```
% Mfile name
%   mtl_aae_sim_approxerror.m

% Version:
%   Matlab R2007a

% Revised:
%   August 28, 2008

% Authors:
%   Luke Snyder, Dr. Autar Kaw
%   University of South Florida
%   kaw@eng.usf.edu
%   Website: http://numericalmethods.eng.usf.edu

% Purpose
%   To illustrate the concept of approximate error, absolute approximate
%   error, relative approximate error and absolute relative approximate
%   error, number of significant digits correct via Maclaurin series of
%   transcendental and trigonometric functions.

% Keywords
%   Approximate Error
%   Relative True Error
%   Significant Digits

% Clearing all data, variable names, and files from any other source and
% clearing the command window after each successive run of the program.
clc
clear all

% Inputs:
%    This is the only place in the program where the user makes the changes
%    based on their wishes
%        Choose the function of your choice by choosing an integer
%        1 for exp(x); 2 for sin(x); 3 for cos(x)
         func_choice=1;

%        Maximum number of terms to use
         n=15;

%        Prespecified tolerance in percentage
         eps=0.01;

%        Value of x at which function is calculated
         x=pi;

% **************************************************************************

disp(sprintf('\n\nConcepts of Error: Approximate Error, Absolute'))
disp(sprintf('Approximate Error, Relative Approximate Error, and'))
disp(sprintf('Absolute Relative Approximate Error'))
disp(sprintf('\nUniversity of South Florida'))
disp(sprintf('United States of America'))
disp(sprintf('kaw@eng.usf.edu'))
disp(sprintf('Website: http://numericalmethods.eng.usf.edu'))
disp(sprintf('\nNOTE: This worksheet illustrates the use of Matlab to find'))
disp('approximate error, absolute approximate error, relative approximate')
disp('error, absolute relative approximate error, and the number of ')
disp('significant digits correct via Maclaurin series of transcendental and')
disp('trigonometric functions.')

disp(sprintf('\n************************Introduction*************************'))

disp(sprintf('The following worksheet demonstrates how to calculate different '))
disp(sprintf('definitions related to approximate error, such as approximate '))
disp(sprintf('error, absolute approximate error, relative approximate error, and '))
disp(sprintf('absolute relative approximate error. The concept is demonstrated '))
disp(sprintf('using an example of a Maclaurin series. The user will choose'))
disp(sprintf('which function to perform the calculation for in the Input section '))
disp(sprintf('of the program. The choices are given as 1 for e^x, 2 for sin(x),'))
disp(sprintf('and 3 for cos(x). The true value of these functions will be'))
disp(sprintf('assumed as given by the Matlab commands for these '))
disp(sprintf('functions.'))

disp(sprintf('\n\n**************************Input Data******************************'))

disp(sprintf('     func_choice = %g',func_choice))
disp(sprintf('     Value of x at which to approximate, x = %g',x))
disp(sprintf('     Maximum number of terms, n = %g',n))
disp(sprintf('     Prespecified tolerance in percentage, eps = %g',eps))

% Using a long format so that all the necessary digits can be shown
format long

disp(sprintf('\n\n**************************Procedure*****************************'))

% Determining which function to use and displaying it in the command
% window. Once function is determined, value is calculated using a
% Maclaurin series in a repetitive loop.

sumprevious = 0;

for i=1:1:n
    if func_choice == 1
        func = ' exp(x)';

        % Using a repetitive loop to calculate the value of function after
        % adding a term.
        sumpresent(i) = sumprevious + (x^(i-1))/(factorial(i-1));
    end

    if func_choice == 2
        func = ' sin(x)';

        % Using a repetitive loop to calculate the value of function after
        % adding a term.
        sumpresent(i) = sumprevious + ((-1)^(i-1))*((x^(2*(i-1)+1))/(factorial(2*i-1)));
    end

    if func_choice == 3
        func = ' cos(x)';

        % Using a repetitive loop to calculate the value of function after
        % adding a term.
        sumpresent(i) = sumprevious + ((-1)^(i+1))*(x^(2*i-2))/(factorial(2*i-2));
    end

    % This line takes into account when i = 1, where
    % approximate errors cannot be calculated as previous approximation
    % does not yet exist.
    if i == 1
        disp(sprintf('\n'));
        str = 'The chosen function to perform calculations on is';
        full = [str,func];
        disp(full);
        disp(sprintf('\nStep %g',i))
        disp(sprintf('-------------------------------------------------------------'))
        disp(sprintf('\n'));
        str1 = 'The value of the function';
        str2 = [' for the first calculation is ', num2str(sumpresent(i)),'.'];
        allstr = [str1,func,str2];
        disp(allstr);
        disp(sprintf('Since only one term has been calculated, relative error '));
        disp(sprintf('calculations are not possible for the first calculation.'))

    end

    % In all other cases where i >= 2, approximate errors can be calculated.
    if i >= 2
        ApproxError(i) = (sumpresent(i) - sumprevious);
        AbsApproxError(i) = abs(sumpresent(i) - sumprevious);
        RelApproxError(i) = ((sumpresent(i) - sumprevious)/sumpresent(i))*100;
        AbsRelApproxError(i) = abs((sumpresent(i) - sumprevious)/sumpresent(i))*100;

        % Calculating the number of correct significant digits that can be
        % used in the final answer.
        SigDigits(i)=floor((2-log10((AbsRelApproxError(i)/100)/0.5)));

        % If SigDigits is less than zero, no significant digits are correct.
        if SigDigits(i) < 0
            SigDigits(i) = 0;
        end

        disp(sprintf('\nStep %g',i));
        disp(sprintf('-----------------------------------------------------------'));
        disp(sprintf('\nEvaluating the function using %g terms',i));
        disp(sprintf('\n     Approximate value of function = %10.10f',sumpresent(i)));
        disp(sprintf('\n     AE(%g) = (%g - %g) = %g',i,sumpresent(i),sumprevious,↙
ApproxError(i)));
        disp(sprintf('     absAE(%g) = |(%g - %g)| * 100 = %g',i,sumpresent(i),↙
sumprevious,AbsApproxError(i)));
```

```
        disp(sprintf('     RAE(%g) = ((%g - %g)/%g) * 100 = %g',i,sumpresent(i),↙
sumprevious,sumpresent(i),RelApproxError(i)));
        disp(sprintf('     absRAE(%g) = |((%g - %g)/%g)| * 100 = %g',i,sumpresent(i),↙
sumprevious,sumpresent(i),AbsRelApproxError(i)));
        disp(sprintf('     SD(%g) = floor((2-log10((%g/100)/0.5))) = %g',i,↙
AbsRelApproxError(i),SigDigits(i)));

    end

    % Evaluating the Absolute Relative Approximate Error and comparing
    % with the prespecified tolerance to determine if sufficient iterations
    % have been performed.
    if i >= 2
    if AbsRelApproxError(i) < eps
        disp(sprintf('\n*************************Results***************************'));
        disp(sprintf('\nThe prespecified tolerance has been met.  No more iterations are↙
necessary.'));
        disp(sprintf('\nThe Absolute Relative Approximate Error was calculated as, absRAE↙
= %g,',AbsRelApproxError(i)));
        disp(sprintf('which is less than the prespecified tolerance, eps = %g.',eps));
        str1 = 'The value of the function ';
        str2 = [' at x = ',num2str(x),' is ', num2str(sumpresent(i),'%10.10f'),'.'];
        allstr = [str1,func,str2];
        disp(allstr);
        disp(sprintf('The number of iterations performed was, i = %g.',i))
        count = i;
        break;
    elseif i==n
        disp(sprintf('\n*************************Results***************************'));
        disp(sprintf('\nThe number of terms used were not sufficient to meet'))
        disp(sprintf('the prespecified tolerance.  The absolute relative approximate'))
        disp(sprintf('error was calculated as AbsRelApproxError = %g, which is not less ',↙
AbsRelApproxError(i)))
        disp(sprintf('than the prespecified tolerance eps = %g.  More terms are↙
required.',eps))
        str1 = 'The approximate value of the function';
        str2 = [' at x = ',num2str(x),' is calculated as '] ;
        str3 = [num2str(sumpresent(i),'%10.10f'),'.'];
        allstr = [str1,func,str2,str3];
        disp(allstr);
        count = n;
    end
    end

    %This line reinitializes xold as xnew which allows for the next
    %sequence of calculations to take place.
    sumprevious = sumpresent(i);
end

% Creating a table of values based on the error calculations previously
% performed.
disp(sprintf('\n\n****************************************Table of↙
Values****************************************'));
disp('Terms       Approx        Approx          Absolute          Relative          Abs↙
Rel     Sig' );
disp('Used        Soln          Error           Approx              Error               Approx↙
Digits');
for i=1:1:count
    if i == 1
        disp(sprintf('%g           %g              N/A                N/A                N/A↙
N/A          0 ',i,sumpresent(i)));
    end
    if i >= 2
        string = '%g        %+1.3e    %+1.3e    %+1.3e      %+1.3e       %+1.3e      %2i';
        disp(sprintf(string,i,sumpresent(i),ApproxError(i),AbsApproxError(i),↙
RelApproxError(i),AbsRelApproxError(i),SigDigits(i)))
    end
    iteration(i) = i;
end

% Plotting the Approximate value of function as a function of the number of
% terms.  The true value as given by Matlab function is also known.
% Matlab functions
if func_choice == 1
    TrueVal = exp(x);
end
if func_choice == 2
    TrueVal = sin(x);
end
if func_choice == 3
    TrueVal = cos(x);
end

clf
% Graph 1: Maclaurin series f(x) vs True value as a function of number of terms.
figure(1)
plot(iteration,sumpresent,'r','LineWidth',1);
hold on
xx = 0:0.01:count;
plot(xx,TrueVal,'g','LineWidth',3);
hold off
if func_choice == 1
    axis([1 count (TrueVal - TrueVal) (TrueVal + TrueVal)])
else
    axis([1 count -1.5  1.5]);
end
title('\bfCalculated Value of f(x) Using Maclaurin Series as a Function of Number of↙
Terms');
xlabel('\bfNumber of Terms Used');
ylabel('\bff(x)');
legend('Calculated Value','True Value');

% Using Matlab's ability to position plots exactly where the user specifies.
scnsize = get(0,'ScreenSize');

% Graph 2: Approximate and Absolute Approximate Errors
fig2 = figure(2);
% First Subplot: Approximate Error vs. Number of Terms Calculated
```

```
set(fig2,'Position',[0.15*scnsize(3),0.45*scnsize(3),0.7*scnsize(3),0.35*scnsize(4)])
subplot(1,2,1);
plot(iteration,ApproxError)
title('\bfApproximate vs. Number of Terms ');
xlabel('\bfNumber of Terms Used')
ylabel('\bfApproximate Error');
axis([2 count min(ApproxError) max(ApproxError)])
% Second Subplot: Absolute Approximate Error vs. Number of Terms Calculated
subplot(1,2,2);
plot(iteration,AbsApproxError);
title('\bfAbs Approximate vs. Number of Terms Used');
xlabel('\bfNumber of Terms Used');
ylabel('\bfAbsolute Approximate Error');
axis([2 count min(AbsApproxError) max(AbsApproxError)])

% Graph 3: Relative and Absolute Approximate Errors
fig3 = figure(3);
% First Subplot: Relative Approximate Error vs. Number of Terms Calculated
set(fig3,'Position',[0.15*scnsize(3),0.1*scnsize(3),0.7*scnsize(3),0.35*scnsize(4)]);
subplot(1,2,1);
plot(iteration,RelApproxError);
title('\bfRel Approximate Error vs. Number of Terms Used');
xlabel('\bfNumber of Terms Used');
ylabel('\bfRelative Approximate Error');
axis([2 count min(RelApproxError) max(RelApproxError)])
% Second Subplot: Absolute Relative Approximate Error vs. Number of Terms
% Calculated.
subplot(1,2,2);
plot(iteration,AbsRelApproxError);
title('\bfAbs Rel Approximate Error vs. Number of Terms Used');
xlabel('\bfNumber of Terms Used');
ylabel('\bfAbsolute Relative Approximate Error');
axis([2 count min(AbsRelApproxError) max(AbsRelApproxError)])

% Figure 4: Number of Significant digits at least correct as a function of
% number of terms used.
figure(4)
set(gca,'YTick',0:1:SigDigits(count))
bar(iteration,SigDigits);
title('\bfNumber of Significant Digits at Least Correct as Function of Number of Terms↙
Used');
xlabel('\bfNumber of Terms Used');
ylabel('\bfNumber of Significant digits at Least Correct');
yuplim = SigDigits(count) + 1;
xuplim = count + 1;
axis([0 xuplim 0 yuplim]);


disp(sprintf('\n\n*************************Conclusion*****************************'))
disp(sprintf('This worksheet shows how the number of terms taken in a Maclaurin'))
disp(sprintf('series affects the accuracy of the calculated answer through the'))
disp(sprintf('analysis of error. Note that though approximate error shows the'))
disp(sprintf('magnitude of the error, it does not indicate how bad the error'))
disp(sprintf('really is. Hence, relative approximate error is used here to give'))
disp(sprintf('a more complete picture of the state of error.'))


disp(sprintf('\n\n***************************Refrences******************************'))
disp('See: <a href = "http://numericalmethods.eng.usf.↙
edu/mtl/gen/01aae/mtl_gen_aae_txt_measuringerror.pdf">Measuring Errors</a>')

disp(sprintf('\n\nLegal Notice: The copyright for this application is owned'))
disp(sprintf('by the author(s). Neither MathWorks nor the author(s)'))
disp(sprintf('are responsible for any errors contained within and are '))
disp(sprintf('not liable for any damages resulting from the use of this'))
disp(sprintf('material. This application is intended for non-commercial,'))
disp(sprintf('non-profit use only. Contact the author for permission if'))
disp(sprintf('you wish to use this application in for-profit activities.'))
```