19CCE203 Computational Electromagnetics – Lab Evaluation – 2
Date: 30/11/2021 (Tuesday)

Aim: Develop a program to perform numerical integration of trigonometric functions using Trapezoidal and Simpson's rules.

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

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

% Perform necessary initializations:
number_of_terms = 15; % Terms of the series.
a = 0; % Lower limit of integral.
b = 2*pi/3; % Upper limit of integral.
number_of_intervals = 10; % Number of integration intervals.
I = (0) * (number_of_terms); % Final array that adds the sums at each iteration.

fprintf("\nThe lower limit has the function value f = sin(%d) = %d.", a, f(a))
fprintf("\nThe upper limit has the function value f = sin(%d) = %d.", b, f(b))

fprintf("\n\nTrapezoidal Rule:")

% For each interval, get the function value.
for n = 1:1:number_of_intervals
    h = (b-a)/n; % Size of each segment.
    s = 0.5*(f(a)+f(b)); % Initial summation of first and last segment.

    % Summation of all intervals.
    for i = 1:1:n-1
        s = s + f(a + i*h);
    end

    % Final obtained integral.
    I(n) = h * s;
    fprintf("\nFor term = %d, the sum is: %d", n, I(n)) % Display each iteration.
end

fig = figure(1);
set(fig, 'color', 'white')
grid on
```

```
xlabel('Number of Integration Values')
ylabel('y = ∫sinx')
title('Trapezoidal Rule')
hold on
plot(I,'-*',"LineWidth", 2)

fprintf("\n\nSimpson's 1/3 Rule:")

% For each interval, get the function value.
for n = 1:1:number_of_intervals
    h = (b-a)/n; % Size of each segment.
    s = (f(a)+f(b)); % Initial summation of first and last segment.

    % Summation of all odd intervals.
    for i = 1:2:n-1
        s = s + 4*f(a + i*h);
    end

    % Summation of all even intervals.
    for i = 2:2:n-1
        s = s + 2*f(a + i*h);
    end

    % Final obtained integral.
    I(n) = h/3 * s;
    fprintf("\nFor term = %d, the sum is: %d", n, I(n)) % Display each iteration.
end

fig = figure(2);
set(fig, 'color', 'white')
grid on
xlabel('Number of Integration Values')
ylabel('y = ∫sinx')
title('Simpson 1/3 Rule')
hold on
plot(I,'-*',"LineWidth", 2)

fprintf("\n\nSimpson's 3/8 Rule:")

% For each interval, get the function value.
for n = 1:1:number_of_intervals
    h = (b-a)/n; % Size of each segment.
    s = (f(a)+f(b)); % Initial summation of first and last segment.

    % Summation of all intervals:
    for i = 1:3:n-1
        s = s + 3*f(a + i*h);
    end

    for i = 2:3:n-1
        s = s + 3*f(a + i*h);
    end

    for i = 3:3:n-1
        s = s + 2*f(a + i*h);
    end

    % Final obtained integral.
    I(n) = (3 * h/8) * s;
```

```
    fprintf("\nFor term = %d, the sum is: %d", n, I(n)) % Display each
iteration.
end

fig = figure(3);
set(fig, 'color', 'white')
grid on
xlabel('Number of Integration Values')
ylabel('y = ∫sinx')
title('Simpson 3/8 Rule')
hold on
plot(I,'-*',"LineWidth", 2)

% Taylor Summation Expression sin(x):
function temp = f(x)
number_of_terms = 15; % Terms of the series.
temp = 0; % Temporary variable to store the previous value of sum.
    for i = 0:number_of_terms-1
        temp = temp + (pwr(-1, i))*(pwr(x, (2 * i) + 1))/ftl((2 * (i)) + 1);
    end
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)

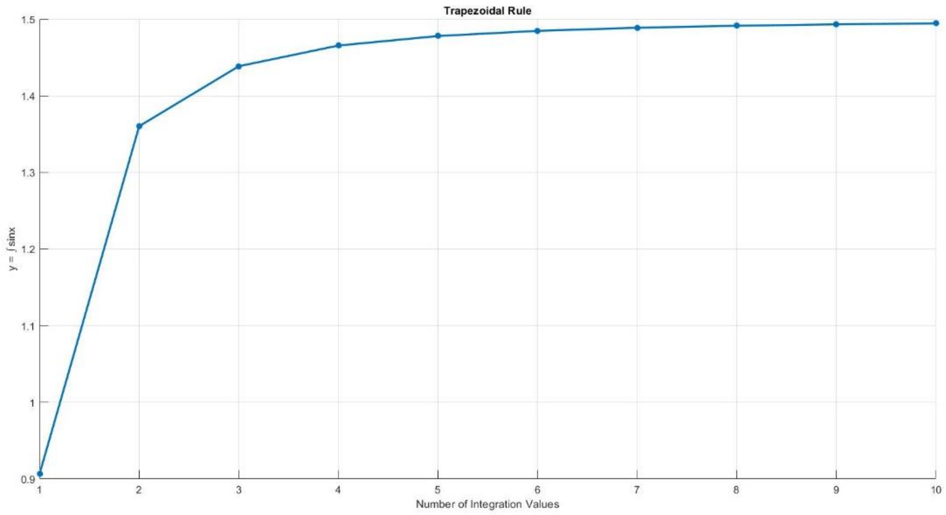
```
clear
clc
close all

% Perform necessary initializations:
number_of_terms = 15; % Terms of the series.
a = 0; % Lower limit of integral.
b = 2*pi/3; % Upper limit of integral.
number_of_intervals = 10; % Number of integration intervals.
I = (0) * (number_of_terms); % Final array that adds the sums at each iteration.

fprintf("\nThe lower limit has the function value f = cos(%d) = %d.", a, f(a))
fprintf("\nThe upper limit has the function value f = cos(%d) = %d.", b, f(b))

fprintf("\n\nTrapezoidal Rule:")

% For each interval, get the function value.
```

```
for n = 1:1:number_of_intervals
    h = (b-a)/n; % Size of each segment.
    s = 0.5*(f(a)+f(b)); % Initial summation of first and last segment.

    % Summation of all intervals.
    for i = 1:1:n-1
        s = s + f(a + i*h);
    end

    % Final obtained integral.
    I(n) = h * s;
    fprintf("\nFor term = %d, the sum is: %d", n, I(n)) % Display each iteration.
end

fig = figure(1);
set(fig, 'color', 'white')
grid on
xlabel('Number of Integration Values')
ylabel('y = ∫cosx')
title('Trapezoidal Rule')
hold on
plot(I,'-*',"LineWidth", 2)

fprintf("\n\nSimpson's 1/3 Rule:")

% For each interval, get the function value.
for n = 1:1:number_of_intervals
    h = (b-a)/n; % Size of each segment.
    s = (f(a)+f(b)); % Initial summation of first and last segment.

    % Summation of all odd intervals.
    for i = 1:2:n-1
        s = s + 4*f(a + i*h);
    end

    % Summation of all even intervals.
    for i = 2:2:n-1
        s = s + 2*f(a + i*h);
    end

    % Final obtained integral.
    I(n) = h/3 * s;
    fprintf("\nFor term = %d, the sum is: %d", n, I(n)) % Display each iteration.
end

fig = figure(2);
set(fig, 'color', 'white')
grid on
xlabel('Number of Integration Values')
ylabel('y = ∫cosx')
title('Simpson 1/3 Rule')
hold on
plot(I,'-*',"LineWidth", 2)

fprintf("\n\nSimpson's 3/8 Rule:")

% For each interval, get the function value.
for n = 1:1:number_of_intervals
    h = (b-a)/n; % Size of each segment.
```

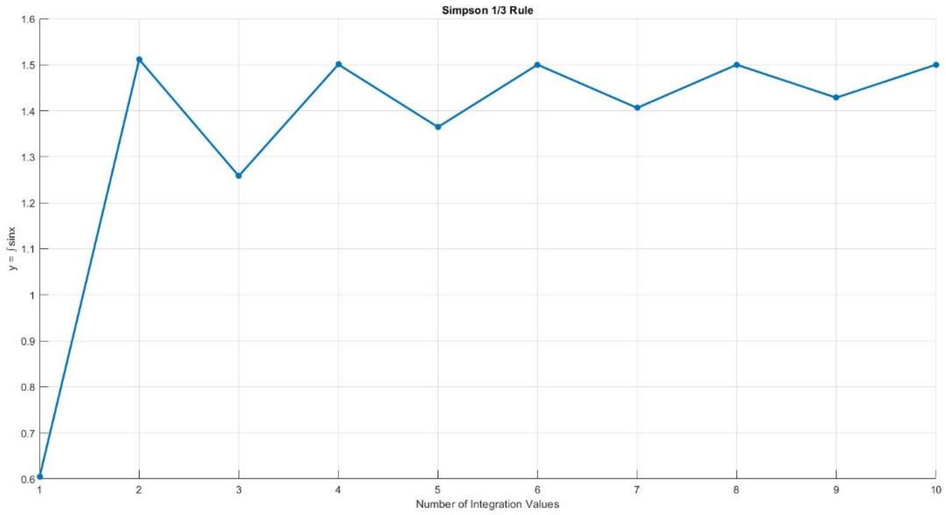
```
    s = (f(a)+f(b)); % Initial summation of first and last segment.

    % Summation of all intervals:
    for i = 1:3:n-1
        s = s + 3*f(a + i*h);
    end

    for i = 2:3:n-1
        s = s + 3*f(a + i*h);
    end

    for i = 3:3:n-1
        s = s + 2*f(a + i*h);
    end

    % Final obtained integral.
    I(n) = (3 * h/8) * s;
    fprintf("\nFor term = %d, the sum is: %d", n, I(n)) % Display each iteration.
end

fig = figure(3);
set(fig, 'color', 'white')
grid on
xlabel('Number of Integration Values')
ylabel('y = ∫cosx')
title('Simpson 3/8 Rule')
hold on
plot(I,'-*',"LineWidth", 2)

% Taylor Summation Expression cos(x):
function temp = f(x)
number_of_terms = 15; % Terms of the series.
temp = 0; % Temporary variable to store the previous value of sum.
    for i = 0:number_of_terms-1
        temp = temp + (pwr(-1, i))*(pwr(x, (2 * i)))/ftl((2 * (i)));
    end
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):

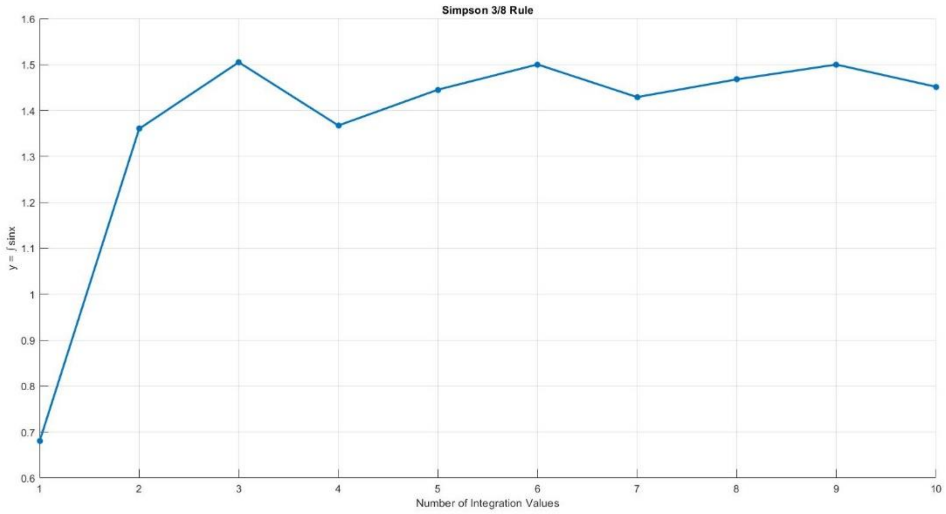
Trapezoidal Rule



Simpson 1/3 Rule

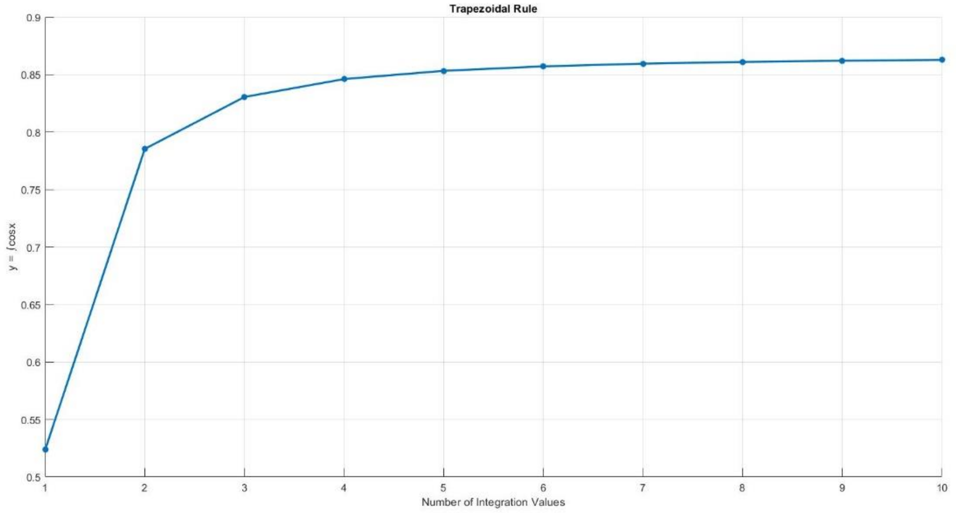


Simpson 3/8 Rule

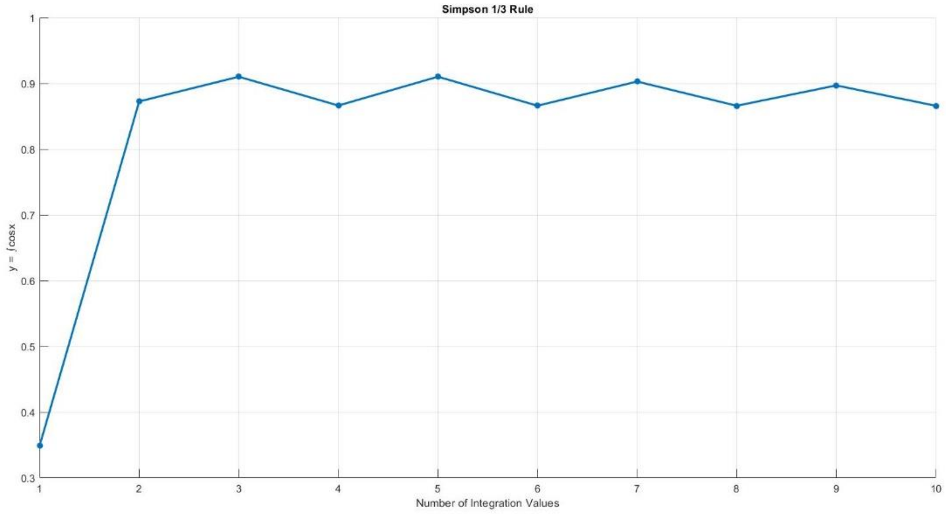


Outputs – cos(x):

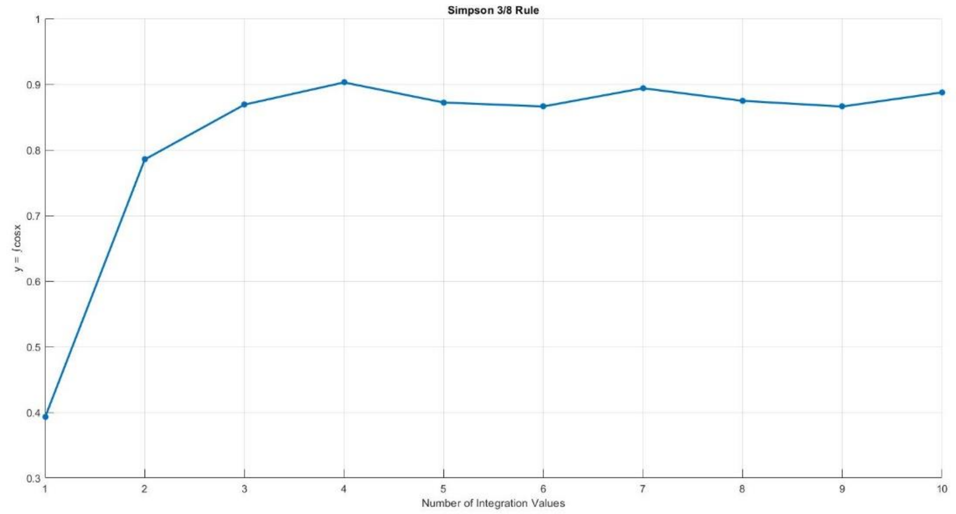
Trapezoidal Rule



Simpson 1/3 Rule



Simpson 3/8 Rule



Inference: Plotting of numerical integration values for sin(x) and cos(x) trigonometric functions along with algorithms of the computation techniques have been implemented using MATLAB and results verified.

(Attached below is the theoretical-computational aspect of the assignment along with a pictorial flowchart representation of the developed program.)

```
The lower limit has the function value f = sin(0) = 0.
The upper limit has the function value f = sin(2.094395e+00) = 8.660254e-01.

Trapezoidal Rule:
For term = 1, the sum is: 9.068997e-01
For term = 2, the sum is: 1.360350e+00
For term = 3, the sum is: 1.438576e+00
For term = 4, the sum is: 1.465573e+00
For term = 5, the sum is: 1.478003e+00
For term = 6, the sum is: 1.484738e+00
For term = 7, the sum is: 1.488793e+00
For term = 8, the sum is: 1.491423e+00
For term = 9, the sum is: 1.493225e+00
For term = 10, the sum is: 1.494513e+00

Simpson's 1/3 Rule:
For term = 1, the sum is: 6.045998e-01
For term = 2, the sum is: 1.511499e+00
For term = 3, the sum is: 1.258217e+00
For term = 4, the sum is: 1.500647e+00
For term = 5, the sum is: 1.364503e+00
For term = 6, the sum is: 1.500126e+00
For term = 7, the sum is: 1.406180e+00
For term = 8, the sum is: 1.500039e+00
For term = 9, the sum is: 1.428314e+00
For term = 10, the sum is: 1.500016e+00

Simpson's 3/8 Rule:
For term = 1, the sum is: 6.801748e-01
For term = 2, the sum is: 1.360350e+00
For term = 3, the sum is: 1.505035e+00
For term = 4, the sum is: 1.367398e+00
For term = 5, the sum is: 1.445344e+00
For term = 6, the sum is: 1.500287e+00
For term = 7, the sum is: 1.429201e+00
For term = 8, the sum is: 1.467745e+00
For term = 9, the sum is: 1.500056e+00
For term = 10, the sum is: 1.451762e+00>>
```

```
The lower limit has the function value f = cos(0) = 1.
The upper limit has the function value f = cos(2.094395e+00) = -5.000000e-01.

Trapezoidal Rule:
For term = 1, the sum is: 5.235988e-01
For term = 2, the sum is: 7.853982e-01
For term = 3, the sum is: 8.305621e-01
For term = 4, the sum is: 8.461489e-01
For term = 5, the sum is: 8.533255e-01
For term = 6, the sum is: 8.572139e-01
For term = 7, the sum is: 8.595552e-01
For term = 8, the sum is: 8.610734e-01
For term = 9, the sum is: 8.621136e-01
For term = 10, the sum is: 8.628574e-01

Simpson's 1/3 Rule:
For term = 1, the sum is: 3.490659e-01
For term = 2, the sum is: 8.726646e-01
For term = 3, the sum is: 9.102414e-01
For term = 4, the sum is: 8.663992e-01
For term = 5, the sum is: 9.102875e-01
For term = 6, the sum is: 8.660979e-01
For term = 7, the sum is: 9.029125e-01
For term = 8, the sum is: 8.660482e-01
For term = 9, the sum is: 8.969727e-01
For term = 10, the sum is: 8.660347e-01

Simpson's 3/8 Rule:
For term = 1, the sum is: 3.926991e-01
For term = 2, the sum is: 7.853982e-01
For term = 3, the sum is: 8.689326e-01
For term = 4, the sum is: 9.028302e-01
For term = 5, the sum is: 8.721810e-01
For term = 6, the sum is: 8.661909e-01
For term = 7, the sum is: 8.939610e-01
For term = 8, the sum is: 8.747438e-01
For term = 9, the sum is: 8.660576e-01
For term = 10, the sum is: 8.875396e-01>>
```

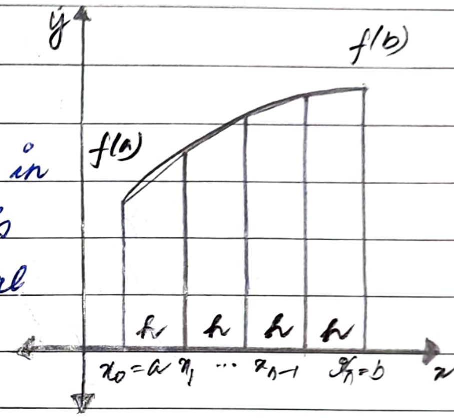
①

classmate Date 09/12/2021 Page 1

NAME - SANTOSH - [CB.EN.U4CCE20053]
DEPARTMENT - COMPUTER AND COMMUNICATION ENGINEERING (CCE) A
SUBJECT - 19CCE203 COMPUTATIONAL ELECTROMAGNETICS

MATLAB ASSIGNMENT III

* TRAPEZOIDAL RULE:



The trapezoidal rule is the most basic numerical integration method. As shown in the figure, the area under the curve is divided to vertical trapezoids having equal width, h, and the upper points coincide with the curve. The area of the first section is $\boxed{A = h\,[f(x_0) + f(x_1)]/2}$

The sum of the areas of the trapezoids can be written as-

$$I = \frac{h}{2}[f(x_0) + f(x_1)] + \frac{h}{2}[f(x_1) + f(x_2)] + \dots$$

$$+ \frac{h}{2}[f(x_{n-2}) + f(x_{n-1})] + \frac{h}{2}[f(x_{n-1}) + f(x_n)]$$

By the sum of similar terms and factoring h, we get-

$$I = h\left\{\frac{1}{2}[f(x_0) + f(x_n)] + f(x_1) + f(x_2) + \dots + f(x_{n-2}) + f(x_{n-1})\right\}$$

Writing the first two terms in terms of the limits a and b,

$$\boxed{I = h\left\{\frac{1}{2}[f(a) + f(b)] + \sum_{i=1}^{n-1} f(x_i)\right\}}$$

Since the step size, h, is constant, the notation of x_i can be implemented in the code as $x_1 \rightarrow a+h$, $x_2 \rightarrow a+2h$ and so on.

$\overrightarrow{PTO}$

(2)

* SIMPSON'S 1/3 RULE :

This method calculates the integral by summing every two divisions at a time, thus, three values of x are taken into account at each turn. To cover the whole domain exactly, the number of strips, n, should even. The formula of the first two strips is -

$$\boxed{A = \frac{1}{3}h\left[f(x_0) + 4f(x_1) + f(x_2)\right]}$$

The integral is computed as the sum of the areas of each pairs of strips.

$$I = \frac{1}{3}h\left[f(x_0) + 4f(x_1) + f(x_2)\right] + \frac{1}{3}h\left[f(x_2) + 4f(x_3) + f(x_4)\right]$$

$$+ \dots + \frac{1}{3}h\left[f(x_{n-4}) + 4f(x_{n-3}) + f(x_{n-2})\right]$$

$$+ \frac{1}{3}h\left[f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right]$$

By grouping terms of similar multipliers, the formula can be rewritten as -

$$I = \frac{1}{3}h\left\{\left[f(x_0) + f(x_n)\right] + 4\left[f(x_1) + f(x_3) + \dots + f(x_{n-3}) + f(x_{n-1})\right] + 2\left[f(x_2) + f(x_4) + \dots + f(x_{n-4}) + f(x_{n-2})\right]\right\}$$

Since the step size, h, is constant, the notation of x_i can be implemented in the code as $x_1 \rightarrow a+h$, $x_2 \rightarrow a+2h$ and so on.

The last form of the integral can simply be put in the following summation -

PTO

$$I = \frac{1}{3} h \left\{ [f(a) + f(b)] + \sum_{i=1,3,5}^{n-1} 4f(x_i) + \sum_{i=2,4,6}^{n-2} 2f(x_i) \right\}$$

The number of strips, n, must be an even integer and summation for-loop increments should be 2.

* SIMPSON'S 3/8 RULE :

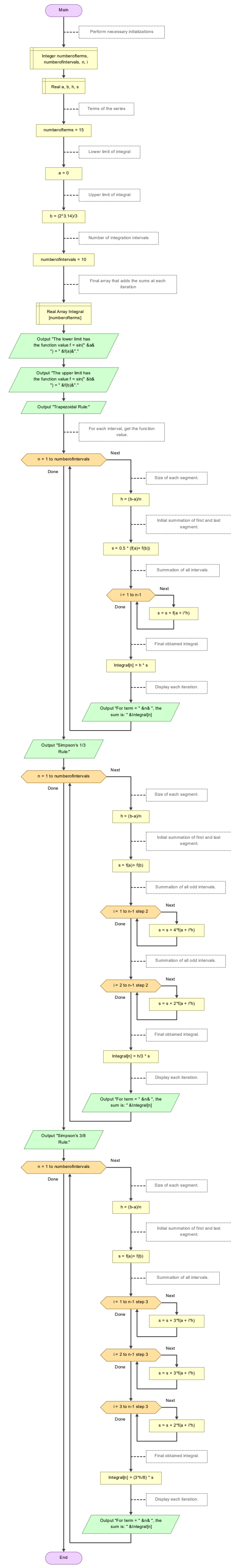
Simpson's 3/8 rule requires the division of the interval into a multiple of 3 subintervals of width h.

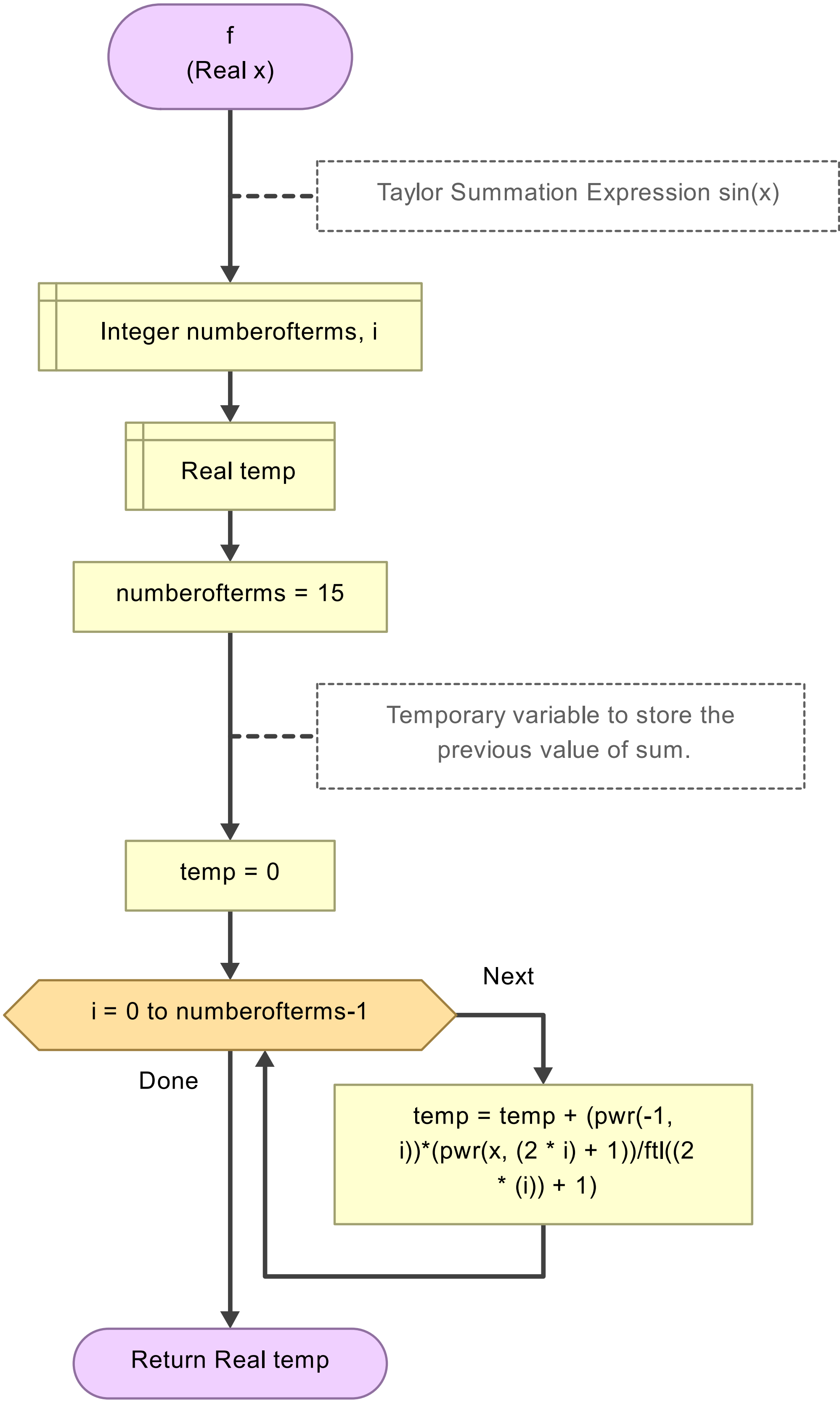
Compute $h = \frac{b-a}{n}$ and check n is a multiple of 3 or not.

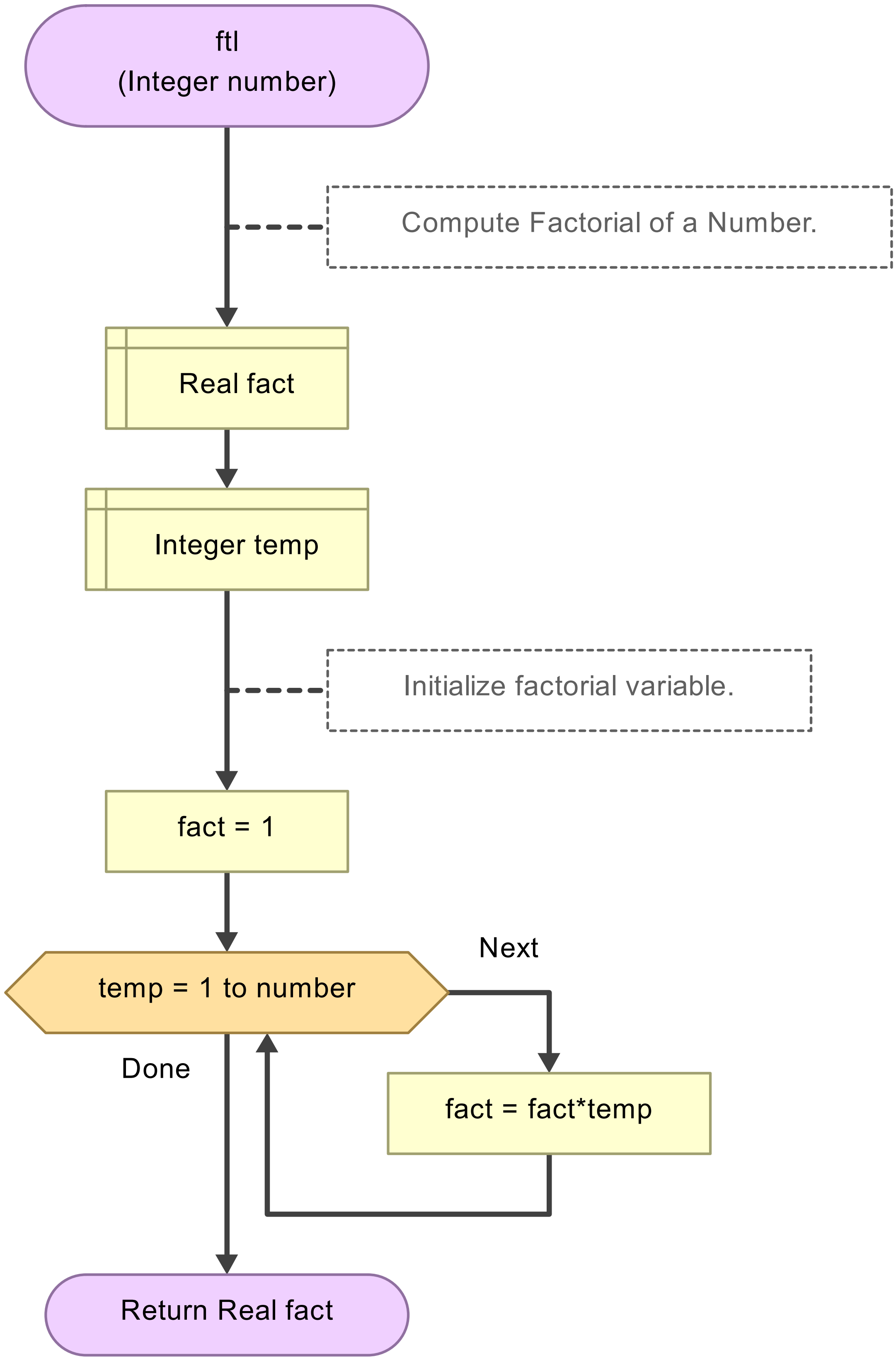
Calculate $y_k = f(a+kh)$ and apply Simpson's 3/8 formula.

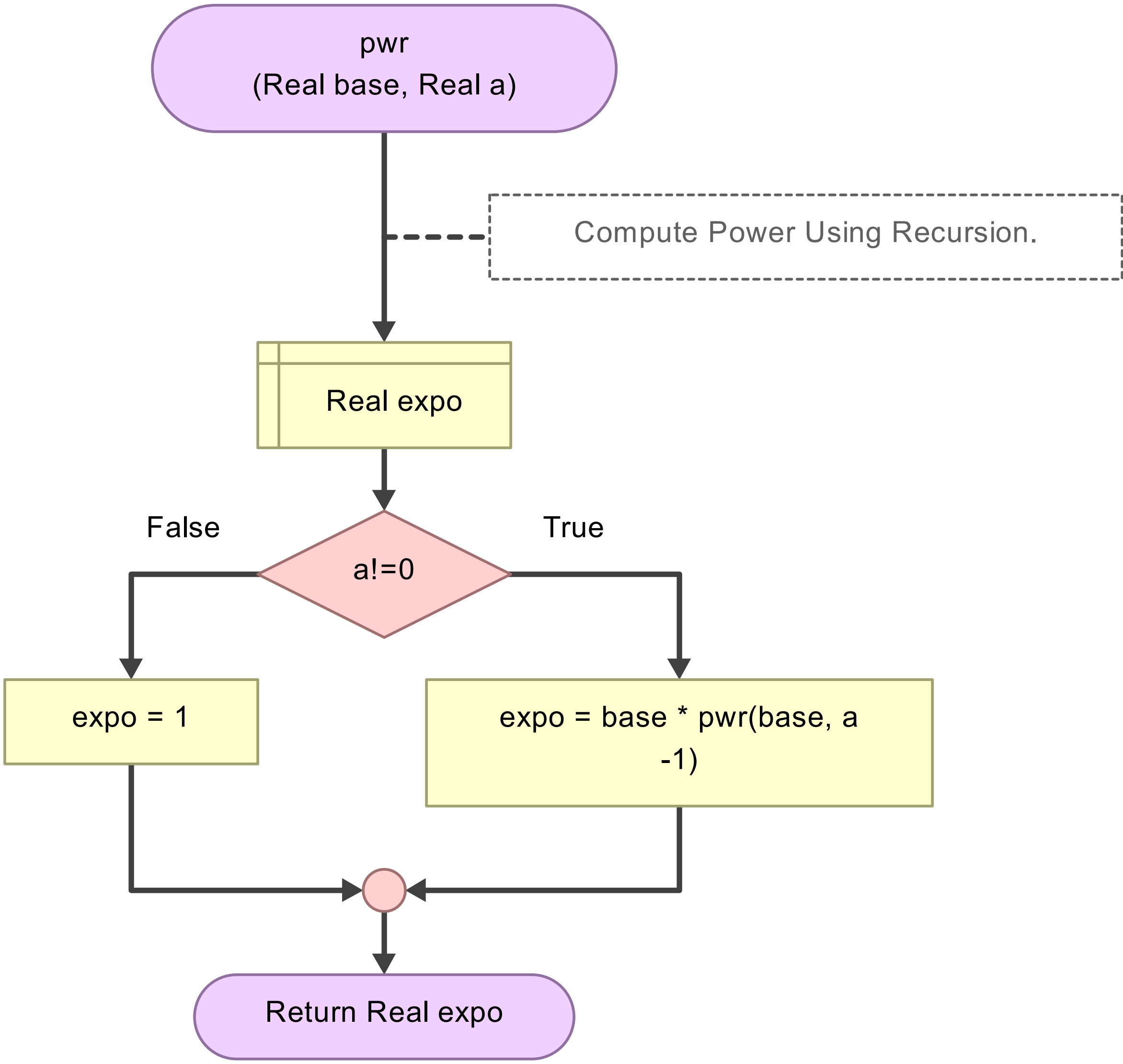
$$\int_a^b f(x)\,dx = \frac{3h}{8}\left[f(a) + 3(y_1 + y_2 + y_4 + y_5 + \cdots + y_{n-1}) + 2(y_3 + y_6 + y_9 + \cdots + y_{n-1}) + f(b) \right]$$

where $y_k = f(a+kh)$; $k = 1, 2, \cdots, n-1$


Main
Perform necessary initializations
Integer numberofterms, numberofintervals, n, i
Real a, b, h, s
Terms of the series
numberofterms = 15
Lower limit of integral
a = 0
Upper limit of integral
b = (2*3.14)/3
Number of integration intervals
numberofintervals = 10
Final array that adds the sums at each iteration
Real Array Integral [numberofterms]
Output "The lower limit has the function value f = sin(" &a& ") = " &f(a)&"."
Output "The upper limit has the function value f = sin(" &b& ") = " &f(b)&"."
Output "Trapezoidal Rule:"
For each interval, get the function value.
n = 1 to numberofintervals
Next
Done
Size of each segment.
h = (b-a)/n
Initial summation of first and last segment.
s = 0.5 * (f(a)+ f(b))
Summation of all intervals.
i = 1 to n-1
Next
Done
s = s + f(a + i*h)
Final obtained integral.
Integral[n] = h * s
Display each iteration.
Output "For term = " &n& ", the sum is: " &Integral[n]
Output "Simpson's 1/3 Rule:"
n = 1 to numberofintervals
Next
Done
Size of each segment.
h = (b-a)/n
Initial summation of first and last segment.
s = f(a)+ f(b)
Summation of all odd intervals.
i = 1 to n-1 step 2
Next
Done
s = s + 4*f(a + i*h)
Summation of all odd intervals.
i = 2 to n-1 step 2
Next
Done
s = s + 2*f(a + i*h)
Final obtained integral.
Integral[n] = h/3 * s
Display each iteration.
Output "For term = " &n& ", the sum is: " &Integral[n]
Output "Simpson's 3/8 Rule:"
n = 1 to numberofintervals
Next
Done
Size of each segment.
h = (b-a)/n
Initial summation of first and last segment.
s = f(a)+ f(b)
Summation of all intervals.
i = 1 to n-1 step 3
Next
Done
s = s + 3*f(a + i*h)
i = 2 to n-1 step 3
Next
Done
s = s + 3*f(a + i*h)
i = 3 to n-1 step 3
Next
Done
s = s + 2*f(a + i*h)
Final obtained integral.
Integral[n] = (3*h/8) * s
Display each iteration.
Output "For term = " &n& ", the sum is: " &Integral[n]
End


f
(Real x)
Taylor Summation Expression sin(x)
Integer numberofterms, i
Real temp
numberofterms = 15
Temporary variable to store the previous value of sum.
temp = 0
i = 0 to numberofterms-1
Next
Done
temp = temp + (pwr(-1, i))*(pwr(x, (2 * i) + 1))/ftl((2 * (i)) + 1)
Return Real temp


ftl
(Integer number)
Compute Factorial of a Number.
Real fact
Integer temp
Initialize factorial variable.
fact = 1
Next
temp = 1 to number
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