ELEN 100L (Electric Circuits II): Project 1, YourName

Table of Contents

Initialize MATLAB Environment 2
Setup global variables 2
Problem 3 2
Problem 4 6
Problem 5 6
Problem 6 8
Program execution complete 12
MATLAB code listing 12

Project 1: Passive Filter Design

The circuit below

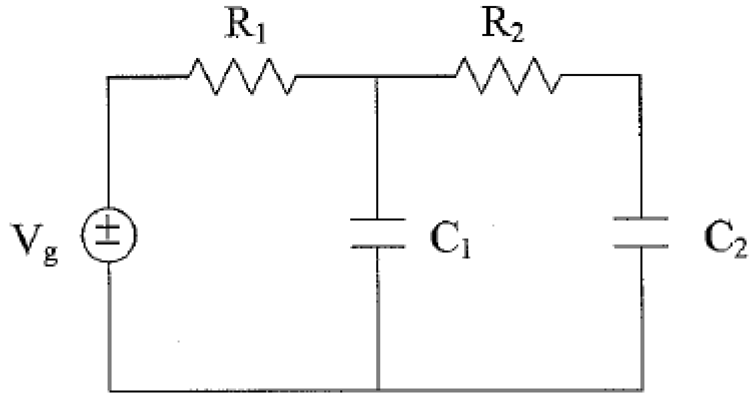

Fig. 1. A passive filter.

it driven by a sinusoidal voltage source of the form $v_g(t) = \cos \omega t$.

Hard Copy Deliverables:

1. Hard copy for hand calculations.
2. A MATLAB script and publish the solution using MATLAB's **publish** feature.
3. Turn in MATLAB scripts and a document of the run-time results.
4. Turn in the Excel file with the measured results.

Soft Copy Deliverables:

1. Turn in MATLAB files.
2. Turn in LTSpice files.
3. Turn in the Excel file with the measured results.

Initialize MATLAB Environment

```
clear; clc; clf; cla; close all;
format long; format compact;
```

Setup global variables

```
% These Ideal Design element values are fixed in the circuit.
VG  =  ?;                      % Generator voltage
R1_ideal =      ?          ;    % Ohms
R2_ideal =      ?          ;    % Ohms
C1_ideal =      ?          ;    % Farads
C2_ideal =      ?          ;    % Farads

% These Actual Design element values are fixed in the circuit.
R1_actual =    ?       ;   % Ohms
R2_actual =    ?       ;   % Ohms
C1_actual =    ?       ;   % Farads
C2_actual =    ?       ;   % Farads

% Setup values for the poles.
w0 =     ?   ;        % Radians/Second
w1 =     ?   ;        % Radians/Second
f0 =     ?   ;        % Hertz
f1 =     ?   ;        % Hertz

% Build an array for the angular frequency and convert it to Hertz.
dw = 10;                   % Step size for analysis
w = [1:dw:w0-dw, ...
     w0, ...
     w0+dw:dw:w1-dw, ...
     w1, ...
     w1+dw:dw:1.0e6];      % Radians/Second (ensure poles are
 included)

 f =      w/(2*pi)     ;            % Hertz

% These values are used for plotting purposes.
fignum = 1;
plot_left   = 1;       plot_right = 2e5;    % x-axis range (Hertz)
plot_bottom = -90;     plot_top   = 5;      % y-axis range (dB)
```

Problem 3

```
fignum = fignum+1;
```

Display the component values for the Ideal and Actual designs.

```
display(' ');
display('The Ideal Design component values are:');
fprintf('    R1 = %+11.4f Ohms.\n',  ?  );
fprintf('    R2 = %+11.4f Ohms.\n',  ?  );
fprintf('    C1 = %+11.4e Farads.\n', ?  );
fprintf('    C2 = %+11.4e Farads.\n', ?  );

display(' ');
display('The Actual Design component values are:');

fprintf('    R1 = %+11.4f Ohms.\n',  ?  );
fprintf('    R2 = %+11.4f Ohms.\n',  ?  );
fprintf('    C1 = %+11.4e Farads.\n', ? );
fprintf('    C2 = %+11.4e Farads.\n', ? );
```

Compute the percent differences between the Ideal and Actual design component values.

```
diff_R1_ideal_actual = ( ?? )/abs(R1_ideal)*100;
diff_R2_ideal_actual = ( ?? )/abs(R2_ideal)*100;
diff_C1_ideal_actual = ( ?? )/abs(C1_ideal)*100;
diff_C2_ideal_actual = ( ?? )/abs(C2_ideal)*100;

display(' ');
display('The percent difference between Ideal and Actual design');
display('component values:');
fprintf('    %% diff R1 = %+8.4f (%%).\n', diff_R1_ideal_actual );

fprintf('    %% diff R2 = %+8.4f (%%).\n', ? );
fprintf('    %% diff C1 = %+8.4f (%%).\n', ? );
fprintf('    %% diff C2 = %+8.4f (%%).\n', ? );
```

Display the poles for the target circuit design transfer function.

```
display(' ');
display('The poles for the circuit are:');
fprintf('    w0 = %+11.4f Radians/Second.\n',w0);
fprintf('    w1 = %+11.4f Radians/Second.\n',w1);
fprintf('    f0 = %+11.4f Hertz.\n',f0);
fprintf('    f1 = %+11.4f Hertz.\n',f1);
```

Setup the matrices used to generate the Bode plots for the Ideal and Actual designs.

```
G1_ideal = [ ...
      (1)            (0)                          (0); ...
      (-1/R1_ideal)  (1/R1_ideal + 1/R2_ideal)    (-1/R2_ideal); ...
      (0)            (-1/R2_ideal)                (1/R2_ideal)];

G2_ideal = [ ??? ];

G3_ideal = [ ??? ];

G1_actual = [ ??? ];
```

```
G2_actual = [ ??? ];

G3_actual = [ ??? ];

B = [ ??? ];
```

Locate the poles in the frequency vector for plotting purposes.

```
% Find the pole values.
pole_1 = 0;
for iter = 1:length(f)             % Locate the first pole
    if (f(iter) == f0)
        pole_1 = iter;
        break;
    end;
end;

pole_2 = 0;
for iter = pole_1+1:length(f)      % Locate the second pole
    if (f(iter) == f1)
        pole_2 = iter;
        break;
    end;
end;
```

Calculate the frequency response for the Ideal and Actual designs.

```
Hw_ideal   = proj1E100_freqresp( ?,?,?,?,?,? );
Hw_actual  = proj1E100_freqresp( ?,?,?,?,?,? );

% Capture the values at the poles.
Hw_ideal_f0  = Hw_ideal(pole_1);
Hw_ideal_f1  = Hw_ideal(pole_2);

Hw_actual_f0 = Hw_actual(pole_1);
Hw_actual_f1 = Hw_actual(pole_2);
```

Generate the plot for $H_{ideal}(f)$ and indicate where the two poles occur.

```
fignum = fignum+1; figObj = figure(fignum);  % Establish a figure
 number
set(fignum, 'Name',['H(f) Ideal Design']);   % Name the figure

Hw_ideal_Plot = semilogx( f, Hw_ideal ,'-r');          % Generate plot
grid on;                                     % Turn grid on
xlabel('Frequency (Hz)');                    % Label the x-axis
ylabel('Amplitude (dB)');                    % Label the y-axis
axis([plot_left, plot_right, ...
      plot_bottom, plot_top]);               % Bound plot
title(['Figure ',num2str(fignum,'%-2.u'),...
       ': H_i_d_e_a_l(f)']);
legend('H_i_d_e_a_l(f)', 'Location', 'NorthEast');

% Add cursors to the plot.
```

```
makedatatip(Hw_ideal_Plot, [pole_1; pole_2]);
```

Generate the plot for $H_{actual}(f)$ and indicate where the two poles occur.

```
fignum = fignum+1; figObj = figure(fignum);  % Establish a figure
 number
set(fignum, 'Name',['H(f) Actual Design']);  % Name the figure

Hw_actual_Plot = semilogx( ? , ? ,'-b');         % Generate plot
grid on;                                     % Turn grid on
xlabel('Frequency (Hz)');                    % Label the x-axis
ylabel('Amplitude (dB)');                    % Label the y-axis
axis([plot_left, plot_right, ...
      plot_bottom, plot_top]);               % Bound plot
title(['Figure ',num2str(fignum,'%-2.u'),...
       ': H_a_c_t_u_a_l(f)']);
legend('H_a_c_t_u_a_l(f)', 'Location', 'NorthEast');

% Add cursors to the plot.
makedatatip(Hw_actual_Plot, [pole_1; pole_2]);
```

Generate the plot for comparing $H_{ideal}(f)$ and $H_{actual}(f)$.

```
fignum = fignum+1; figObj = figure(fignum);  % Establish a figure
 number
set(fignum, 'Name', ...
    ['H(f) Ideal and Actual Design']);       % Name the figure

Hw_ideal_actual_Plot = ...
    semilogx( ? , ? ,'-r', ...
              ? , ? ,'-b');                      % Generate plot
grid on;                                     % Turn grid on
xlabel('Frequency (Hz)');                    % Label the x-axis
ylabel('Amplitude (dB)');                    % Label the y-axis
axis([plot_left, plot_right, ...
      plot_bottom, plot_top]);               % Bound plot
title(['Figure ',num2str(fignum,'%-2.u'),...
       ': H_i_d_e_a_l(f) and H_a_c_t_u_a_l(f)']);
legend('H_i_d_e_a_l(f)', 'H_a_c_t_u_a_l(f)', 'Location', 'NorthEast');
```

Calculate the percent difference between $H_{ideal}(f)$ and $H_{actual}(f)$ at the two poles.

```
diff_0_ideal_actual = ( ?? )/abs(Hw_ideal_f0)*100;
diff_1_ideal_actual = ( ?? )/abs(Hw_ideal_f1)*100;

display(' ');
display('The difference between Ideal and Actual designs at the
 poles:');
fprintf('    Ideal  Design H(%+10.4f) = %+8.4f (dB).\n', f0,
 Hw_ideal_f0);
fprintf('    Actual Design H(%+10.4f) = %+8.4f (dB).\n', f0,
 Hw_actual_f0);
fprintf('        %% diff = %+8.4f (%%).\n', diff_0_ideal_actual);
```

```
fprintf('    Ideal  Design H(%+10.4f) = %+8.4f (dB).\n', f1, ? );
fprintf('    Actual Design H(%+10.4f) = %+8.4f (dB).\n', f1, ? );
fprintf('        %% diff = %+8.4f (%%).\n', ? );
```

Problem 4

```
fignum = fignum+1;
```

The LTSpice model for the circuit is shown below.

```
fignum = fignum+1;
```

The LTSpice model for the simulation result is shown below.

```
fignum = fignum+1;
```

Calculate the percent difference between $H_{actual}(f)$ and $H_{LTSpice}(f)$ actual designs at the two poles.

```
Hw_ltspice_f0 =     ?      ;      % dB
Hw_ltspice_f1 =     ?      ;      % dB
f0_ltspice =      ?    ;          % Hertz
f1_ltspice =      ?    ;          % Hertz

diff_0_actual_ltspice =  ( ?? )/abs(Hw_actual_f0)*100;
diff_1_actual_ltspice =  ( ?? )/abs(Hw_actual_f1)*100;

display(' ');
display('The percent difference between MATLAB and LTSpice Actual');
display('designs at the poles:');
fprintf('    Actual  MATLAB H(%+10.4f) = %+8.4f (dB).\n', ....
         f0, ? );
fprintf('    Actual LTSpice H(%+10.4f) = %+8.4f (dB).\n', ...
         f0_ltspice, Hw_ltspice_f0);
fprintf('        %% diff = %+8.4f (%%).\n', diff_0_actual_ltspice);

fprintf('    Actual  MATLAB H(%+10.4f) = %+8.4f (dB).\n', ...
         ? , ? );
fprintf('    Actual LTSpice H(%+10.4f) = %+8.4f (dB).\n', ...
         ? , ? );
fprintf('        %% diff = %+8.4f (%%).\n', ? );
```

Problem 5

```
fignum = fignum+1;
```

Vary the Actual design component values and calculate the frequency response for each variation.

```
% Declare the number of component value iterations.
value_sets =  5 ;

% Build the actual component vector
Q_actual = [R1_actual, R2_actual, C1_actual, C2_actual];

% Generate the frequency response values for the specified number of
% iterations.
[Hw_actual_varied, Q_actual_varied] = ...
             proj1E100_freqresp_varied( ?,?,?,?,? );

% Capture the values at the poles.
Hw_actual_varied_f0 = zeros(1,value_sets);
Hw_actual_varied_f1 = zeros(1,value_sets);
for iter = 1:value_sets
    Hw_actual_varied_f0(iter)  = Hw_actual_varied(iter, pole_1);
    Hw_actual_varied_f1(iter)  = Hw_actual_varied(iter, pole_2);
end;
```

Generate the plot for variations in the Actual design component values and display all $H_{varied}(f)$ curves on a single plot.

```
fignum = fignum+1; figObj = figure(fignum);  % Establish a figure
 number
set(fignum, 'Name', ...
    ['H(f) Actual Design Varied']);          % Name the figure

Hw_actual_varied_Plot = ...
    semilogx( ? , ? );            % Generate plot
grid on;                                     % Turn grid on
xlabel('Frequency (Hz)');                    % Label the x-axis
ylabel('Amplitude (dB)');                    % Label the y-axis
axis([plot_left, plot_right, ...
      plot_bottom, plot_top]);               % Bound plot
title(['Figure ',num2str(fignum,'%-2.u'),...
       ': Varied H_a_c_t_u_a_l(f)']);
legend('H_1(f)', 'H_2(f)', 'H_3(f)', 'H_4(f)', 'H_5(f)', ...
       'Location', 'NorthEast');
```

Calculate the percent difference between $H_{varied}(f)$ and $H_{actual}(f)$ at the two poles of each variation.

```
diff_0_actual_varied = ...
    (Hw_actual_varied_f0-Hw_actual_f0)/abs(Hw_actual_f0)*100;
diff_1_actual_varied = ...
    ( ?? )/abs( ? )*100;

display(' ');
display('The difference between Varied and Actual designs at the
 poles:');
for iter = 1:value_sets
    diff_R1_actual_varied = ...
```

```
        (Q_actual_varied(iter,1)-R1_actual)/abs(R1_actual)*100;
    diff_R2_actual_varied = ...
        (Q_actual_varied(iter,2)- ? )/abs( ? )*100;
    diff_C1_actual_varied = ...
        (Q_actual_varied(iter,3)- ? )/abs( ? )*100;
    diff_C2_actual_varied = ...
        (Q_actual_varied(iter,4)-C2_actual)/abs( ? )*100;

    fprintf('    Variation Component Set %-2.u: \n', iter);
    fprintf('        R1 = %+11.4f Ohms,    %% diff = %+8.4f (%%).
\n', ...
            Q_actual_varied(iter,1), diff_R1_actual_varied);
    fprintf('        R2 = %+11.4f Ohms,    %% diff = %+8.4f (%%).
\n', ...
            Q_actual_varied(iter,2), ? );
    fprintf('        C1 = %+11.4e Farads,  %% diff = %+8.4f (%%).
\n', ...
            Q_actual_varied(iter,3), diff_C1_actual_varied);
    fprintf('        C2 = %+11.4e Farads,  %% diff = %+8.4f (%%).
\n', ...
            Q_actual_varied(iter,4), ? );
    fprintf('            Varied Design H(%+10.4f) = %+8.4f (dB).
\n', ...
            f0, Hw_actual_varied_f0(iter));
    fprintf('            Actual Design H(%+10.4f) = %+8.4f (dB).
\n', ...
            f0, Hw_actual_f0);
    fprintf('                %% diff = %+8.4f (%%).\n', ...
            diff_0_actual_varied(iter));
    fprintf('            Varied Design H(%+10.4f) = %+8.4f (dB).
\n', ...
            ? , ?  );
    fprintf('            Actual Design H(%+10.4f) = %+8.4f (dB).
\n', ...
            ? , ?  );
    fprintf('                %% diff = %+8.4f (%%).\n', ...
             ? );
end;
```

Problem 6

```
fignum = fignum+1;
```

Display the measured values for the components used in the Actual design.

```
R1_meas =    ?   ;          % Ohms
R2_meas =    ?   ;          % Ohms
C1_meas =     ?         ;   % Farads
C2_meas =     ?         ;   % Farads

display(' ');
fprintf('Measured component values are:\n');
```

```
fprintf('    R1 = %+11.4f Ohms.\n',  ?  );
fprintf('    R2 = %+11.4f Ohms.\n',  ?  );
fprintf('    C1 = %+11.4e Farads.\n', ?  );
fprintf('    C2 = %+11.4e Farads.\n', ?  );
```

Compute the percent differences between the Measured and Actual design component values.

```
diff_R1_meas_actual =  (R1_meas-R1_actual)/abs(R1_actual)*100;
diff_R2_meas_actual =  ?? ;
diff_C1_meas_actual =  ?? ;
diff_C2_meas_actual =  ?? ;

display(' ');
display('The percent difference between Measured and Actual design');
display('component values:');
fprintf('    %% diff R1 = %+8.4f (%%).\n', diff_R1_meas_actual );

fprintf('    %% diff R2 = %+8.4f (%%).\n', ? );
fprintf('    %% diff C1 = %+8.4f (%%).\n', ? );
fprintf('    %% diff C2 = %+8.4f (%%).\n', ? );
```

Import the measured data for processing.

The measured values for frequency response are contained in the external Excel spreadsheet file named "ELEN100L_Project_1_Measured_Results.xlsx". These measured values are imported into MATLAB at run-time using MATLAB's **Import** feature. For the solution shown below, the initial "import" was executed to generate an external function file which can be called at run-time.

```
[freq_meas,Vg_meas,Vo_meas] = importfile_problem6...
    ('ELEN100L_Project_1_Measured_Results.xlsx','Sheet1',2,81);

% Convert the measured column vectors to single row vectors.
dim = size(freq_meas);   rows = 1;     columns = dim(1,1);
freq_meas = reshape(freq_meas, rows, dim(1,1));
Vg_meas   = reshape(Vg_meas  , rows, dim(1,1));
Vo_meas   = reshape(Vo_meas  , rows, dim(1,1));
```

Locate the poles in the frequency vector for plotting purposes.

```
converge_criteria = 1.0;

% Find the pole values.
pole_1_meas = 0;
for iter = 1:length(freq_meas)            % Locate the first pole
    if (abs(freq_meas(iter) - f0) <= converge_criteria)
        pole_1_meas = iter;
        break;
    end;
end;

pole_2_meas = 0;
for iter = pole_1_meas+1:length(freq_meas)     % Locate the second
 pole
    if (abs(freq_meas(iter) - f1) <= converge_criteria)
        pole_2_meas = iter;
```

```
        break;
    end;
end;
```

Calculate the frequency response for the Measured Actual design.

```
Hw_measured_actual = ?? ;  % |H(w)| in decibels (dB) is a function
% of Vo_meas and Vg_meas

% This section of code is used to generate an expected frequency
 response
% based upon the measured component values.
G1_sim_meas_actual = [ ?? ];

G2_sim_meas_actual = [ ?? ];

G3_sim_meas_actual = [ ?? ];

Hw_sim_meas_actual = ...
    proj1E100_freqresp( ? , ? ,...
                        ? , ? , ? , ?);

% Capture the values at the poles.
Hw_meas_actual_f0 = Hw_meas_actual(pole_1_meas);
Hw_meas_actual_f1 = Hw_meas_actual(pole_2_meas);

Hw_sim_meas_actual_f0 = Hw_sim_meas_actual(pole_1_meas);
Hw_sim_meas_actual_f1 = Hw_sim_meas_actual(pole_2_meas);
```

Generate the plot for $H_{measured}(f)$ and indicate where the two poles occur.

```
fignum = fignum+1; figObj = figure(fignum);  % Establish a figure
 number
set(fignum, 'Name', ...
    ['H(f) Measured Actual Design']);        % Name the figure

Hw_meas_actual_Plot = semilogx(...
          ? , ? ,'-m');                          % Generate plot
grid on;                                     % Turn grid on
xlabel('Frequency (Hz)');                    % Label the x-axis
ylabel('Amplitude (dB)');                    % Label the y-axis
axis([plot_left, plot_right, ...
      plot_bottom, plot_top]);               % Bound plot
title(['Figure ',num2str(fignum,'%-2.u'),...
       ': H_m_e_a_s_u_r_e_d(f)']);
legend('H_m_e_a_s_u_r_e_d(f)', 'Location', 'NorthEast');

% Add cursors to the plot.
makedatatip(Hw_meas_actual_Plot, [pole_1_meas; pole_2_meas]);
```

Generate the plot for $H_{simulatemeasured}(f)$ and indicate where the two poles occur.

```
fignum = fignum+1; figObj = figure(fignum);  % Establish a figure
 number
```

```
set(fignum, 'Name', ...
  ['H(f) Simulate Measured Actual Design']); % Name the figure

Hw_sim_meas_actual_Plot = semilogx(...
          ? , ? ,'-g');                           % Generate plot
grid on;                                      % Turn grid on
xlabel('Frequency (Hz)');                     % Label the x-axis
ylabel('Amplitude (dB)');                     % Label the y-axis
axis([plot_left, plot_right, ...
      plot_bottom, plot_top]);                % Bound plot
title(['Figure ',num2str(fignum,'%-2.u'),...
       ': H_s_i_m_ _m_e_a_s(f)']);
legend('H_s_i_m_ _m_e_a_s(f)', 'Location', 'NorthEast');

% Add cursors to the plot.
makedatatip(Hw_sim_meas_actual_Plot, [pole_1_meas; pole_2_meas]);
```

Generate the plot for comparing $H_{measured}(f)$, $H_{simulatemeasured}(f)$ and $H_{actual}(f)$.

```
fignum = fignum+1; figObj = figure(fignum);  % Establish a figure
 number
set(fignum, 'Name', ...
    ['H(f) Measured and Actual Design']);    % Name the figure

semilogx( ? , ? ,'-b',...
          ? , ? ,'-m', ...
          ? , ? ,'-g');                           % Generate plot
grid on;                                      % Turn grid on
xlabel('Frequency (Hz)');                     % Label the x-axis
ylabel('Amplitude (dB)');                     % Label the y-axis
axis([plot_left, plot_right, ...
      plot_bottom, plot_top]);                % Bound plot
title(['Figure ',num2str(fignum,'%-2.u'),...
       ': H_a_c_t_u_a_l(f), H_m_e_a_s(f), and H_s_i_m_ _m_e_a_s(f)']);
legend('H_a_c_t_u_a_l(f)', 'H_m_e_a_s(f)', ...
       'H_s_i_m_ _m_e_a_s(f)',...
       'Location', 'NorthEast');
```

Calculate the percent difference between $H_{measured}(f)$, $H_{simulatemeasured}(f)$, and $H_{actual}(f)$ actual designs at the two poles.

```
f0_measured = freq_meas(pole_1_meas);         % Hertz
f1_measured = freq_meas(pole_2_meas);         % Hertz

diff_0_meas_actual = ...
    (Hw_meas_actual_f0-Hw_actual_f0)/abs(Hw_actual_f0)*100;
diff_1_meas_actual =  ?? ;

diff_0_sim_meas_actual = ?? ;
diff_1_sim_meas_actual = ?? ;

display(' ');
display('The percent difference between MATLAB and Measured Actual');
```

```
display('designs at the poles:');
fprintf('    Actual     MATLAB H(%+10.4f) = %+8.4f (dB).\n', ....
        f0, Hw_actual_f0);
fprintf('    Actual   Measured H(%+10.4f) = %+8.4f (dB).\n', ...
        f0_measured, Hw_meas_actual_f0);
fprintf('        %% diff = %+8.4f (%%).\n', diff_0_meas_actual);
fprintf('    Actual     MATLAB H(%+10.4f) = %+8.4f (dB).\n', ....
        f1, Hw_actual_f1);
fprintf('    Actual   Measured H(%+10.4f) = %+8.4f (dB).\n', ...
        f1_measured, Hw_meas_actual_f1);
fprintf('        %% diff = %+8.4f (%%).\n', diff_1_meas_actual);

display(' ');
display('The percent difference between MATLAB and the simulated');
display('Measured Actual designs at the poles:');
fprintf('    Actual     MATLAB H(%+10.4f) = %+8.4f (dB).\n', ....
        f0, Hw_actual_f0);
fprintf('    Simulate Measured H(%+10.4f) = %+8.4f (dB).\n', ...
        f0_measured, Hw_sim_meas_actual_f0);
fprintf('        %% diff = %+8.4f (%%).\n', diff_0_sim_meas_actual);
fprintf('    Actual     MATLAB H(%+10.4f) = %+8.4f (dB).\n', ...
        f1, Hw_actual_f1);
fprintf('    Simulate Measured H(%+10.4f) = %+8.4f (dB).\n', ...
        f1_measured, Hw_sim_meas_actual_f1);
fprintf('        %% diff = %+8.4f (%%).\n', diff_1_sim_meas_actual);
```

Program execution complete

```
display(' ');
disp('Program execution complete....');
```

MATLAB code listing

Published with MATLAB® R2015b