SOEN 6011:SOFTWARE ENGINEERING PROCESSES

Deliverable #1

SHASHANK RAO
SID:40104247
https://github.com/ShashankRao17/SOEN6011-Software-Engineering-Processes

1 Problem-1

1.1 Description

The common schoolbook definition of the cosine of an angle θ in a right-angled triangle is given by,

$$\cos\ \theta = \frac{adjacent}{hypotenuse}$$

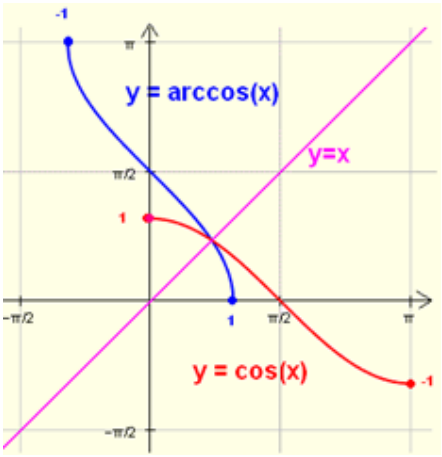
In mathematics, the inverse trigonometric functions (also called arcus function, antitrigonometric functions or cyclometric functions) are the inverse of the basic trigonometric functions (Specifically they are the inverse of sine, cosine, tangent, cotangent, secant and cosecant functions) and are used to obtain an angle from any of the angle's trigonometric ratios. Thus, similar to the definition of cosine, the arccos can be defined as,

$$\arccos\ \theta = \frac{hypotenuse}{adjacent}$$

1.2 Graph, Domain & Range of arccos(x)

Arccos(x) is the inverse function of f(x)=cos(x) for $0 \leq x \leq \pi$. The domain of y=arccos(x) is the range of f(x)=cos(x) for $0 \leq x \leq \pi$ and given by the interval [-1,1]. The range of arccos(x) is the domain of f which is given by the interval $[0,\pi]$.

The graph, domain and range of both cos(x) and arccos(x) is as shown below,



1.3 Arccos Table

The below table contains some of the commonly calculated values of x for various angles(θ) in Radian(Rad) & Degrees($^\circ$).

x	arccos(x) (Rad)	arccos(x) ($^\circ$)
-1	π	180°
$-\sqrt{3}/2$	$5\pi/6$	150°
$-\sqrt{2}/2$	$3\pi/4$	135°
$-1/2$	$2\pi/3$	120°
0	$\pi/2$	90°
$1/2$	$\pi/3$	60°
$\sqrt{2}/2$	$\pi/4$	45°
$\sqrt{3}/2$	$\pi/6$	30°
1	0	0°

2 Problem-2

2.1 Problem Statement

To develop a system in Java to calculate the result for the trigonometric function arccos(x).

2.2 Requirements

(Requirements are denoted by unique identifiers preceeding with 'R'.)

Below are few constraints that need to be followed:

R1. **Interface requirement:** The system(function) shall interact with the user for input needed for start of computation.

R2. **Quality requirement:** The system shall be flexible, reliable, reusable and maintainable due to the complex nature of calculations involved.

R3. **Functional requirement:** The system shall display appropriate error messages/have appropriate error handling mechanism to ensure understanability to the user in case of incorrect input given.

2.3 Assumptions

(Constraints are denoted by unique identifiers preceeding with 'C'.)

Below are few constraints that need to be followed:

C1. The user shall input the value of x in the range of $-1 \leq x \leq 1$.

C2. The value of π upto 10 decimal places shall be considered for computation and related calculations.

3 Problem-3

3.1 Algorithm 1-Pseudocode

This program allows users to calculate the inverse of cos(arccos) for a variable x and fetch the respective measure in degrees using the **Taylor series expansion formula**.

function power(argument1,argument2){
Calculates the result for $(argument1)^{argument2}$
if argument1 is equal to 0
    return undefined error
else if argument2 is equal to 0
    return 1
else
    return the value of first argument raised to the power of second argument
end
}

function factorial(argument1){
Calculates the factorial(n!) of the passed argument
Check the validity of the argument
if argument1 less than 0
    return undefined error
else if argument1 is equal to 0
    return 1
else
    return the factorial of the passed argument
end
}

function acos(argument1){
Calculates the inverse of cosine(arccos) for a variable in the range of [-1,1] using the Taylor series expansion formula:

$arccos(x) = \frac{\pi}{2} - arcsin(x)$
$arccos(x) = \frac{\pi}{2} - \sum_{n=0}^{\infty} \frac{2n!}{2^{2n}(n!)^2} \frac{x^{2n+1}}{2n+1}$ which converges for -1≤x≤1

if argument1 is less than -1 or greater that 1
    return invalid input error
else
    Call the factorial and power function to perform the appropriate calculations
return the inverse of cosine(arccos(x))
end
}

{
In the **main function**
Print prompt "Input the value of variable for finding arccos"
Take the input from user
Send the input to the acos function and display the result for the user
}

The above algorithm is the most accurate of the 2 algorithms in this section. This algorithm is distributed into various functions thereby using the core object oriented principles of Java such as polymorphism and encapsulation. The more the number of iterations in the formula, the lesser would be the error coefficient. It also makes use of recursion to find the factorial of the numbers needed in the formula.

3.2 Algorithm 2-Pseudocode

This program allows users to calculate the inverse of cos(arccos) for a variable x and fetch the respective measure in degrees using the Chebyshev-Pade quotient approximation.

function acos(argument){
if argument is less than -1 or greater than 1
    return invalid input error
else
    calculate result using the following Chebyshev-Pade quotient approximation:
result $= \frac{\pi}{2} - (0.5689111419 - 0.2644381021 * argument - 0.4212611542 * (2 * argument - 1) * (2 * argument - 1) + 0.1475622352 * (2 * argument - 1) * (2 * argument - 1) * (2 * argument - 1))/(2.006022274 - 2.343685222 * argument + 0.3316406750 * (2 * argument - 1) * (2 * argument - 1) + 0.02607135626 * (2 * x - 1) * (2 * x - 1) * (2 * x - 1))$
 end
}

{
In the **main function**
Print prompt "Input the value of variable for finding arccos"
Take the input from user
Send the input to the acos function
}

The above algorithm is a simpler approximation of arccos(x) with lesser memory utilization and load on JVM. Being directly associated with approximated quotients, the calculations are lighter and easy to compute.

3.3 Advantages vs Disadvantages

Pseudocode-1

Advantages: 1. Accuracy is more than pseudocode-2 with an error coefficient of ≈ 0.6-0.7 off the original value.

2. Better readability and understandability of code due to the disctribution of various critical functions rather than have a congestion of information(or code).

Disadvantages: 1. Lengthier implementation of code and possibility of error in logic due to the need of various functions for end output.

Pseudocode-2

Advantages: 1. The simplest of implementation since the code involves mostly constants which do not need to be computed separately.
2. Less intensive on the JVM due to the need of only a single function for the end output computation.

Disadvantages: 1. The error coefficient is the major concern due to difference of ≈ 7-8 degrees off the original value.

4 References

i. http://mathworld.wolfram.com/Cosine.html

ii. https://www.analyzemath.com/Graphing/graphing_arccosine.html

iii. https://en.wikipedia.org/wiki/Inverse_trigonometric_functions

iv. https://www.rapidtables.com/math/trigonometry/arccos.html#definition

v. https://ieeexplore.ieee.org/document/8559686

vi. https://www.mathportal.org/formulas/pdf/taylor-series-formulas.pdf

vii. https://cs.uwaterloo.ca/ kogeddes/papers/Numapprox/Numapprox.html