arccos(x)

Deliverable 3 - inverse Cosine:

The original plan was to use the CORDIC algorithm to solve for the inverse trigonometric function. However, due to overall complexity and issues related to time management the algorithm was implemented using a mixture of Taylor series expansion and trigonometric identities

The function:

* The arccos(x) function is contained in a separate class which would be imported by the parser and the main class where the GUI is implemented
* It only accepts one main attribute i.e value
* It has 3 main methods and 1 support method
  + - \_\_init\_\_ ()- constructer class
    - asin\_taylor\_sum() - calculates asin using Taylor series expansion
    - asin\_interval() - Even though the convergence radius of the series expansion is 1,the series will eventually converge for -1 < x < 1, convergence is indeed painfully slow close to the limits of this interval. The solution is to somehow avoid these parts of the interval. As a result, I’ve used series interval.
    - arccos() - using trig identity using asin() to solve for acos():
* (self.pi / 2.0) - self.asin(self.value)
* The interval is divided into:
  + using original algorithm for |x| <= 1/sqrt(2),
  + using the identity arcsin(x) = pi/2 - arcsin(sqrt(1-x^2)) for 1/sqrt(2) < x <= 1.0,
  + using the identity arcsin(x) = -pi/2 + arcsin(sqrt(1-x^2)) for -1.0 <= x < -1/sqrt(2).

This way we can transform our input x into [-1/sqrt(2),1/sqrt(2)], where convergence is relatively fast.

Pseudocode

function asin\_taylor(String mNum)

temp = value  
 factor = 1.0  
 divisor = 1  
 asin\_sum = value

for i in range (0, 40):  
 temp \*= value \* value  
 divisor \*= 2.0  
 factor \*= (2.0 \* i + 1.0) / ((i + 1.0) \* 2.0)  
 asin\_sum += (factor \* temp) / divisor  
 return asin\_sum

function asin(self, value):  
 if abs(value) is less than 0.71:  
 return self.asin\_taylor\_sum(value)  
 else if value is greater than 0:  
 temp = float(1.0 - (value \* value))  
 return (self.pi / 2.0) - self.asin\_taylor\_sum(math.sqrt(temp))  
 else:  
 temp = float(1.0 - (value \* value))  
 return self.asin\_taylor\_sum(math.sqrt(temp)) - (self.pi / 2.0)

function acos(String mNum)

return (pi/2) - asin(value)