Difficulty:** | Easy |
| **Description:** | If the output is -0.0, the function shall remove any signs and return 0.0 |
| **Rationale:** | Using a sign with 0 is redundant and might be confusing to the user |

| Requirement ID: F2-REQ3 | |
|---|---|
| **Version:** | 1.0 |
| **Type:** | Non-functional |
| **Owner:** | Arnab Roy |
| **Priority:** | 3 |
| **Difficulty:** | Medium |
| **Description:** | The time complexity of the program should be in O(n) |
| **Rationale:** | To keep program execution fast and within 1 second |

| **Requirement ID: F2-REQ4** | |
|---|---|
| **Version:** | 1.0 |
| **Type:** | Non-functional |
| **Owner:** | Arnab Roy |
| **Priority:** | 1 |
| **Difficulty:** | Hard |
| **Description:** | All angles above 90 degree shall be converted within 90 degrees |
| **Rationale:** | Calculation of smaller values is faster and all angles greater than 90 degrees can be converted within 90 degrees, so it is unnecessary to calculate higher values |

| **Requirement ID: F2-REQ5** | |
|---|---|
| **Version:** | 1.0 |
| **Type:** | Functional |
| **Owner:** | Arnab Roy |
| **Priority:** | 1 |
| **Difficulty:** | Easy |
| **Description:** | The output of the function shall be a double |
| **Rationale:** | Most outputs of tangents are in decimal and removing the fractional part would make the output inaccurate |

| **Requirement ID: F2-REQ6** | |
|---|---|
| **Version:** | 1.0 |
| **Type:** | Non-functional |
| **Owner:** | Arnab Roy |
| **Priority:** | 1 |
| **Difficulty:** | Medium |
| **Description:** | The output of the tangent function shall be accurate upto 6 digits after the decimal |
| **Rationale:** | Implementation of a simple algorithm has the trade-off of accuracy |

| **Requirement ID: F2-REQ7** | |
|---|---|
| **Version:** | 1.0 |
| **Type:** | Non-functional |
| **Owner:** | Arnab Roy |
| **Priority:** | 1 |
| **Difficulty:** | Medium |
| **Description:** | The output shall be rounded to a value of 7 digits after the decimal |
| **Rationale:** | Since the algorithm is accurate upto 6 decimal places and sometimes 7, the output would be rounded to 7 digits after the decimal |

# 3 Algorithms

## 3.1 Pseudocode

### 3.1.1 The algorithm for sin(x)/cos(x)

**Algorithm 1** tangent(angle) using sin(angle)/cos(angle)

```
result = calculateSin(angle)/calculateCos(angle)
if result is equal to ∞ then
    throw error
end if
Round result to 7 digits
return result
```

**Algorithm 2** calculateSin(angle)

```
convert angle to 360 degrees
if angle is less than 0 then
    store the sign of angle
    angle = absolute value of angle
end if
determine the quadrant of the angle
signInQuadrant = calculate the sign of the angle in determined quadrant
shouldSubtract = true
angleInRadian = angle * 3.141592654 / 180
squareOfRadianAngle = angleInRadian * angleInRadian
numerator = angleInRadian * angleInRadian * angleInRadian
digitToFactorial = 3
result = angleInRadian
for i = 1 . . . 9 do
    nTerm = numerator / factorials[digitToFactorial]
    if shouldSubtract then
        nTerm *= -1
        shouldSubtract = false
        else
        shouldSubtract = true
    end if
    result += nTerm
    numerator = numerator * squareOfRadianAngle digitToFactorial += 2
end for
result = roundTo7Digits(result)
return result == 0 ? 0 : result * sign * signInQuadrant
```

**Algorithm 3** calculateCos(angle)

```
1: convert angle to 360 degrees
2: if angle is less than 0 then
3:     angle = absolute value of angle
4: end if
5: determine the quadrant of the angle
6: signInQuadrant = calculate the sign of the angle in determined quadrant
7: shouldSubtract = true
8: angleInRadian = angle * 3.141592654 / 180
9: squareOfRadianAngle = angleInRadian * angleInRadian
10: numerator = squareOfRadianAngle
11: digitToFactorial = 2
12: result = 1
13: for i = 1 . . . 9 do
14:     nTerm = numerator / factorials[digitToFactorial]
15:     if shouldSubtract then
16:         nTerm *= -1
17:         shouldSubtract = false
18:
19:     else
20:         shouldSubtract = true
21:     end if
22:     result += nTerm
23:     numerator = numerator * squareOfRadianAngle
24:     digitToFactorial += 2
25: end for
26: result = roundTo7Digits(result)
27: return result == 0 ? 0 : result * signInQuadrant
```

### 3.1.2 The algorithm for the Taylor series of tangent function

**Algorithm 4** tangent(angle) using Taylor series

```
1: nominator[13] = [1, 1, 2, 17, 62, 1382, 21844, 929569, 6404582, 443861162,
   18888466084, 113927491862, 58870668456604]
2: denominator[13] = [1, 3, 15, 315, 2835, 155925, 6081075, 638512875, 10854718875,
   1856156927625, 194896477400625, 49308808782358125, 3698160658676859375]
3: result = 0
4: squareOfAngle = angle * angle
5: for test = 0 ... 12 do
6:     result += angle * nominator[test] / denominator[test]
7:     angle *= squareOfAngle
8: end for
9: return result
```

## 3.2 Description of the implemented algorithm

We have implemented the tangent function using the sin/cos formula. For this, we need the Taylor series for sine and cosine. At first, the angle in degrees provided by the user is converted within 90 degrees and the quadrant is determined. The it is converted to radians. The algorithm considers upto 9 terms and achieves an accuracy upto 6 decimal digits.

### 3.2.1 Time complexity

O(N)

### 3.2.2 Space complexity

O(T), where T is the number of factorials generated.

## 3.3 The advantages and disadvantages

We have implemented the tangent(angle) using the sin(angle)/cos(angle) algorithm. It has several advantages of the Taylor series of the tangent function. To understand this, we will first discuss about the disadvantage of the Taylor series of tangent.

### 3.3.1 Disadvantage of Taylor series of tangent

- The algorithm is not accurate for smaller terms. A large number of terms (approximately 30) needs to be taken to get an accurate result.
- The denominator of the 13th term of the series is 3698160658676859375. It is evident that the denominators of the larger terms will exceed the capacity of the primitive data types in Java.

- Keeping the Taylor series small results in output whose fractional parts are largely deviated from the accurate result.

### 3.3.2 Advantage of sin(x)/cos(x)

- Needs only 9 terms for the Taylor series of sine and cosine to make the output accurate upto 6 fractional parts
- Smaller number of terms mean faster execution

# 4 Information about the debugger used

For the purpose of debugging the method written for calculating the tangent of an angle, the Intellij IDEA debugger was used. This debugger comes pre-bundled with the Intellij IDEA IDE. This makes coding and debugging streamlined.

## 4.1 Advantages

Using a pre-bundled IDE comes with many advantages, one of which is ease of debugging when writing code. Some of the capabilities that this debugger provides is:

- Shows the variables and their updated values for each line of code. Values are shown in the code inline for each variable and also all variables and values are shown in the variables window.
- Allows jumping to variable, variable type and method declaration.
- Allows changing variable values during debugging, change program flow and execute conditional code branches.
- Allows evaluation of complex code expressions on the fly.
- Enables pausing the debugging procedure and check the call stack and the running threads.
- Enables setting breakpoints on lines. The debugger stops code execution at the breakpoint and the user can manually execute the remaining code statements, one by one.
- Allows dragging a breakpoint and dropping it on another line to move a breakpoint from one line to another. Also allows setting multiple breakpoints and disabling them.
- Enables setting conditional breakpoints, where the application is stopped at that line if a certain condition is met.

- Allows stepping over a method, stepping into a method or stepping out of a method during debugging. JDK methods are stepped over by default and the focus is towards user written methods and statements.
- Enables "run to cursor" feature, where the debugger does not stop at lines until the one at which the mouse cursor is at.

### 4.2 Disadvantages

The debugger is very resource intensive. It can easily slow down a machine with 8GB RAM and Intel Core i7 processor for very large programs.

## 5 Effort to make the program better

When writing the program, focus was given to make it correct, efficient, maintainable, robust, and usable. An explanation of the steps taken to make them such is given below.

### 5.1 Correctness

According to the requirements, the program needs to be correct up-to 6 decimal digits of a Casio scientific calculator. The result of the tangent function in the program depends on the results of the sine and the cosine functions. These functions were implemented using the Taylor series and severals steps were taken to increase their accuracy.

- At first, only 3 terms from the Taylor series were considered, which later was increased to 9. The inclusion of more terms increased the accuracy but was not enough.
- It was observed that for smaller angle values, the calculations were very accurate. But for higher values, the results started to deviate. So, all angles were first converted within 360 degrees.
- To further increase the accuracy, they were converted to 90 degrees and the quadrant was calculated to determine the sign (+ or -) of the result. This also eliminated a problem during calculation of cosine of 180 which outputted 1.0000008 instead of 1.
- For many calculations, Java shows -0. These results were checked so that the output for 0 does not have any sign.

### 5.2 Efficiency

To calculate the tangent of an angle, the Taylor series algorithm was implemented which is a $O(n^2)$ algorithm because of the calculation of the factorials for each term. To make it efficient, the values of the factorials were calculated beforehand at once and stored in an array. To calculate the factorial of n, n was multiplied with the value of (n-1)th index in the array and the result was stored at nth index of the array. This algorithm

takes place in O(n) time and the values can be accessed in O(1) time. Thus the run time complexity of the algorithm was reduced to O(n) and the space complexity increased to O(n). The rounding function runs in O(1) time complexity.

### 5.3 Maintainability

To make the program maintainable, the lines of code were properly documented by comments. Most code, which would be repeatable and reusable, were separated into separate small functions. The methods return appropriate error messages. The code was written in a way so that it is easier to read and meaningful variable names were used.

### 5.4 Robustness

For the tangent function, one pitfall is when the value of cosine is 90 degrees. For this case, an `ArithmeticException` was thrown with an error message that could be clearly understood by the user.

### 5.5 Usability

The functions written for the tangent, cosine and sine functions do not have input prompts themselves. They require driver programs to take the input from the user and pass it as arguments. The output and error messages from the tangent function notifies the user about the result as a clearly understandable string.

## 6 Unit Testing Guidelines

- Each test case has a small description
- Test case have been designed mapping them to the functional requirements
- Test cases have been assigned unique IDs
- Each test have the name of the method that are tested
- Test cases are elaborate
- All corner cases have been tested

## 7 Description of Checkstyle

For this project, an extension of Checkstyle for IntelliJ IDEA had been used. The source code was analyzed with the Google style. It generated 121 warnings, most of which were related to code indentation. It suggested a tab width of 2 spaces instead of 4. This type of indentation makes the code clearer to read. Moreover, another type of warning was to comment before classes and methods using JavaDoc. It also warned about comments

that are more than 100 words long on a single line.
The advantages of using CheckStyle is that it enforces strict coding conventions which is useful across teams. The code analysis was also very fast.
One disadvantage of the extension was that, it did not show the errors inline, like most static source code analysers do.

# 8 Code review of $\Gamma(x)$

## 8.1 Purpose

The purpose of this review is to check whether the quality of code in the module is degrading the quality of the system, i.e the Calculator app. Some standard guidelines have been followed to review the code.

## 8.2 Guidelines

For this code review, the Google developer guidelines which is available at https://google.github.io/eng-practices/review/reviewer/looking-for.html have been followed. The guidelines are as follows:

- Check whether the code components interact well with other code components.
- The code functionality satisfies the users
- Outputs from error messages are well written and articulated
- Naming conventions for functions, variables, classes are good and communicate well to the user
- The code has unit tests that are maintainable and not complex
- The code has well written comments that add value and are not too lengthy
- The code follows a style guide
- The code is not over engineered
- No unnecessary import statements have been used

## 8.3 Approach

Several approaches were followed when reviewing the code and they are as follows:

- The code was read through and it was identified whether the naming conventions were good and whether the comments were useful and did not exceed 100 characters per line. Effort was given to check whether any function war too large and whether it needed breaking down into smaller functions.

- The code was run using a debugger line by line. The return values and error messages were checked against the requirements.
- The test cases were run to check if all had passed. It was also checked whether the test cases had clearly understandable assertions.
- It was checked whether the code followed a consistent style.

### 8.4 Results of review

| Attribute | Result |
|---|---|
| **Code component interaction:** | PASS |
| **Correct functionality:** | PASS |
| **Clear error messages:** | PASS |
| **Naming convention:** | PASS |
| **Unit tests:** | PASS |
| **Useful comments:** | PASS |
| **Style guide:** | PASS |
| **Code base not too complex:** | PASS |
| **No unnecessary imports:** | PASS |

### 8.5 Conclusion

The code-base of this module had all the necessary quality attributes. My only suggestion would be if in some small places, the coding styles would not be neglected, for example for the spacing around curly braces.

## 9 Testing the $ab^x$ function

The author of the function provided all the unit tests which were done using the JUnit library. Since we had a difference in the development environment, at first I was facing some issues with the JUnit library configuration. After resetting the configurations, JUnit worked correctly.
At first I individually ran the test cases for various types of input. They all passed. Finally, I ran the test class all together and that passed as well.
The IDE I was using was IntelliJ IDEA and I had JUnit 5 installed along with Juniper as dependencies. The OS I am using is MacOS 11.5.

## References

[1] "Tangents and slopes." https://www2.clarku.edu/faculty/djoyce/trig/tangents.html. Accessed: 2021-07-22.

[2] https://math.stackexchange.com/questions/755083/real-world-tangent-functions. Accessed: 2021-07-22.

[3] https://en.wikipedia.org/wiki/Unit_circle. Accessed: 2021-07-22.

[4] https://www.varsitytutors.com/hotmath/hotmath_help/topics/graphing-tangent-function. Accessed: 2021-07-22.