PROBLEM 1. PROJECT DESCRIPTION

The arccosine of x is defined as the inverse function of cosine of x where x lies in the range of

$-1 \leq x \leq 1$

Domain and range:

The domain of the arccosine function is from 1 to +1 inclusive and the range is from 0 to radians inclusive (or from 0 to 180). The arccosine functions can be extended to the complex numbers, in which case the domain is all complex numbers.

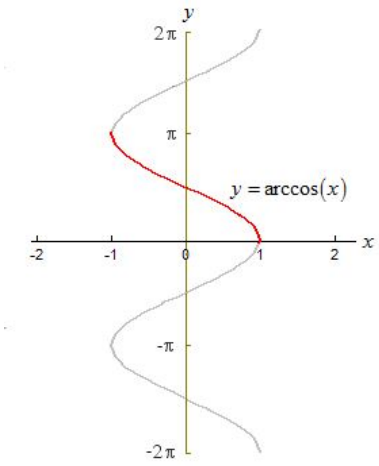

Figure 1: Figure 1: y = arccos(x)

Characteristics of the function y=arccos(x) that make it unique from another inverse trigonometric functions:

1. Domain is in the range of [-1,1].
2. Range is part of [0, π].
3. It is neither even or odd function.
4. It is a decreasing function.

PROBLEM 2: REQUIREMENTS

- First Requirement

  ID = FR1

  Functional Requirements

  Version = 1.0

  Difficulty = Hard

  Description = The result should always be in radians for a given input x.

- Second Requirement

  ID = FR2

  Type = Functional Requirements

  Version = 1.0

  Difficulty = Hard

  Description = For real numbers ranging between -1 to +1, the results should always be in the range of [0, π]

- Third Requirement

  ID = FR3

  Type = Functional Requirements

  Version = 1.0

  Difficulty = Hard

  Description = For certain special arguments, Arccos should automatically evaluate to the exact values

- Fourth Requirement

  ID = FR4

  Type = Functional Requirements

  Version = 1.0

  Difficulty = Easy

  Description = The function should be suitable for numerical manipulation.

- Fifth Requirement

  ID = FR5

  Type = Functional Requirements

  Version = 1.0

  Difficulty = Easy

  Description = The function should be evaluated by arbitrary numerical precision

- Sixth Requirement

  ID = FR6

  Type = Functional Requirements

  Version = 1.0

  Difficulty = Easy

  Description = The function should return an error message upon infinite value as input

Assumptions and Dependency

To reduce the difficulty and risks involved in this projects, we have assumed that the user doesn't give any inputs ranging outside the interval of arc cosine function.

PROBLEM 3: PSEUDOCODE

Algorithm 1: Recursive Approach

```
Function acos (double x){
//It returns inverse cosine value in degrees
/ the returned angle is in the range 0.0 through pi
        //IF the argument i.e., x=NaN or its absolute value is ≥ 1, THEN
            //it is applicable only for values between -1 to 1
            //if the value is other than this it gives error
       // RETURN input error "NaN"
   //ELSE
       // RETURN value of arc cosine of x = (1+x)/pow(1-x,3)
    //END IF
}
```

Algorithm 2: Loop Approach

```
Function acos(double x) {
     //WHILE(x ≤ 1 and x ≥ −1 )
            // x = (1+x)/pow(1-x,3)
        //END WHILE
        //IF (x ≥ 1)
        // throw error message
        //END IF
}
```

Advantages of the choosing algorithm 1 approach

Recursive approach adds clarity and (sometimes) reduces the time needed to write and debug code (but doesn't necessarily reduce space requirements or speed of execution). Reduces time complexity. Performs better in solving problems based on tree structures.

Disadvantages of choosing algorithm 1 approach

It is usually slower due to the overhead of maintaining. It usually uses more memory.

[References]

1. [1]"Inversetrigonometricfunctions",En.wikipedia.org,2019.[Online].Available:https://en.wikipedia.org/wiki/Inverse_trigonometric_functions. [1]"Inverse trigonometric functions", En.wikipedia.org, 2019. [Online]. Available:

2. Reference.wolfram.com. (2019). ArcCos—Wolfram Language Documentation. [online] Available at: https://reference.wolfram.com/language/ref/ArcCos.html