# MT360_UR7_2_12_1
# Transients on T-lines

## Table of Contents



# Title

```
Authors: Ulaby and Ravaioli
Book Edition: 7th
Problem #:
Last Updated: 1/26/2016
```

# Description

This assignment is designed to help you become familiar with the DC V transient response. In this assignment you will be working with a transmission line that has a length of 1 meter, and with a characteristic impedance of 50 Ohms. The signal sent across the transmission line has a phase velocity of 0.8c, with c being the speed of light.

This is an interactive assignment. Most of the code if provided; however to complete the assignment you will need to change parameters and study what affect the load impedance, and source impedance have on the settling time (the time required for the signal to settle within a certain tolerance percentage of it's final value).

The simulation is performed inside of a while-loop that continues until the the signal settles within the tolerance level from the load's perspective or the user inputs 'crtl+c' in the command window.

# Exercise

Note that the tolerance level is calculated from the perspective of the generator.

1. Finish writing the program by inserting the proper code in the spaces indicated by "INSERT CODE HERE".

2. The source and load impedances have been initially set to 50 Ohms. Run the simulation and note the number of times the signal traverses the T-line before settling within the tolerance level.

3. Set the load reflection coefficient to 1 in order to simulate an open-circuit. Run the simulation and note the number of times the signal traverses the T-line before settling within the tolerance level from the perspective of the load and generator.

4. Set the load reflection coefficient to -1 in order to simulate a short-circuit. Run the simulation and note the number of times the signal traverses the T-line before settling within the tolerance level from the perspective of the load and generator.

5. Set the generator impedance to 10 Ohms and load impedance to 50 Ohms. Run the simulation and note the number of times the signal traverses the T-line before settling within the tolerance level from the perspective of the load and generator.

6. Set the generator impedance to 50 Ohms and load impedance to 10 Ohms. Run the simulation and note the number of times the signal traverses the T-line before settling within the tolerance level from the perspective of the load and generator.

7. Set the generator impedance to 10 Ohms and load impedance to 10 Ohms. Run the simulation and note the number of times the signal traverses the T-line before settling within the tolerance level.

8. Experiment on your own by changing parameters: Zo, Zl, Zg, tolerance etc.

# Questions

1. From the load's perspective, does an open circuit or a closed circuit settle faster?

2. From the generator's perspective, does the signal settle faster when the load impedance matches or the generator impedance matches the line impedance?

3. From the load's perspective, does the signal settle faster when the load impedance matches or the generator impedance matches the line impedance?

4. When both the load and generator impedances are mismatched (are not equivalent to the line impedance), what affect does it have on the settling time from either the generator's or load's perspective?

# Useful Information

Currently, there is no useful information provided to you.

# Provided Code

```
clear all
param

% Parameters
Zo = 50;          % Characteristic Impedance of the line, Ohms
Vg = 1;           % Generator voltage, V
Zg = 50;          % Generator impedance, Ohms
Zl = 50;          % Load impedance, Ohms
l = 1;            % Transmission line length, m
Up = 0.80*P.c;    % Phase velocity, m/s.
T = l/Up;         % Time it takes for the signal to travel the length
                % of the transmission line.

% You will need to be able to calculate the load reflection
 coefficient at
% the load as a function of the load and line impedances as well as
```

```
% specifying the value when the line terminates in an open or short.
gammaL = 1;               % Reflection coefficient at the load. Use
 this
                        % gammaL when working with short or open T-
line.
gammaL = (Zl-Zo)/(Zl+Zo); % Reflection coefficient at the load. Use
 this
                        % gammaL when working with load and line
                        % impedances.
gammaG = % INSERT CODE HERE; % Reflection coefficient at the generator
 in
                           % terms of the generator and line
 impedance.

I1_plus = Vg/(Zg+Zo); % Initial current, A
V1_plus = % INSERT CODE HERE; % Initial voltage, Ohms

Vf = % INSERT CODE HERE; % Voltage at t = infinity in terms of load
 and
                         % generator reflection coefficients, V

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%                    The following Code is given to you
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

tolerance = 0.05;      % This term indicates the desired percent error
                     % between the final voltage and the current
 voltage
                     % at the generator.
error = 1;             % Indicates the current percent error between
 the final
                     % voltage and the current voltage from the
 generator's
                     % perspective with 1 indicating 100 percent error.
N = 0;                 % Keeps count of the number of times the signal
 traverses
                     % the transmission line.
t = 0;                 % Keeps track of the elapsed time,s.

Vc_plus = V1_plus;     % The additional voltage added to the signal
 when it
                     % propagates forwards, towards the load, V.
Vc_minus = 0;          % The additional voltage added to the signal
 when it
                     % propagates in the reverse direction, towards the
                     % generator, V.

% The following variables are used for the simulation.
count = 0;                     % Counts the number of iterations in
                             % the while-loop.
factor = 100;                  % Divides the length of the T-line into
 this
                             % many points.
```

```
time_step = T/factor;          % The represented time elapsed during
 each
                             % iteration,s.
Tlength= linspace(0,1,factor);% The array represents the length of the
                             % transmission line normalized to 1. That
 is,
                             % the length of the T-line is 1, and
 divided
                             % into 'factor' segments.
V = zeros(1,factor);           % Stores the current voltage at each
 point on the T-line.
forward = 1;                   % Keeps track of the direction of the
 transient signal. 1
                             % indicates that the signal is moving
 towards
                             % the load, and -1 indicates that the
 signal
                             % is moving towards the generator.

figure(1), clf;                % Create the figure.
p_handle = plot(Tlength,V);    % Create a handle to the plot
axis([0 1 -Vg Vg])             % Set the axes' limits
lineColor = 'b';               % Current color of the line
xlabel('Voltage(l)')
ylabel('Position on T-line');

% The while loop displays Transients of a transmission line.
while (error > tolerance) % The while loop continues until the error
 is within
                         % tolerance.

  %
  lc = mod(count,factor); % Used for indexing the array V.

  %
  % Updates the current voltage
  if forward == 1 % Signal is propagating towards the load.
      V(lc+1) = V(lc+1) + Vc_plus;
  else % Signal is propagating towards the generator.
      V(factor - lc) = V(factor - lc) + Vc_minus;
  end

  %
  % After the signal has traversed the length of the T-line the
  % direction, error, Vc_minus, Vc_plus N will be updated.
  if(lc == factor-1) % Update occurs when the signal reaches the load.
      forward = forward * -1;
      if forward == -1
          Vc_minus = Vc_plus*gammaL;
          lineColor = 'r';
      else
          Vc_plus = Vc_minus*gammaG;
          error = abs(error*gammaG*gammaL);
          lineColor = 'b';
```

```
        end
        N = N+1;
    end
                                %
    pause(0.01);                % Pauses during each iteration, s.
    count = count + 1;          % Update count every iteration.
    t = t+time_step;            % Update the current time.
    set(p_handle, 'YData',V,'color',lineColor); % Update the plot.
```

end

```
% Displays the final data.

info = {['gammaG = ', num2str(gammaG)];
['gammaL = ', num2str(gammaL)];
['Vf = ', num2str(Vf)];
['VL = ', num2str(V(length(V)))];
['t = ', num2str(t)];
['N = ',num2str(N)]};

disp(info);
```

# Solution

*Published with MATLAB® R2015a*