

SOEN 6011 DELIVERABLE ONE

Qing Li

Student id: 40082701

1 Function Description

$$F2 : \tan x = \frac{\textit{opposite}}{\textit{adjacent}} \quad (1)$$

Domain: all real numbers except the values $\frac{\pi}{2} + \pi k$ for all integers k

Co-domain: all real numbers \mathbb{R} .

Characteristic:

-Period = π

-X intercepts: $x = k\pi$, where k is an integer.

-Y intercepts: $y = 0$

-Symmetry: since $\tan(-x) = -\tan(x)$ then $\tan(x)$ is an odd function and its graph is symmetric with respect to the origin.

-Intervals of increase/decrease: over one period and from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$, $\tan(x)$ is increasing.

-Vertical asymptotes: $x = \frac{\pi}{2} + k\pi$, where k is an integer.

2 Function Requirements

1. The input X shall be a real number.
 - Type attribute: Design Constraints
 - Priority: High
2. The calculation result shall be a real number.
 - Type attribute: Design Constraints
 - Priority: High
3. The user shall input a real number X into the user interface.
 - Type attribute: Functional
 - Priority: High
4. Non-real number input shall be detected by the user interface.
 - Type attribute: Functional
 - Priority: High
5. Empty input shall be detected by the user interface.
 - Type attribute: Functional
 - Priority: High
6. When error inputs are detected, the user interface shall report error messages to users.
 - Type attribute: Functional
 - Priority: High
7. When the user interface receives an approval input X, the user interface shall pass this value X to the function to do the calculation.
 - Type attribute: Functional
 - Priority: High

8. When the function receives a real number X through the user interface, the function shall calculate the result of $\tan(X)$.
 - Type attribute: Functional
 - Priority: High
9. When calculation is finished, the result of calculation shall be returned to the user interface.
 - Type attribute: Functional
 - Priority: High
10. When a calculation result $\tan(X)$ is returned to the user interface, the user interface shall show the result to the user.
 - Type attribute: Functional
 - Priority: High
11. The calculation result shall be accurate to 6 decimal places.
 - Type attribute: Performance
 - Priority: High
12. Performance: After inputting X , user should receive the calculation result from the function within 20 seconds.
 - Type attribute: Non-Functional
 - Priority: Medium
13. Reusability: The function $\tan(X)$ may be composited with other scientific calculation functions to form a scientific calculator.
 - Type attribute: Non-Functional(assumption)
 - Priority: Low
14. Reliability: The function shall return the correct result for any approval input X .
 - Type attribute: Non-Functional
 - Priority: High

3 Function Algorithms

[Pseudocode Format]

The pseudocode format used by our team refers pseudocode format of CLRS.

[Algorithm selection]

I select two algorithms for implementation of my function $\tan(x)$:

1. Calculating $\tan(x)=\sin(x)/\cos(x)$ based on iteration.
2. Calculating $\tan(x)$ by using Taylor series based on recursion.

$$\tan x = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} 2^{2n} (2^{2n} - 1) B_{2n}}{(2n)!} x^{2n-1}$$

[Algorithm 1]

Description:

Calculating the value of $\tan(x)$ by using function $\tan(x)=\sin(x)/\cos(x)$, and for $\sin(x)$ and $\cos(x)$, I use The Maclaurin series to calculate each of them. The subordinate functions are the power function and the factorial function. The whole algorithm is coded based on iteration.

Pseudocode:

TAN(x)

```
1: for  $k = 1$  to 24
2:    $i = 2 * k - 1$ 
3:    $j = 2 * k - 2$ 
4:    $s = cons * pwr(x, i) / fac(i)$ 
5:    $c = cons * pwr(x, j) / fac(j)$ 
```

```

6:   $\sin = \sin + s$ 
7:   $\cos = \cos + c$ 
8: end for
9:  $\tan = \sin / \cos$ 
PWR( $x, n$ )
1: if  $n=0$  then then
2:   Return 1
3: end if
4: for  $i = 1$  to  $n$ 
5:    $power = power * x$ 
6: end for
7: return  $power$ 
FAC( $n$ )
1: if  $n = 0$  or  $n = 1$  then then
2:   return 1
3: end if
4: for  $i = 2$  to  $n$ 
5:    $pdt = pdt * i$ 
6: end for
7: return  $pdt$ 

```

Technical reasons

Time complexity: $O(n^2)$

Advantages:

1. By using iteration, you could just have a single set of local variables, this saves the time and memory that would be used for passing these things in the recursive calls.
2. Iterative functions are typically faster than their recursive counterparts.

Disadvantages:

Iteration is more difficult to understand in some algorithms. Because the code using iteration doesn't have a good readability and understandability.

[Algorithm 2]

Description:

Calculating the value of $\tan(x)$ by using Taylor series directly. The sub-ordinate functions are the power function, the factorial function and the Bernoulli function. The whole algorithm is coded based on recursion.

Pseudocode:

TAN(x)

```
1: for  $i = 1$  to 24
2:    $e = 1.0 * (pwr(-1, i - 1) * pwr(2, 2 * i) * (pwr(2, 2 * i) - 1.0) * Bernoulli(2 * i) * pwr(x, 2 * i - 1)) / fac(2 * i)$ 
3:    $sum = sum + e$ 
4: end for
```

PWR(x, n)

```
1: if  $n = 0$  then then
2:   return 1
3: else
4:   return  $x * pwr(x, n - 1)$ 
5: end if
```

FAC(n)

```
1: if  $n = 0$  or  $n = 1$  then then
2:   return 1
3: else
4:   return  $n * fac(n - 1)$ 
5: end if
```

BERNOULLI(n)

```
1: if  $n = 1$  then then
2:   return 1
3: else
4:   while  $k$ 
5:      $k = k - 1$ 
6:      $B = B + -1.0 * (fac(n) * Bernoulli(k)) / (fac(n - k) * fac(k) * (x - k + 1))$ 
7:   end while
```

8: end if

Technical reasons:

Time complexity: $O(n^2)$

Advantages:

1. Code using recursion is usually simplicity and readability.
2. Using recursion, the length of the program can be reduced.

Disadvantages:

1. It requires extra storage space. The recursive calls and automatic variables are stored on the stack. For every recursive calls separate memory is allocated to automatic variables with the same name.
2. Often the algorithm may require large amounts of memory if the depth of the recursion is very large. If the programmer forgets to specify the exit condition in the recursive function, the program will execute out of memory.
3. The recursion function is not efficient in execution speed and time.

4 Function Implementation

github: https://github.com/loliamserious/SOEN6011-QingLi_TeamF

Debugger:

CheckStyle:

correct, efficient, maintainable, robust, usable