Laboratory 1 : Voltage and Current on a Lossless Transmission Line

In this laboratory, we (meaning you) will use a numerical model for the behaviour of harmonic wave voltage $v(z,t)$ and current $i(z,t)$ along a lossless transmission line versus position z and time t.

You are to use MATLAB for this laboratory. MATLAB, short for "matrix laboratory", is an easy-to-use environment for numerical calculations that allows scripting and is particularly well-suited for matrix and vector calculations. For instructions on various options for accessing MATLAB via the Department of Electrical and Computer Engineering at McGill University, please see:
www.mcgill.ca/ece/department/it-and-technical-services/it-faqs
mcgill.service-now.com/itportal/

There are a wide variety of resources for learning how to use MATLAB. For help on the definition or use of a function called `name`, you may enter `help name` at the command line. For example, `help help` gives you information on the `help` function. Detailed information about MATLAB functions are available at the MATLAB website: www.mathworks.com

See appendices A and B at the end of this document on how to create your own MATLAB function and script.

1. Instantaneous forward and backward waves along a lossless transmission line

Write two MATLAB functions that give the instantaneous voltage and current for a forward wave and a backward wave on a lossless transmission line,

```
[ vf if ]=forward_wave(v0,omega,phi,Z0,vp,z,t)

[ vb ib ]=backward_wave(v0,omega,phi,Z0,vp,z,t)
```

using the following equations for instantaneous voltage and current on a lossless transmission line,

$$v_f(z,t) = v_0 \cos\left(\omega\left(t - \frac{z}{v_p}\right) + \varphi\right) \qquad i_f(z,t) = \frac{v_0}{Z_0}\cos\left(\omega\left(t - \frac{z}{v_p}\right) + \varphi\right)$$

$$v_b(z,t) = v_0 \cos\left(\omega\left(t + \frac{z}{v_p}\right) + \varphi\right) \qquad i_b(z,t) = -\frac{v_0}{Z_0}\cos\left(\omega\left(t + \frac{z}{v_p}\right) + \varphi\right)$$

where v_0 is the voltage amplitude, ω is the angular frequency, φ is the phase, Z_0 is the characteristic impedance of the lossless line, v_p is the phase velocity of the lossless line, z is the position on the line, and t is the time.

2. Movie of a forward wave and a backward wave

Consider the case of v_0 = 1 V, ω = 2 π x 10^9 Hz, φ = 0 rad, Z_0 = 50 Ω and v_p = 2 x 10^8 m/s. Confine your numerical studies to a segment of transmission line 0 ≤ z ≤ 0.40 m over a duration of time 0 ≤ t ≤ 5 ns.

Using your functions above, write a MATLAB script that creates a movie for the forward going voltage wave and backward going voltage wave. To do this, create a vector `z` of distance values, a vector `t` of time values, a matrix `vf` of forward voltage wave values and a matrix `vb` of backward voltage wave values. Your variables should be defined such that `vf(m,n)` corresponds to the forward voltage at position `z(m)` and time `t(n)`. Be sure that you have sufficiently many points of distance and time to capture the wave propagation accurately in your movie. The script segment below can be used to create a movie from your values of `z`, `t`, `vf` and `vb`.

```
for k=1:length(t)
   plot(z, vf(k,:),'b'); hold on;
   plot(z, vb(k,:),'r'); hold off;
   xlabel('z [m]');
   ylabel('v(z,t)[V]');
   title('instantaneous voltage on a lossless line');
   legend('v_f(z,t)',' v_b(z,t)');
   axis([0 4 -1 1]);
   M(k)=getframe;
end;
```

To replay the movie, execute `movie(M)` at the command line.

№ 1: Show your results to the teaching assistant.

An easy way to save a Graphics Interchange Format (GIF) file of your movie is the `movie2gif(M,'file')` function, available for download here:
www.mathworks.com/matlabcentral/fileexchange/17463-movie-to-gif-converter

You may, optionally, wish to make a movie of the corresponding forward and backward current waves.

3. Movie of a standing wave

Consider again the case of v_0 = 1 V, ω = 2 π x 10^9 Hz, φ = 0 rad, Z_0 = 50 Ω and v_p = 2 x 10^8 m/s. Again, confine your numerical studies to a segment of transmission line $0 \le z \le 0.40$ m over a duration of time $0 \le t \le 5$ ns.

Using your functions above, write a MATLAB script that creates a movie of the voltage and current that result from the simultaneous presence of both the forward wave and backward wave, as defined by,

$$v_s(z,t) = v_0 \cos\left(\omega\left(t - \frac{z}{v_p}\right) + \varphi\right) + v_0 \cos\left(\omega\left(t + \frac{z}{v_p}\right) + \varphi\right)$$
$$i_s(z,t) = \frac{v_0}{Z_0} \cos\left(\omega\left(t - \frac{z}{v_p}\right) + \varphi\right) - \frac{v_0}{Z_0} \cos\left(\omega\left(t + \frac{z}{v_p}\right) + \varphi\right)$$

The resulting voltage $v_s(z,t)$ and current $i_s(z,t)$ are standing waves. Look carefully at your movie for voltage standing wave and your movie for current standing wave. Do the maxima in voltage and maxima in current occur at the same positions *z* ? Do the maxima in voltage and maxima in current occur at the same times *t* ?

№ 2: Show your results to the teaching assistant.

4. Phasor representation of waves

Write a MATLAB function that gives the instantaneous voltage *v*(*t*) of a the complex phasor representation *V*,

```
[ v ] = ph2inst(V,omega,t)
```

using the following equation,

$$\mathrm{v}(t) = \mathrm{real}\{\, V \exp(j\omega t)\, \}$$

For efficiency, you may wish to write your function such that the input `V` is a vector of voltage phasor values *V*(*z*) along the position *z*, the input `omega` is the radian frequency ω, the input `t` is a vector of time *t* values, and the output `v` is a matrix of corresponding instantaneous voltage values *v*(*z*,*t*).

Confirm that your conversion function works by making a movie of the instantaneous voltage *v*(*z,t*) versus time *t* corresponding to the phasor $V = j \exp(+j\beta z)$ with $\beta = (2\pi/0.20)$ rad/m. Again, confine your numerical studies to a segment of transmission line $0 \le z \le 0.40$ m over a duration of time $0 \le t \le 5$ ns. Does the wave travel in the direction you expect? Is the instantaneous voltage *v*(*z,t* = 0) what you expect? Can you create a wave travelling in the opposite direction?

№ 3: Show your results to the teaching assistant.

APPENDIX A: MATLAB function definition

Create a new script entitled “funexample.m” with the following content:

```
% funexample
%
% author: James Tiberius Kirk
% last revision: 01-08-2020
%
% A very fun example of a MATLAB function for students of
% ECSE-354 found in the appendix of the general lab instructions, including
% a cosine and complex exponential description of periodic time signals
% input_a : amplitude in arbitrary units
% input_b : frequency in rad s^-1
% output_a : cosine evaluated from t = 0 s to t = 10 s
% output_b : complex exponential evaluated from t = 0 s to t = 10 s
% output_c : time from t = 0 s to t = 10 s
%

function [output_a output_b output_c] = funexample(input_a, input_b)

timevector = 0:0.01:10;
output_a = input_a*cos(input_b*timevector);
output_b = input_a*exp(1i*input_b*timevector);
output_c = timevector;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Notice the use of the delimiter `%` to allow the introduction of comments. In the MATLAB Command Window, you may now use this function. Executing `[A,B,timevec]=funexample(2,3);` at the command line will produce an output with `A`, `B` and `timevec` for `input_a = 2` and `input_b = 3`.

What does `help funexample` do?

APPENDIX B: MATLAB figures

We often wish to visualize numerical results in figures. This can also be automated in a script environment. Create a new script entitled “superfun.m” with the following content:

```
% superfun
%
% authors: James Tiberius Kirk and Jean-Luc Picard
% last revision: 02-08-2020
% description: A super fun example of a MATLAB script for students of
% ECSE-353 found in the appendix of the general lab instructions. The script
% uses the funexample function to generate data that is plotted beautifully.
%

clear all;   % clear the working environment

[ A,B,timevec ] = funexample(2,3);   % create a dataset using ‘fun example’

fig = figure(1);   % plot the cosine versus time
h = plot(timevec, A);
set(h,’Linwidth’,1);
set(gca,'LineWidth',1);
set(gca,'FontSize',18);
grid;
xlabel('Time [s]');
ylabel('Voltage [V]');
title('cosine');
set(gcf,'paperunits','points');
set(gcf,'PaperPosition', [0 0 800 600]);
set(gcf,'PaperSize',[800 600]);
axis([0 10 -2 2]);
print -dpdf 354_example_cosine

fig = figure(2);   % plot the real and imaginary exponential versus time
h = plot(timevec,real(B),timevec,imag(B));
set(h,’LineWidth’,1);
set(gca,'LineWidth',1);
set(gca,'FontSize',18);
grid;
legend('real','imaginary');
xlabel('Time [s]');
ylabel('Voltage [V]');
title('complex exponential');
set(gcf,'paperunits','points');
set(gcf,'PaperPosition', [0 0 800 600]);
set(gcf,'PaperSize',[800 600]);
axis([0 10 -2 2]);
print -dpdf 354_example_exponential

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Test the script above by executing `superfun` at the command line. Be sure to check the PDF output files. Note that you can adjust all the figure parameters with a script, at the command line, or with the graphical user interface.

You may find it useful to copy the figure plotting commands above for preparing your own figures.