

# CONCORDIA UNIVERSITY

SOEN 6011 - SOFTWARE ENGINEERING PROCESS

---

## ETERNITY - FUNCTION ARCCOS(X)

---

PRAVALLIKA BASAM

STUDENT ID : 40200923

<https://github.com/pravallikabasam/eternity-soen6011>

# 1 Introduction

The inverse trigonometric functions such as  $\sin^{-1}$ ,  $\cos^{-1}$ ,  $\tan^{-1}$  are used to find the unknown measure of an angle of a right triangle when two side lengths are known. The inverse functions are denoted by the prefix "arc". In this section, We will discuss the arccos function, also referred to as the Inverse Cosine function. All trigonometric functions are typical of the  $\Theta$  radian when expressed in radians. However, the inverse functions lie under the radian of  $(r\Theta)$ . Trigonometric functions typically have a one-to-one relationship, whereas all inverse functions have a one-to-many relationship. As a result, the ranges of the arccos function are usually the proper subsets of domains of original trigonometric functions.

## 1.1 Domain and range of $\cos^{-1}$

The relation between the Inverse and the original Cosine function can be better explained using the following example:

If we were to calculate the value of

$$\cos^{-1}(-1/2)$$

The statement previously will translate to:

$$\begin{aligned} \arccos(-1/2) &= \theta \\ \cos \theta &= -1/2 \end{aligned}$$

After solving the angles on the unit ellipse, The value of  $\arccos(-1/2)=120^\circ$ .

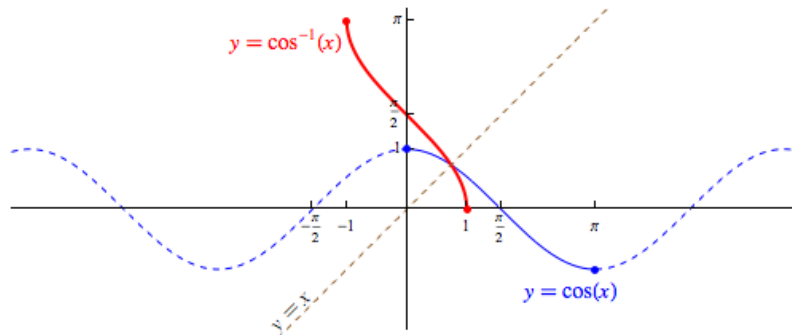


Figure 1: Graph for Cos and ArcCos Functions

## 1.2 Characteristics of inverse function

1. The inverse function  $\cos^{-1} x$  has a 3 branch points:  $x=\pm 1$  ,  $x=\infty$
2. It is a single value function and continuous on the interval  $(-\infty,1]$  and from below on the interval  $[1,\infty]$ .
3. Function is neither even nor odd.
4. It is an analytic function of  $x$  and can be defined over complex numbers.

## 2 Requirements

This document is intended to represent the functional requirements for the  $y = \cos^{-1}$  function. Functional requirements shall be indicated by the letter "FR" for the sake of simplicity, and non-functional requirements by the letter "NFR". All requirements will be met with version 1.0. Following are the requirements needed to perform the function:

### 2.1 Functional Requirements

- ID: FR1  
Owner: Pravallika Basam  
Version: 1.0  
Description: If the value of  $x$  is greater than 1 or less than -1. System should display an error.  
Rationale: arccos function is defined on the domain  $[-1,1]$
- ID: FR2  
Owner: Pravallika Basam  
Version: 1.0  
Description: If the function has an inverse that is also a function, then there can only be one  $y$  for every  $x$ .  
Rationale: The function is continuous on the interval  $(-\infty,1]$  and from below on the interval  $[1,\infty]$ .
- ID: FR3  
Owner: Pravallika Basam  
Version: 1.0  
Description: The 1st quadrant will always include the positive values whereas the 2<sup>nd</sup> quadrant will always contain the negative values.  
Rationale: The principal value range of the function is  $[0,\pi]$ .
- ID: FR4  
Owner: Pravallika Basam

Version: 1.0

Description: The type of the input data should always be double. An exception would be thrown for any other data type.

Rationale: arccos function is defined on the domain  $[-1,1]$

- ID: FR5

Owner: Pravallika Basam

Version: 1.0

Description: The value for  $\cos^{-1}(1/x) = \sec^{-1}(x)$

Rationale: The function should satisfy other related trigonometric functions.

- ID: FR6

Owner: Pravallika Basam

Version: 1.0

Description: The value for  $\cos^{-1}(x) = \pi - \sin^{-1}(x)$

Rationale: The function should satisfy other related trigonometric functions.

- ID: FR7

Owner: Pravallika Basam

Version: 1.0

Description: The value for  $\cos^{-1}(-x) = \pi - \cos^{-1}(x)$

Rationale: The function should satisfy other related trigonometric functions.

## 2.2 Non-Functional Requirements

- ID: NFR1

Owner: Pravallika Basam

Version: 1.0

Description: The system should produce accurate results in 1 second.

Rationale: The algorithm ought to be quick enough to produce the right results.

- ID: NFR2

Owner: Pravallika Basam

Version: 1.0

Description: Users should easily determine what the system is about and what it is doing.

Rationale: Even a novice user should be able to quickly understand the system's user interface.

### 3 Algorithm and PseudoCode

The inverse of a given number  $x$  is the inverse of the cosine function at an angle  $\theta$ . The arccos function can be calculated in a variety of ways. The function can be solved using following methods:

- Maclaurin's Series.
- Taylor's Series.
- Recursion

The Maclaurin's Series and Taylor's series are represented for the inverse cosine function as,

$$\sin^{-1}(x) = x + \frac{1}{2} \frac{x^3}{3} + \frac{1}{2} \frac{3}{4} \frac{x^5}{5} + \frac{1}{2} \frac{3}{4} \frac{5}{6} \frac{x^7}{7} + \dots \quad |x| < 1$$

$$\begin{aligned} \cos^{-1}(x) &= \frac{\pi}{2} - \sin^{-1}(x) \\ &= \frac{\pi}{2} - x + \frac{1}{2} \frac{x^3}{3} + \frac{1}{2} \frac{3}{4} \frac{x^5}{5} + \frac{1}{2} \frac{3}{4} \frac{5}{6} \frac{x^7}{7} + \dots \end{aligned}$$

Taylor's series is used to compute arccos due to its simplicity. This series can also be used to solve complex arguments.

---

**Algorithm 1** Calculate arccos function

---

```

1: procedure FACT( $c$ )
2:    $factorial \leftarrow 1$  for  $i \leftarrow 1, exponent$  do
3:      $factorial \leftarrow factorial * i$ 
4:
5:   return  $factorial$ 
6: end procedure

7: procedure CALCULATEARCCOS( $x, n$ )
8:    $pi \leftarrow 3.14$ 
9:    $i \leftarrow (value - 1)/(value + 1)$  for  $i \leftarrow 1, \infty$  do
10:     $factorial \leftarrow (factorial * i)$ 
11:     $sum \leftarrow pow(x, i) FACT(i)()$ 
12:
13:     $approx \leftarrow 1 + value$ 
14:     $z \leftarrow (pi/2) - approx$ 
15: end procedure

16:  $a \leftarrow CALCULATENATURALLOG(x)$  ▷ Calculates  $\ln x$ 
17:  $b \leftarrow CALCULATENATURALLOG(b)$  ▷ Calculates  $\ln b$ 
18:  $result \leftarrow a/b$  ▷ Final result of  $\log_b x$ 

```

---

## 4 Pros and Cons of Taylor's Series

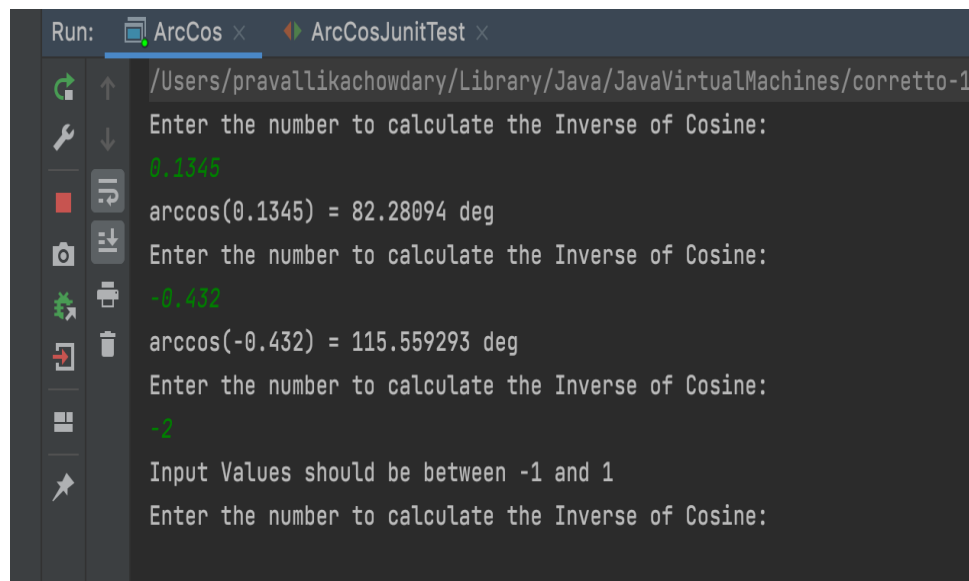
### Pros

- The simplicity of Taylor's series is its main advantage.
- The most difficult problems can often be solved with the help of this series.
- The Taylor's series is useful in creating the foundation for numerous functions and procedures.
- Additionally, Taylor's series helps in obtaining theoretical error boundaries.

### Cons

- The time required to solve the equations is its main drawback.
- In Taylor's series, the round-off error and truncation error might come that disturbs the whole calculation.
- It is not as efficient when comes to direct approximation.

## 5 Working of the Function



```
Run: ArcCos x ArcCosJUnitTest x
/Users/pravallikachowdary/Library/Java/JavaVirtualMachines/corretto-1
Enter the number to calculate the Inverse of Cosine:
0.1345
arccos(0.1345) = 82.28094 deg
Enter the number to calculate the Inverse of Cosine:
-0.432
arccos(-0.432) = 115.559293 deg
Enter the number to calculate the Inverse of Cosine:
-2
Input Values should be between -1 and 1
Enter the number to calculate the Inverse of Cosine:
```

Figure 2: Text-based user interface of the arccos implementation

Java is used for the arccos function implementation. The IDE used to develop code and test cases is called IntelliJ. And, The type of interface is a text-based user interface (TUI).

## 6 Debugger

### 6.1 Description

IntelliJ IDEA provides a debugger for Java code. Run your software while it is connected to the debugger during a debugging session. The debugger's will then interrupt the execution of the program and give you details about what's going on internally at each step. This makes it easier to find and correct errors in the program.

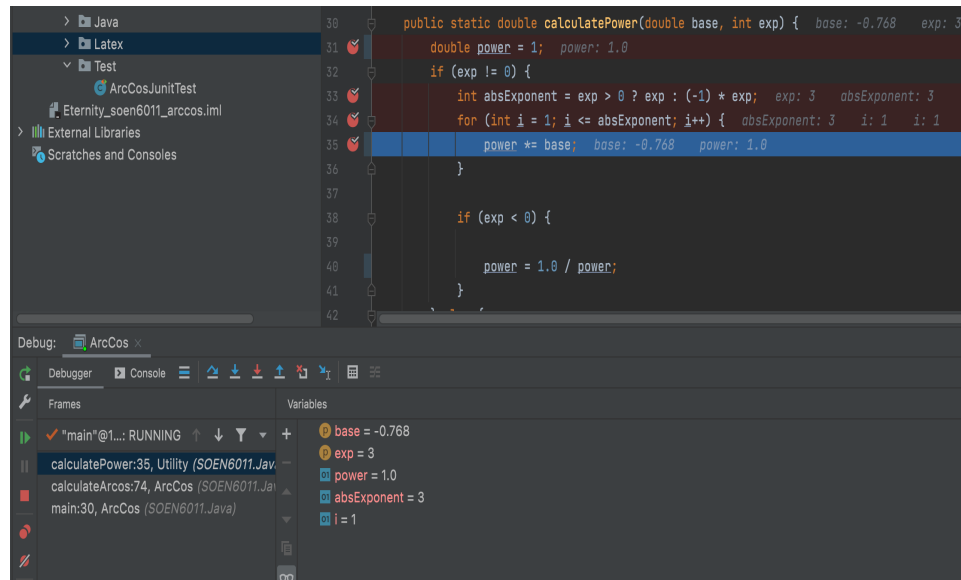


Figure 3: Working of an IntelliJ Debugger

### 6.2 Pros

- Debugging mode makes it incredibly simple to test predicted outputs as variables' values can be changed instantly.
- It can step into or over a statement using step filtering technique.
- It offers event-based breakpoints.
- It offers a feature that shows the item's logical structure, enabling us to comprehend the object in a meaningful structure or view.

### 6.3 Cons

- The JVM design will cause method breakpoints to significantly slow down the debugger. The overall performance may even be slowed down as a

result of this.

- When the number of breakpoints is high, the debugger may crash.

## 7 Quality Attributes

### 7.1 Correctness

The function is checked against all potential values of  $x$ , and the outcomes are confirmed using a real scientific calculator.

### 7.2 Efficient

The program is designed to produce output quickly, perhaps in less than a second. Additionally, the program is space-efficient as less memory is utilised to store the variables.

### 7.3 Maintainable

Since the program is divided into methods for performing different functions, adding new functionality or making changes is made simpler. Any developer may easily understand the code by reading the comments and the documentation.

### 7.4 Robust

In order to prevent software failure, the application is tested for every possible type of incorrect input.

### 7.5 Usable

Since the program uses a textual interface, which is more practical and understandable, anyone may do the task effectively.

## 8 Quality of Source Code

### 8.1 Description

To evaluate the quality of source code, SonarLint is used. As you code, this IntelliJ IDE addon helps you find and resolve quality and security issues. Similar to a spell checker, SonarLint highlights errors while offering immediate feedback and detailed remedial instructions to produce clean code instantly.



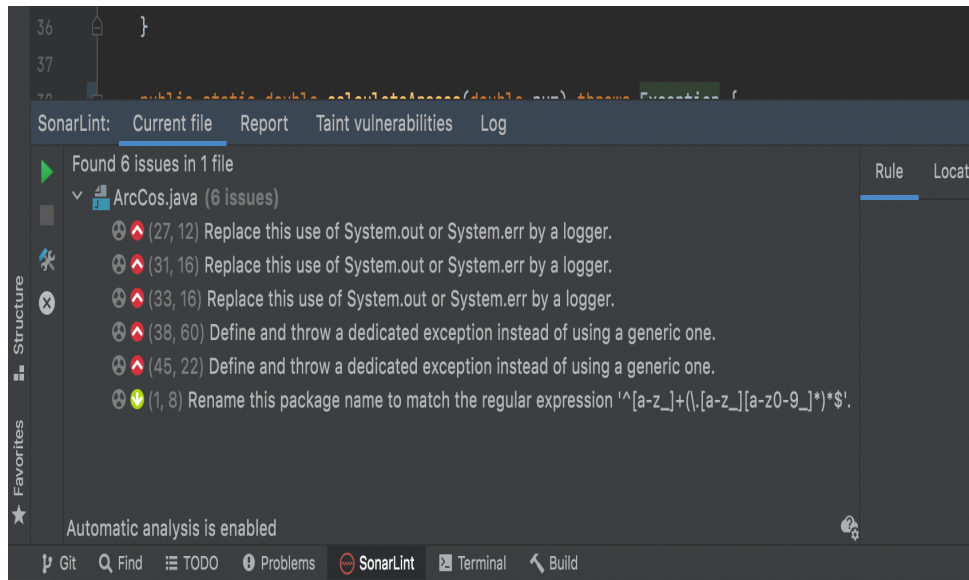


Figure 4: Working of SonarLint

## 8.2 Advantages

- Highly configurable and supports any coding standard.
- It highlights source code that is not reachable.
- It provides the appropriate error/warning messages as per the coding style applied.

## 8.3 Disadvantages

- SonarLint does not support viewing your code coverage in IntelliJ.
- There is no mechanism to report concerns with SonarLint.

## 9 Testing

The project makes advantage of JUnit testing to examine how methods inside classes behave. The test cases written are understandable and clear. Every method had test cases prepared for the anticipated outcomes and occasionally exception-throwing instances to see if the method could handle the exceptions the manner we wanted. Test Suite is built keeping the frequent code changes in mind.

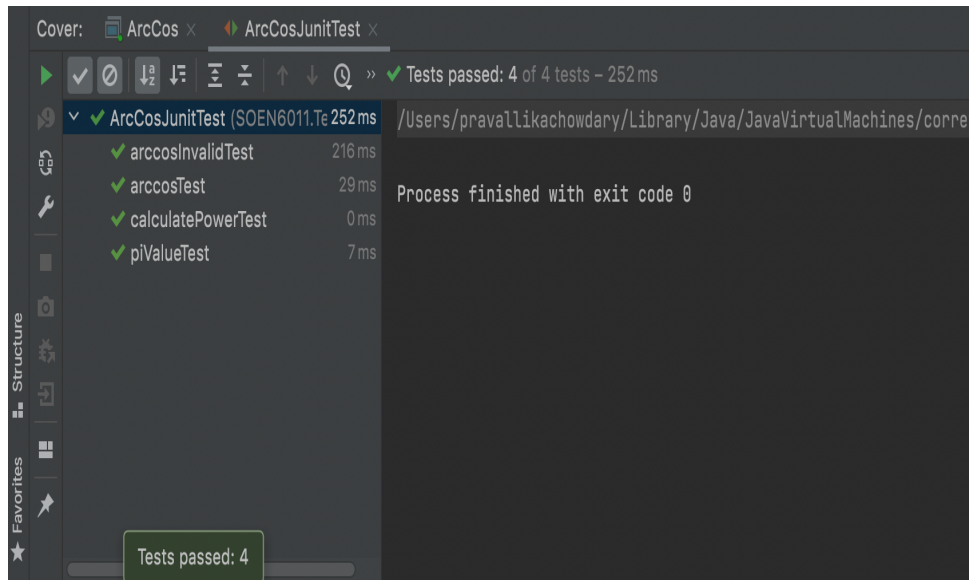


Figure 5: Working of JUnit test cases

## References

- [1] Handbook of Mathematical Functions by Milton Abramowitz and Irene A. Stegun  
<https://personal.math.ubc.ca/~cbm/aands/abramowitz-and-stegun.pdf>
- [2] Inverse Trigonometric Functions  
<https://byjus.com/maths/inverse-trigonometric-functions>
- [3] Taylor Series Expansions of Inverse Trigonometric Functions  
<https://www.mathsisfun.com/algebra/taylor-series.html>
- [4] SonarLint  
<https://plugins.jetbrains.com/plugin/7973-sonarlint>
- [5] Debugger  
<https://www.jetbrains.com/help/idea/debugging-code.html>
- [6] OverLeaf LaTeX editor  
<https://www.overleaf.com>
- [7] JUnit Test Cases  
<https://junit.org/junit5/docs/current/user-guide>
- [8] Google programming style  
<https://google.github.io/styleguide/javaguide.html>