Function:- arccos(x)

Birva Shah (Student ID: 40070973)

July 7, 2019

1 Problem-1

1.1 Definition

The arccosine of x is defined as the inverse cosine function of x when -1≤x≤1. When the cosine of y is equal to x:

$$\cos y = x \tag{1}$$

Then the arccosine of x is equal to the inverse cosine function of x, which is equal to y:

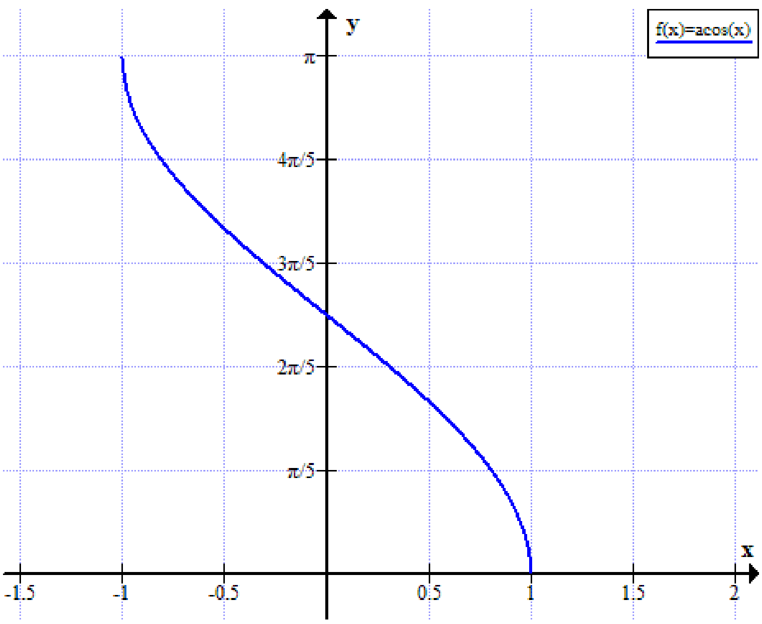
$$\arccos(x) = \cos^{-1} x = y \tag{2}$$

1.2 Domain and Range

The domain of arccos(x) is -1≤x≤1 and the range of arccos(x) is 0≤y≤π (0°≤y≤180°).

1.3 Characteristics of arccos(x)

- This function is neither even nor odd.
- It is a decreasing function.
- Graph of arccos(x)



References

[1] RapidTables,
`https://www.rapidtables.com/math/trigonometry/arccos.html`

[2] Emathhelp,
`https://www.emathhelp.net/notes/algebra-2/trigonometry/function-y-arccos-x/`

Function:- arccos(x)

Problem-2

Birva Shah (Student ID: 40070973)

12 July 2019

1 Assumptions

- In function $arccos(x)$, x is a real number.
- Function returns the value of $arccos(x)$ in radian.
- If the argument of the function is NaN, then the result is NaN.

2 Requirements

(a) **ID =** REQ-1

Type = Functional Requirement

Version = 1.0

Difficulty = Easy

Description = User shall give input value x between -1 and 1 inclusive to satisfythe constraint that the domain of the function $arccos(x)$ is $-1 \leq x \leq 1$.

Rationale = The rationale behind this requirement is that the output of the function $arccos(x)$ is undefined if the value of x is not between -1 and 1 inclusive.

(b) **ID =** REQ-2

Type = Functional Requirement

Version = 1.0

Difficulty = Nominal

Description = The system shall take input x to give the output of the function in radian. For example: $arccos(0.5) = 1.4719....$

Rationale = The rationale behind this requirement is that only one input x which is real number is required to calculate result of arccos(x).

(c) **ID** = REQ-3

Type = Functional Requirement

Version = 1.0

Difficulty = Nominal

Description = The system shall calculate the value of $arccos(x)$ up to the precision of four decimals to get the stable output. For example: $arccos(0.5) = 1.4719$

Rationale = The rationale behind this requirement is that the function might give an output that has infinite number of decimals points.

References

[1] RapidTables,
`https://www.rapidtables.com/math/trigonometry/arccos.html`

[2] Emathhelp,
`https://www.emathhelp.net/notes/algebra-2/trigonometry/function-y-arccos-x/`

[3] Microsoft,
`https://docs.microsoft.com/en-us/powerapps/maker/canvas-apps/functions/function-trig`

[4] Mathonweb,
`http://mathonweb.com/help`$_e book/html/algorithms.htmarcsinCliffsNotes$,
`https://www.cliffsnotes.com/study-guides/trigonometry/inverse-functions-and-equations/inverse-cosine-and-inverse-`

Function:- arccos(x)
Problem-3

Birva Shah (Student ID: 40070973)

19 July 2019

1 Description

There are 2 ways to find arccos(x). One is iterative approach and the other is recursive approach. Taylor's series for the evaluation has been used.

$$\arccos x = \frac{\pi}{2} - \sum_{n=0}^{\infty} \frac{(2n)!}{2^{2n}(n!)^2} \frac{x^{2n+1}}{(2n+1)}, |x|<1 \tag{1}$$

Comparison between these two approaches have been showed in this document. The time complexity of both the algorithms is same but it has been observed that iterative approach is better than the recursive one.
The recursive approach has resulted in high memory consumption compared to the other one.

2 Advantages and Disadvantages of Iterative Algorithm

- If implemented during the earlier stages of the development process allows the team to find functional or design related flaws as early as possible.
- Easily adaptable to the ever-changing needs of the project as well as the client.
- It is the best suited for agile organizations and less time is spent on documenting and more on designing to implement iterative model.
- More resources may be required.
- It is not suitable for small projects project.

Algorithm 1 Calculating: $arccos(x)$ using Iterative Algorithm

```
function PI()
    1. pi_value ← 0.0
    2. for k ≤ 9999
        first ← power(−1, k)
        second ← (2 ∗ k) + 1
        value ← first/second
        pi_value ← pi_value + value
    3. pi_value ← 4 ∗ pi_value
    4. returnpi_value
function ARCCOS(x)
    in : valueofx
    out : calculatedvalueofarccos(x)inradian
    1. ans ← 0
    2. for n ≤ 89
        a = factorial(2 ∗ n)
        if(Double.isInfinite(a))
            break
        b ← power(2, (2 ∗ n))
        c ← factorial(n)
        d ← power(c, 2)
        A ← (a/(b ∗ d))
        exp ← (2 ∗ n) + 1
        e ← power(num, exp)
        B ← e/exp
        AB ← (A ∗ B)
        ans ← ans + AB
    3. pivalue ← pi()
    4. finalans ← ((pivalue/2) − ans)
    5. returnfinalans
function POWER(c, j)
    in : valueofcandj
    out : valueofpower(c, j)
    1. ans ← 1.0
    2. if(j == 0)
        ans ← 1
       else
        for i ≤ j
            ans ← c ∗ ans
    3.return ans
function FACTORIAL(i)
    in : valueofi
    out : valueoffactorial(i)
    1. ans ← 1.0
    2. if(i == 0)
        ans ← 1
       else
        forj ≤ i
            ans ← ans ∗ j
    3. returnans
```

Algorithm 2 Calculating: $arccos(x)$ using Recursive Algorithm

function $PI()$
1. $pi\_value \leftarrow 0.0$
2. for $k \leq 9999$
   $first \leftarrow power(-1, k)$
   $second \leftarrow (2 * k) + 1$
   $value \leftarrow first/second$
   $pi\_value \leftarrow pi\_value + value$
3. $pi\_value \leftarrow 4 * pi\_value$
4. $returnpi\_value$

function $ARCCOS(x)$
$in : valueofx$
$out : calculatedvalueofarccos(x)inradian$
$ans \leftarrow FUNC(x, 0, 0)$
$ans \leftarrow ((PI/2) - ans)$
$returnans$

function $FUNC(value, steps, ans)$
$in : valueofvalue, stepsandans$
$out : valueoffunc(value, steps, ans)$
1. $a = factorial(2 * steps)$
2. $if(Double.isInfinite(a))$
   $stepsByMethod = steps - 1$
   $returnans$
3. $b \leftarrow power(2, (2 * n))$
4. $c \leftarrow factorial(n)$
5. $d \leftarrow power(c, 2)$
6. $A \leftarrow (a/(b * d))$
7. $exp \leftarrow (2 * n) + 1$
8. $e \leftarrow power(num, exp)$
9. $B \leftarrow e/exp$
10. $AB \leftarrow (A * B)$
11. $ans \leftarrow ans + AB$
12. $steps \leftarrow steps + 1$
13. $returnFUNC(value, steps, ans)$

function $POWER(c, j)$
$in : valueofcandj$
$out : valueofpower(c, j)$
1. $ans \leftarrow 1.0$
2. $if(j == 0)$
   $ans \leftarrow 1$
   else
   for $i \leq j$
   $ans \leftarrow c * ans$
3. return ans

function $FACTORIAL(i)$
$in : valueofi$
$out : valueoffactorial(i)$
1. $ans \leftarrow 1.0$
2. $if(i == 0)$
   $ans \leftarrow 1$
   else
   $forj \leq i$
   $ans \leftarrow ans * j$
3. $returnans$