19CCE203 Computational Electromagnetics – Lab Evaluation – 1
Date: 13/10/2021 (Wednesday)

Aim: Develop a program to plot Taylor series expansion of trigonometric functions.

Software:
MATLAB R2021b – academic use
Publisher – MathWorks
Version – Update 2 (9.11.0.1837725)
64-bit (win64)

Theory:
A Taylor series is a polynomial of infinite degree that can be used to represent many different functions, particularly functions that aren't polynomials. Taylor series has applications ranging from classical and modern physics to the computations that your hand-held calculator makes when evaluating trigonometric expressions.

The partial sum formed by the first n + 1 terms of a Taylor series is a polynomial of degree n that is called the nth Taylor polynomial of the function. Taylor polynomials are approximations of a function, which become generally better as n increases. Taylor's theorem gives quantitative estimates on the error introduced by the use of such approximations.

If the Taylor series of a function is convergent, its sum is the limit of the infinite sequence of the Taylor polynomials. A function may differ from the sum of its Taylor series, even if its Taylor series is convergent. A function is analytic at a point x if it is equal to the sum of its Taylor series in some open interval (or open disk in the complex plane) containing x. This implies that the function is analytic at every point of the interval (or disk).

The usual trigonometric functions and their inverses have the following Maclaurin series:

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \qquad \text{for all } x$$

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots \qquad \text{for all } x$$

$$\tan x = \sum_{n=1}^{\infty} \frac{B_{2n}(-4)^n\left(1-4^n\right)}{(2n)!} x^{2n-1} = x + \frac{x^3}{3} + \frac{2x^5}{15} + \cdots \qquad \text{for } |x| < \frac{\pi}{2}$$

All angles are expressed in radians. The numbers B_k appearing in the expansions of tan x are the Bernoulli numbers.

Graph Plotting Algorithm:
To plot the graph of a function, you need to take the following steps –
1. Define the x-axis and corresponding y-axis values as lists.
2. Plot them on canvas using the plot() function.
3. Give a name to x-axis and y-axis using xlabel() and ylabel() functions.
4. Give a title to your plot using the title() function.
5. To set the visibility of the grid inside the figure to on, we use the grid() function.
6. Finally, to view your legend, we use the show() function.

Code:

sin(x)

```
clear
clc
close all

fprintf("\nThe Taylor Series for sinx is as follows: -")
% Given values of x, taken in radians:
values = [0, pi/6, pi/4, pi/3, pi/2, 2*pi/3, pi, 2*pi, 0.429*pi, 0.683*pi]; %
Values of x

for store_values = 1:1:10 % Array to store indices of x values.
    fprintf("\n\nTaylor Series for x[%d] = %d is as follows: -", store_values,
values(store_values))

    % Perform necessary initializations.
    number_of_terms = 20; % Terms of the series.
    temp = 0; % Temporary variable to store the previous value of sum.
    sum = (0) * (number_of_terms); % Array to store sigma summation for each value
of x.
    array = (0) * (number_of_terms); % Final array that adds the sums at each
iteration.

    % For each number of terms, get the function value.
    for i = 0:number_of_terms-1
        % sin(x) = E (x^(2n+1)) * (-1)^n/(2n+1)! using the series expansion.
        sum(i+1) = (pwr(-1, i))*(pwr(values(store_values), (2 * i) + 1))/ftl((2 *
(i)) + 1);

        temp = temp + sum(i+1); % Add the sum of previous iteration.
        array(i+1) = temp; % Save it to the final array of summation.
        fprintf("\nFor term = %d, the sum is: %d", i+1, array(i+1)) % Display each
iteration.
    end

    % Identify  the  number  of  terms  required to arrive at a convergent value.
    for i = 2:number_of_terms-1

        % Check whether the previous sum value is equal to the next.
        % Rounding off up to five decimal places for better approximation results.
        if (round(array(i),5) == round(array(i-1),5))
            fprintf("\nThe number of terms required to arrive at a convergent
value for x[%d] = %d is %d.\n", store_values, values(store_values), i)
            break;
        end

    end

    % A single plot contains graphs of five x values.
    if store_values <= 5
        fig = figure(1);
        set(fig, 'color', 'white')
        grid on
        xlabel('Number of Terms')
        ylabel('Function Value')
        title('sin(x) taylor series for first five values of x')

        % For proper legend marking, string concatenation used.
        hold on
        plot(array,'-
*',"LineWidth",2,'DisplayName',strcat('sin(',num2str(values(store_values)),')'))

    else
        fig = figure(2);
        set(fig, 'color', 'white')
        grid on
        xlabel('Number of Terms')
        ylabel('Function Value')
        title('sin(x) taylor series for next five values of x')

        % For proper legend marking, string concatenation used.
        hold on
        plot(array,'-
*',"LineWidth",2,'DisplayName',strcat('sin(',num2str(values(store_values)),')'))

    end

    % Show the legend markings.
    legend('show')
end

% Compute Factorial of a Number:
function fact = ftl(number)
fact = 1; % Initialize factorial variable
    for temp = 1:number
        fact = fact * temp;
    end
end

% Compute Power Using Recursion:
function expo = pwr(base, a)
    if a~=0
        expo = base * pwr(base, a-1);
        return
    else
        expo = 1;
        return
    end
end
```

cos(x)

```
clear
clc
close all

fprintf("\nThe Taylor Series for cosx is as follows: -")
% Given values of x, taken in radians:
values = [0, pi/6, pi/4, pi/3, pi/2, 2*pi/3, pi, 2*pi, 0.429*pi, 0.683*pi]; %
Values of x

for store_values = 1:1:10 % Array to store indices of x values.
    fprintf("\n\nTaylor Series for x[%d] = %d is as follows: -", store_values,
values(store_values))

    % Perform necessary initializations.
    number_of_terms = 20; % Terms of the series.
    temp = 0; % Temporary variable to store the previous value of sum.
    sum = (0) * (number_of_terms); % Array to store sigma summation for each
value of x.
    array = (0) * (number_of_terms); % Final array that adds the sums at each
iteration.

    % For each number of terms, get the function value.
    for i = 0:number_of_terms-1
        % cos(x) = E (x^(2n)) * (-1)^n/(2n)! using the series expansion.
        sum(i+1) = (pwr(-1, i))*(pwr(values(store_values), (2 * i)))/ftl(2 * (i));

        temp = temp + sum(i+1); % Add the sum of previous iteration.
        array(i+1) = temp; % Save it to the final array of summation.
        fprintf("\nFor term = %d, the sum is: %d", i+1, array(i+1)) % Display each
iteration.
    end

    % Identify  the  number  of  terms  required to arrive at a convergent value.
    for i = 2:number_of_terms-1

        % Check whether the previous sum value is equal to the next.
        % Rounding off up to five decimal places for better approximation results.
        if (round(array(i),5) == round(array(i-1),5))
            fprintf("\nThe number of terms required to arrive at a convergent
value for x[%d] = %d is %d.\n", store_values, values(store_values), i)
            break;
        end

    end

    % A single plot contains graphs of five x values.
    if store_values <= 5
        fig = figure(1);
        set(fig, 'color', 'white')
        grid on
        xlabel('Number of Terms')
        ylabel('Function Value')
        title('cos(x) taylor series for first five values of x')

        % For proper legend marking, string concatenation used.
        hold on
        plot(array,'-
*',"LineWidth",2,'DisplayName',strcat('cos(',num2str(values(store_values)),')'))

    else
        fig = figure(2);
        set(fig, 'color', 'white')
        grid on
        xlabel('Number of Terms')
        ylabel('Function Value')
        title('cos(x) taylor series for next five values of x')

        % For proper legend marking, string concatenation used.
        hold on
        plot(array,'-
*',"LineWidth",2,'DisplayName',strcat('cos(',num2str(values(store_values)),')'))

    end

    % Show the legend markings.
    legend('show')
end

% Compute Factorial of a Number:
function fact = ftl(number)
fact = 1; % Initialize factorial variable
    for temp = 1:number
        fact = fact * temp;
    end
end

% Compute Power Using Recursion:
function expo = pwr(base, a)
    if a~=0
        expo = base * pwr(base, a-1);
        return
    else
        expo = 1;
        return
    end
end
```

tan(x)

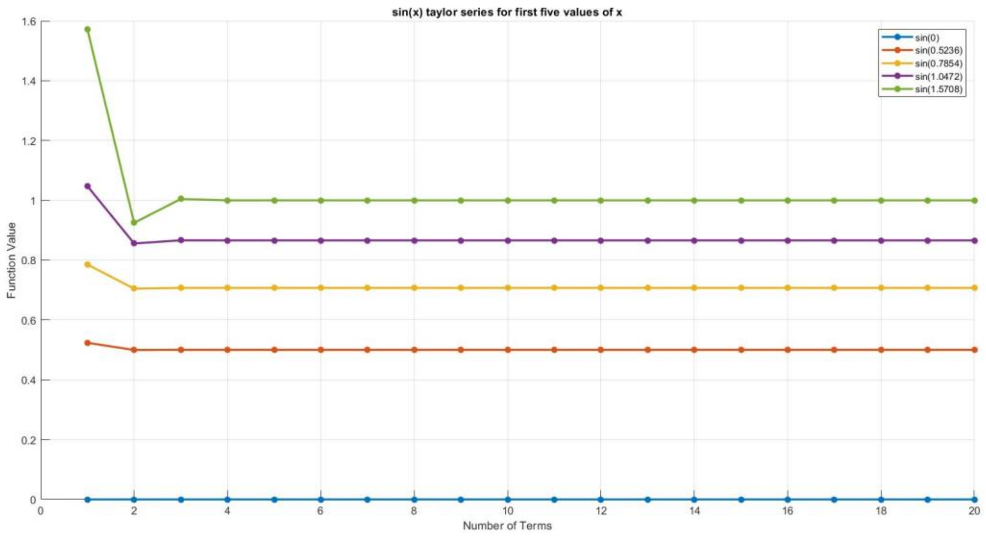
```
clear
clc
close all

fprintf("\nThe Taylor Series for tanx is as follows: -")
% Given values of x, taken in radians:
values = [0, pi/6, pi/4, pi/3, pi/2, 2*pi/3, pi, 2*pi, 0.429*pi, 0.683*pi]; %
Values of x

for store_values = 1:1:10 % Array to store indices of x values.
    fprintf("\n\nTaylor Series for x[%d] = %d is as follows: -", store_values,
values(store_values))

    % Perform necessary initializations.
    number_of_terms = 20; % Terms of the series.
    temp = 0; % Temporary variable to store the previous value of sum.
    sum = (0) * (number_of_terms); % Array to store sigma summation for each value
of x.
    array = (0) * (number_of_terms); % Final array that adds the sums at each
iteration.

    % For each number of terms, get the function value.
    for i = 0:number_of_terms-1
        % tan(x) = sin(x)/cos(x) using the series expansion.
        sum(i+1) = ((pwr(-1, i))*(pwr(values(store_values), (2 * i) + 1))/ftl((2 *
(i)) + 1)) / ((pwr(-1, i))*(pwr(values(store_values), (2 * i)))/ftl(2 * (i)));

        temp = temp + sum(i+1); % Add the sum of previous iteration.
        array(i+1) = temp; % Save it to the final array of summation.
        fprintf("\nFor term = %d, the sum is: %d", i+1, array(i+1)) % Display each
iteration.
    end

    % Identify  the  number  of  terms  required to arrive at a convergent value.
    for i = 2:number_of_terms-1
```

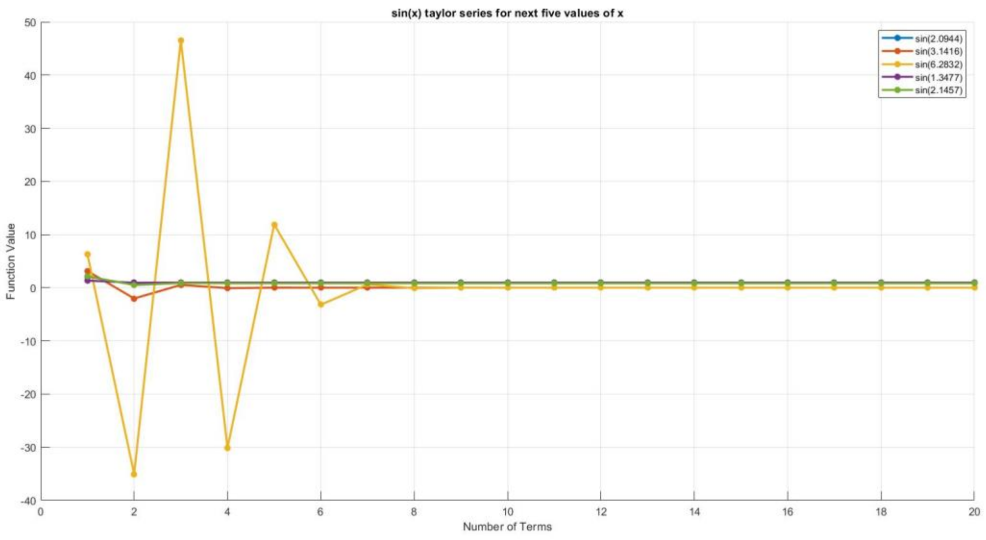
```
        % Check whether the previous sum value is equal to the next.
        % Rounding off up to five decimal places for better approximation
results.
        if (round(array(i),5) == round(array(i-1),5))
            fprintf("\nThe number of terms required to arrive at a convergent
value for x[%d] = %d is %d.\n", store_values, values(store_values), i)
            break;
        end

    end

    % A single plot contains graphs of five x values.
    if store_values <= 5
        fig = figure(1);
        set(fig, 'color', 'white')
        grid on
        xlabel('Number of Terms')
        ylabel('Function Value')
        title('tan(x) taylor series for first five values of x')

        % For proper legend marking, string concatenation used.
        hold on
        plot(array,'-
*',"LineWidth",2,'DisplayName',strcat('tan(',num2str(values(store_values)),')'))

    else
        fig = figure(2);
        set(fig, 'color', 'white')
        grid on
        xlabel('Number of Terms')
        ylabel('Function Value')
        title('tan(x) taylor series for next five values of x')

        % For proper legend marking, string concatenation used.
        hold on
        plot(array,'-
*',"LineWidth",2,'DisplayName',strcat('tan(',num2str(values(store_values)),')'))

    end

    % Show the legend markings.
    legend('show')
end

% Compute Factorial of a Number:
function fact = ftl(number)
fact = 1; % Initialize factorial variable
    for temp = 1:number
        fact = fact * temp;
    end
end

% Compute Power Using Recursion:
function expo = pwr(base, a)
    if a~=0
        expo = base * pwr(base, a-1);
        return
    else
        expo = 1;
```

```
            return
        end
    end
```

Outputs:

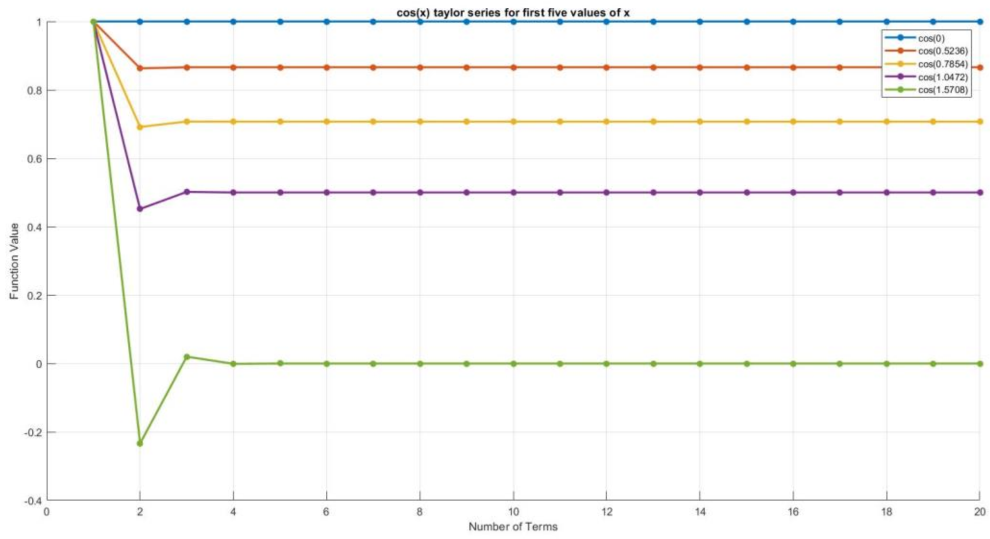
sin(x)

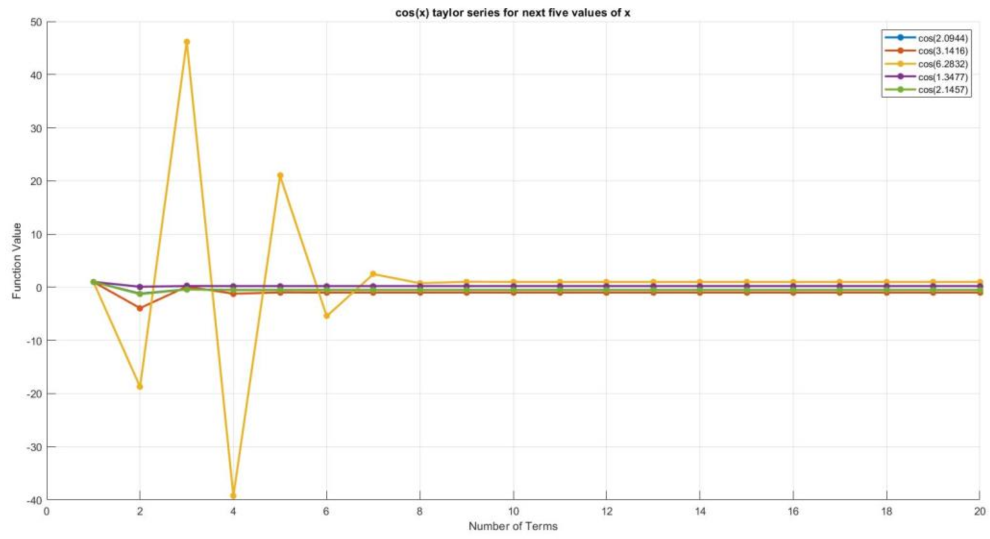




```
The number of terms required to arrive at a convergent value for x[1] = 0 is 2.
The number of terms required to arrive at a convergent value for x[2] = 5.235988e-01 is 4.
The number of terms required to arrive at a convergent value for x[3] = 7.853982e-01 is 5.
The number of terms required to arrive at a convergent value for x[4] = 1.047198e+00 is 6.
The number of terms required to arrive at a convergent value for x[5] = 1.570796e+00 is 6.
The number of terms required to arrive at a convergent value for x[6] = 2.094395e+00 is 8.
The number of terms required to arrive at a convergent value for x[7] = 3.141593e+00 is 9.
The number of terms required to arrive at a convergent value for x[8] = 6.283185e+00 is 14.
The number of terms required to arrive at a convergent value for x[9] = 1.347743e+00 is 6.
The number of terms required to arrive at a convergent value for x[10] = 2.145708e+00 is 7.
```

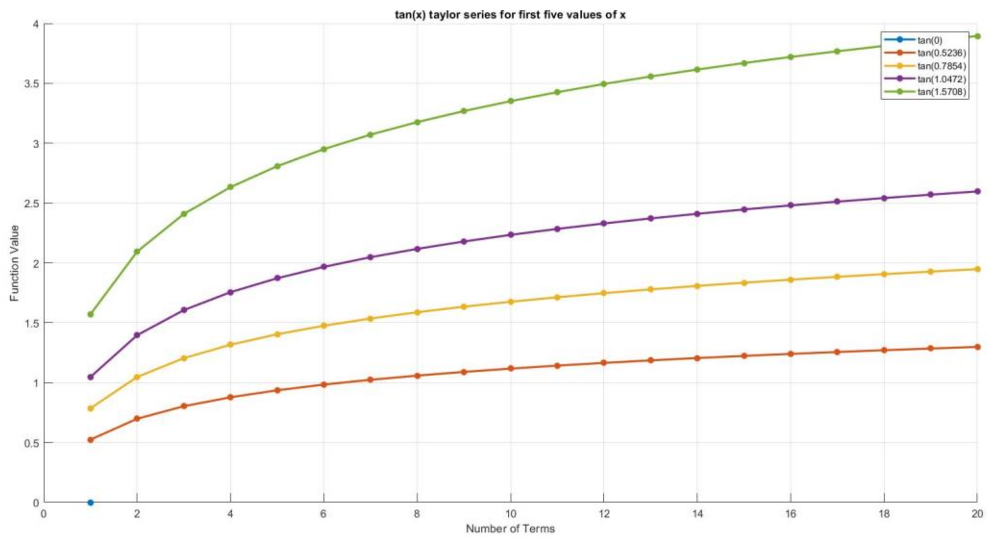
cos(x)

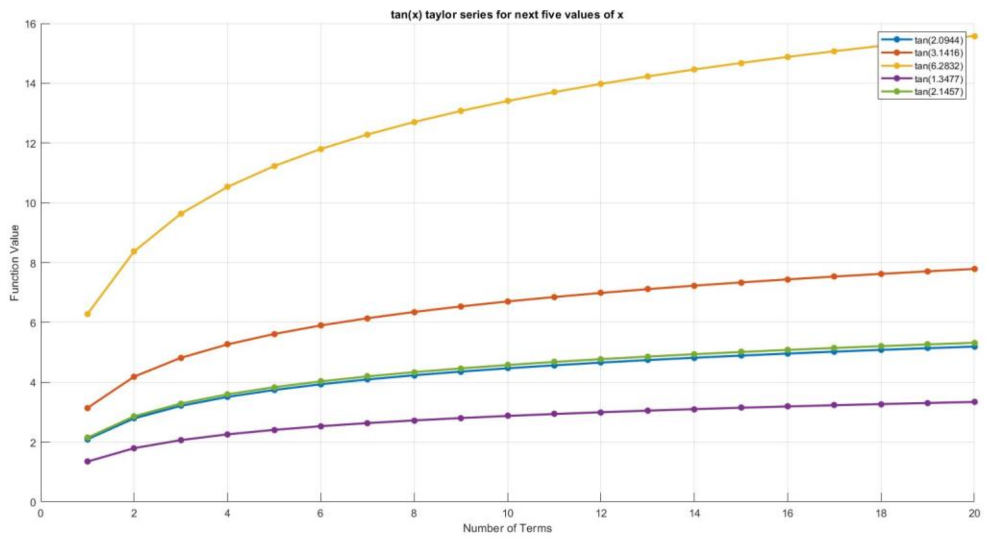




The number of terms required to arrive at a convergent value for x[1] = 0 is 2.
The number of terms required to arrive at a convergent value for x[2] = 5.235988e-01 is 5.
The number of terms required to arrive at a convergent value for x[3] = 7.853982e-01 is 6.
The number of terms required to arrive at a convergent value for x[4] = 1.047198e+00 is 6.
The number of terms required to arrive at a convergent value for x[5] = 1.570796e+00 is 7.
The number of terms required to arrive at a convergent value for x[6] = 2.094395e+00 is 8.
The number of terms required to arrive at a convergent value for x[7] = 3.141593e+00 is 9.
The number of terms required to arrive at a convergent value for x[8] = 6.283185e+00 is 14.
The number of terms required to arrive at a convergent value for x[9] = 1.347743e+00 is 6.
The number of terms required to arrive at a convergent value for x[10] = 2.145708e+00 is 8.

tan(x)

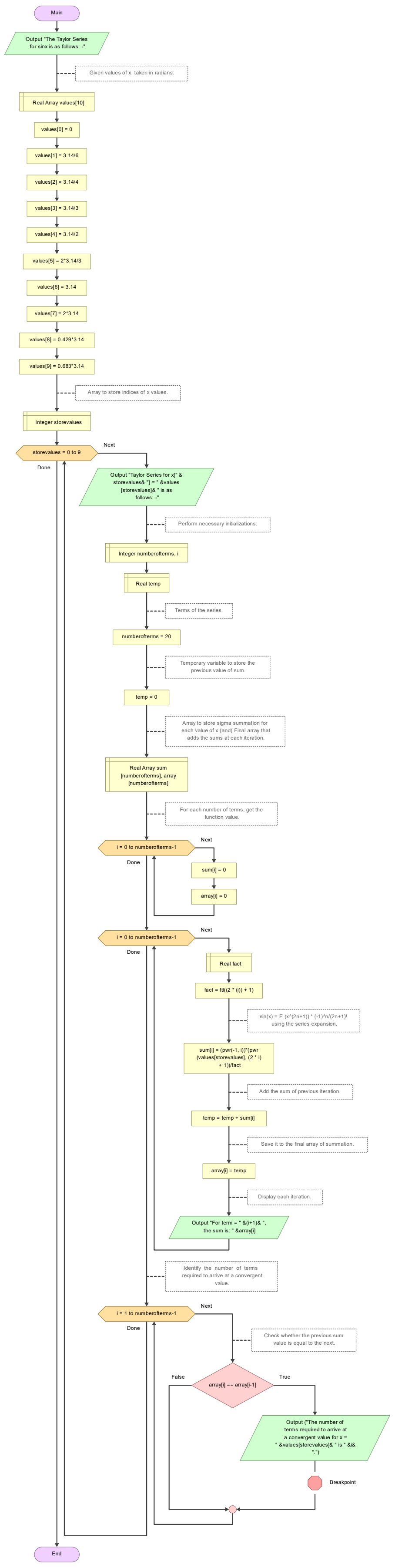


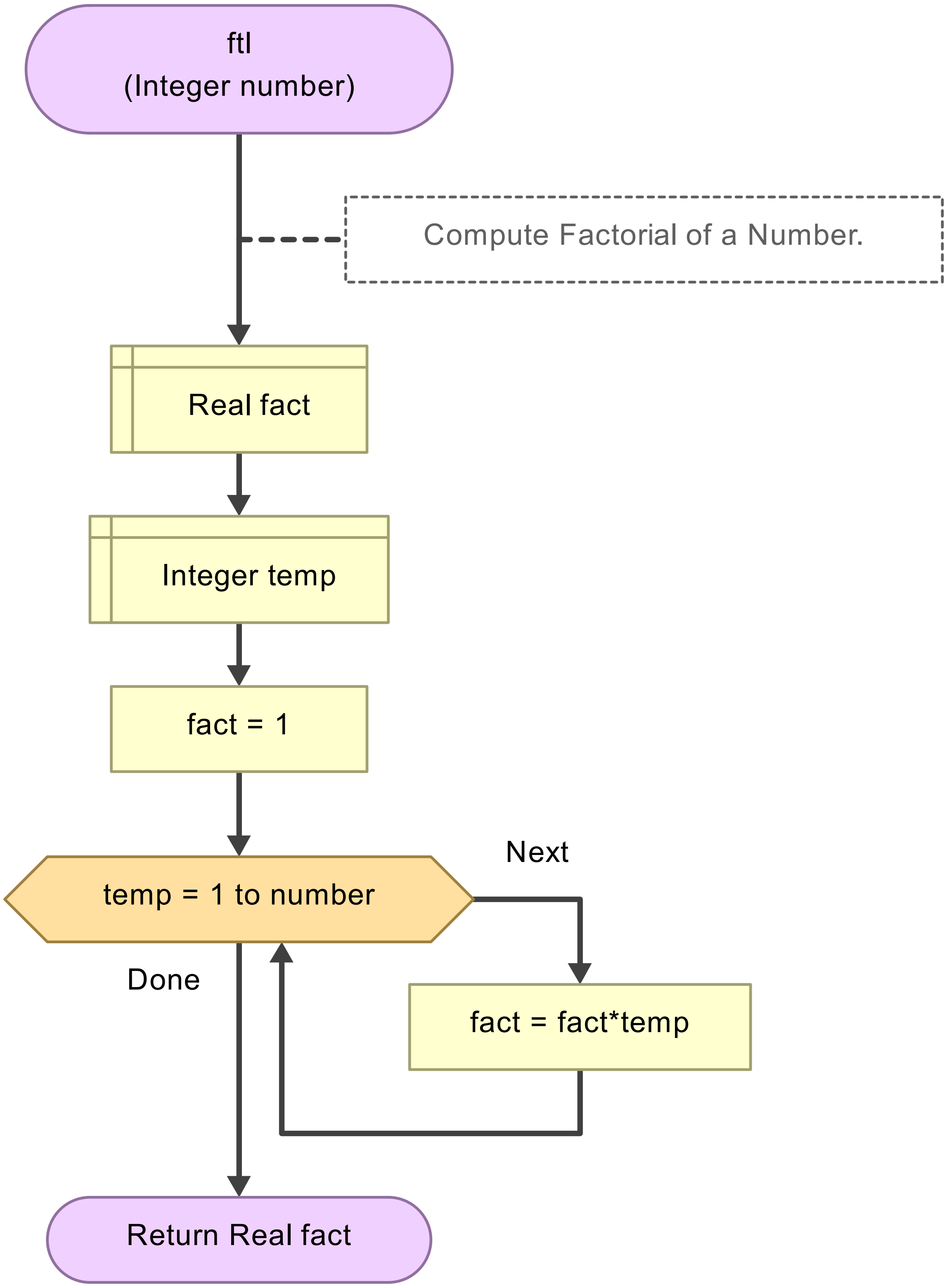


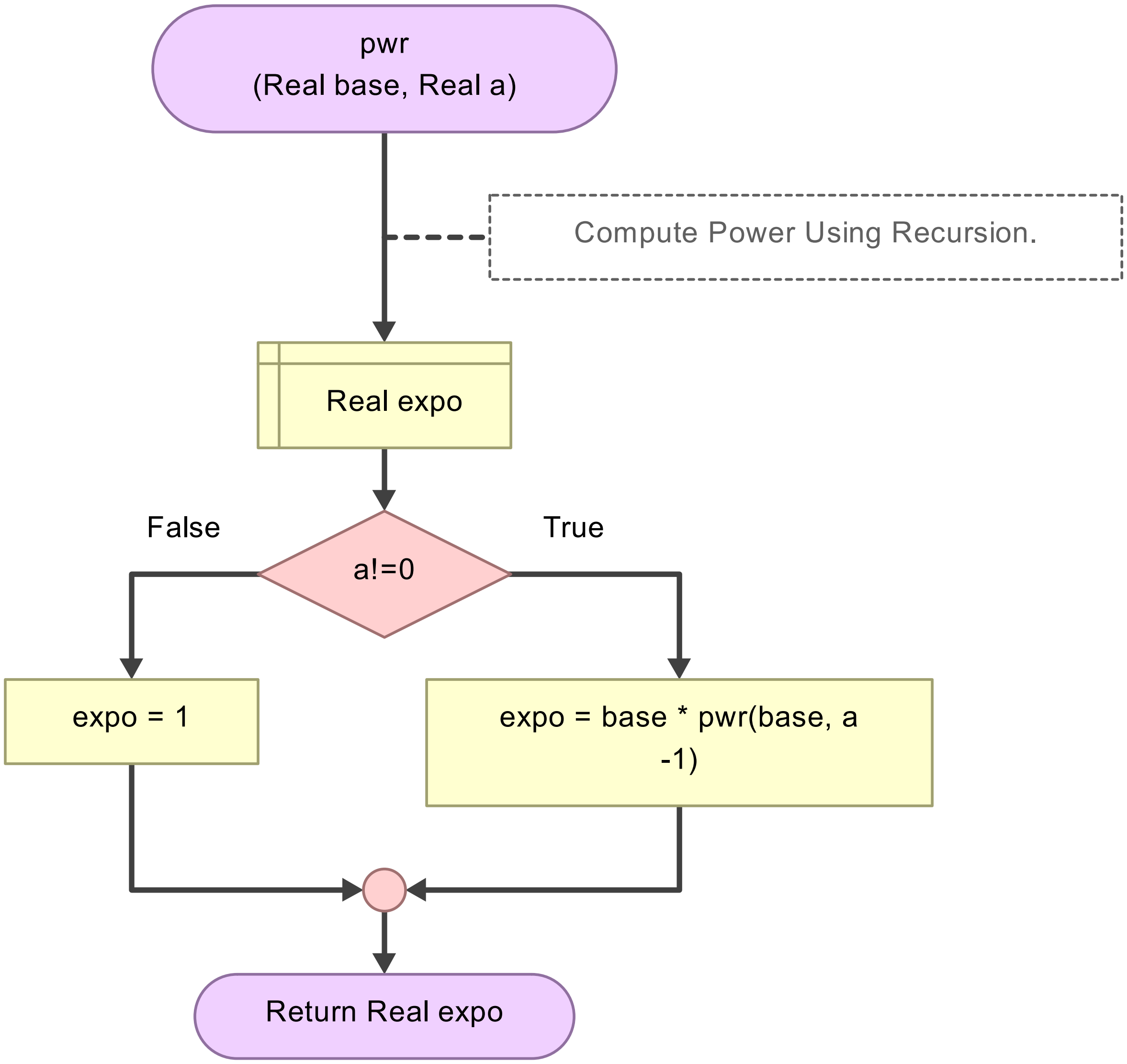
Inference: Plotting of Taylor series expansion for various trigonometric functions along with algorithm have been implemented using MATLAB and results verified.

References:

1. https://brilliant.org/wiki/taylor-series/
2. https://en.wikipedia.org/wiki/Taylor_series
3. MathWorks MATLAB Community Forum
4. Multiple YouTube Videos for Better Understanding of MATLAB


Main
Output "The Taylor Series for sinx is as follows: -"
Given values of x, taken in radians:
Real Array values[10]
values[0] = 0
values[1] = 3.14/6
values[2] = 3.14/4
values[3] = 3.14/3
values[4] = 3.14/2
values[5] = 2*3.14/3
values[6] = 3.14
values[7] = 2*3.14
values[8] = 0.429*3.14
values[9] = 0.683*3.14
Array to store indices of x values.
Integer storevalues
storevalues = 0 to 9
Next
Done
Output "Taylor Series for x[" & storevalues& "] = " &values [storevalues]& " is as follows: -"
Perform necessary initializations.
Integer numberofterms, i
Real temp
Terms of the series.
numberofterms = 20
Temporary variable to store the previous value of sum.
temp = 0
Array to store sigma summation for each value of x (and) Final array that adds the sums at each iteration.
Real Array sum [numberofterms], array [numberofterms]
For each number of terms, get the function value.
i = 0 to numberofterms-1
Next
Done
sum[i] = 0
array[i] = 0
i = 0 to numberofterms-1
Next
Done
Real fact
fact = ftl((2 * (i)) + 1)
sin(x) = E (x^(2n+1)) * (-1)^n/(2n+1)! using the series expansion.
sum[i] = (pwr(-1, i))*(pwr (values[storevalues], (2 * i) + 1))/fact
Add the sum of previous iteration.
temp = temp + sum[i]
Save it to the final array of summation.
array[i] = temp
Display each iteration.
Output "For term = " &(i+1)& ", the sum is: " &array[i]
Identify the number of terms required to arrive at a convergent value.
i = 1 to numberofterms-1
Next
Done
Check whether the previous sum value is equal to the next.
False
array[i] == array[i-1]
True
Output ("The number of terms required to arrive at a convergent value for x = " &values[storevalues]& " is " &i& ".")
Breakpoint
End


ftl
(Integer number)
Compute Factorial of a Number.
Real fact
Integer temp
fact = 1
temp = 1 to number
Next
Done
fact = fact*temp
Return Real fact


pwr
(Real base, Real a)
Compute Power Using Recursion.
Real expo
False
True
a!=0
expo = 1
expo = base * pwr(base, a -1)
Return Real expo