Transmission Line Functions

Assignment 4

Due: 2021/03/05

1 Introduction

In this assignment, we examine a more realistic transmission line which takes into account the frequency dependence of its characteristic parameters. In the previous assignment, we assumed that wave propagation and characteristic impedance were constant for all frequencies. While this is a useful model to get a quick and general idea of how the transmission line will behave under certain conditions, it does not fully characterize the true behaviour that one would expect from a transmission line. In reality, both the characteristic impedance and the wave propagation characteristics are affected by the frequencies involved; in order to properly model these effects, one must have a transmission line's parameters as functions of frequency. This assignment is an exploration of these frequency dependent parameters.

2 Setup

For this assignment, a data file is provided, *Data_Assign04.txt*, which provides a value for each of the different transmission line parameters as a function of frequency. Two different current paths are provided for the transmission line frequency characteristics, Eigenmode 1, where the return conductor is the ground (zero sequence) and Eigenmode 2, where the return conductor is a normal conductor. The capacitance C and the shunt conductance G are assumed to be constant with frequency, and G is a non-zero, but small value.

From these values, the following plots are requested:

- The characteristic impedance $Z_c(\omega)$ in *magnitude* in units of (Ω/km) and *phase angle* in units of $(radians)$ (Figure 1)
- The propagation function $e^{-\gamma(\omega)l}$ in *magnitude* (0.0 - 1.0) and *phase angle*, where the angle of the function is monotonic and in units of $(radians)$ (Figures 2, 3)
- The attenuation $\alpha(\omega)$ in units of $(nepers/km)$ (Figure 4)
- The phase displacement $\beta(\omega)$ in units of $(radians/km)$ (Figure 5)
- The propagation speed $a(\omega)$ in units of (km/s) (Figure 6)

The data was converted to a CSV file and a Python script (found in Listing 5.1) was written to parse through the data, calculate the required quantities, and generate plots to be used in this report. All values were kept in their existing unit per km quantities, but were scaled to remove any prefix. Specifically, capacitance was converted from μF to F and inductance was converted from mH to H.

For the characteristic impedance $Z_c(\omega)$, the following equation was used:

$$Z_c(\omega) = \sqrt{\frac{R(\omega) + j\omega L(\omega)}{G(\omega) + j\omega C(\omega)}} \tag{1}$$

As mentioned previously, the values for C and G are considered constant for all frequencies. The resulting complex number's polar representation was plotted for both its magnitude and phase.

The following equation is used for the propagation function:

$$e^{-\gamma(\omega)l} = e^{-\alpha(\omega)l} \cdot e^{-j\beta(\omega)l} \tag{2}$$

where

$$\gamma(\omega) = \sqrt{[R(\omega) + j\omega L(\omega)]\,[G(\omega) + j\omega C(\omega)]} = \alpha(\omega) + j\beta(\omega) \tag{3}$$

The propagation function was a little trickier because using the resulting complex number's angle meant that the result would always be within the range of $(-\pi, \pi)$. The resulting phase plot would then be non-monotonic. Instead, one can simple represent the phase as $-\beta(\omega)l$, which is the phase displacement in units of $(radians/km)$ multiplied by the line length of 300 in units of (km) giving us the phase in $(radians)$. The magnitude of the propagation function is simply the magnitude of the resulting complex number.

The attenuation $\alpha(\omega)$ and phase displacement $\beta(\omega)$ functions are derived from $\gamma(\omega)$'s real and imaginary components, as shown above. The only equation remaining is the propagation speed $a(\omega)$, which is derived from the angular frequency $(radians/s)$ and the phase displacement $(radians/km)$ which gives us a propagation speed in (km/s).

$$a(\omega) = \frac{\omega}{\beta(\omega)} \tag{4}$$

With these equations, all of the plots requested by this assignment can be plotted. They can be found in the following Results section with a discussion of their characteristics in the Conclusion section.

3 Results

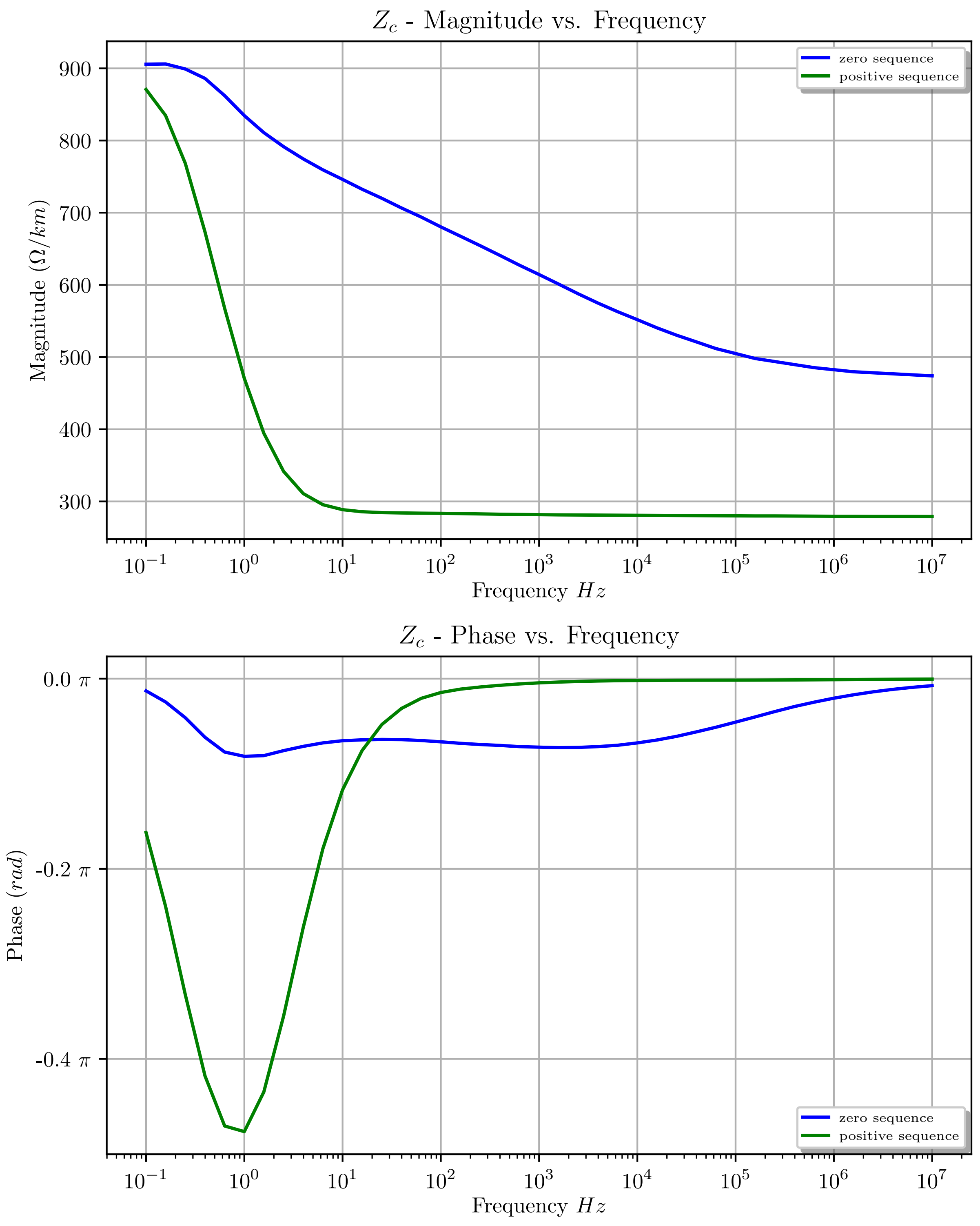
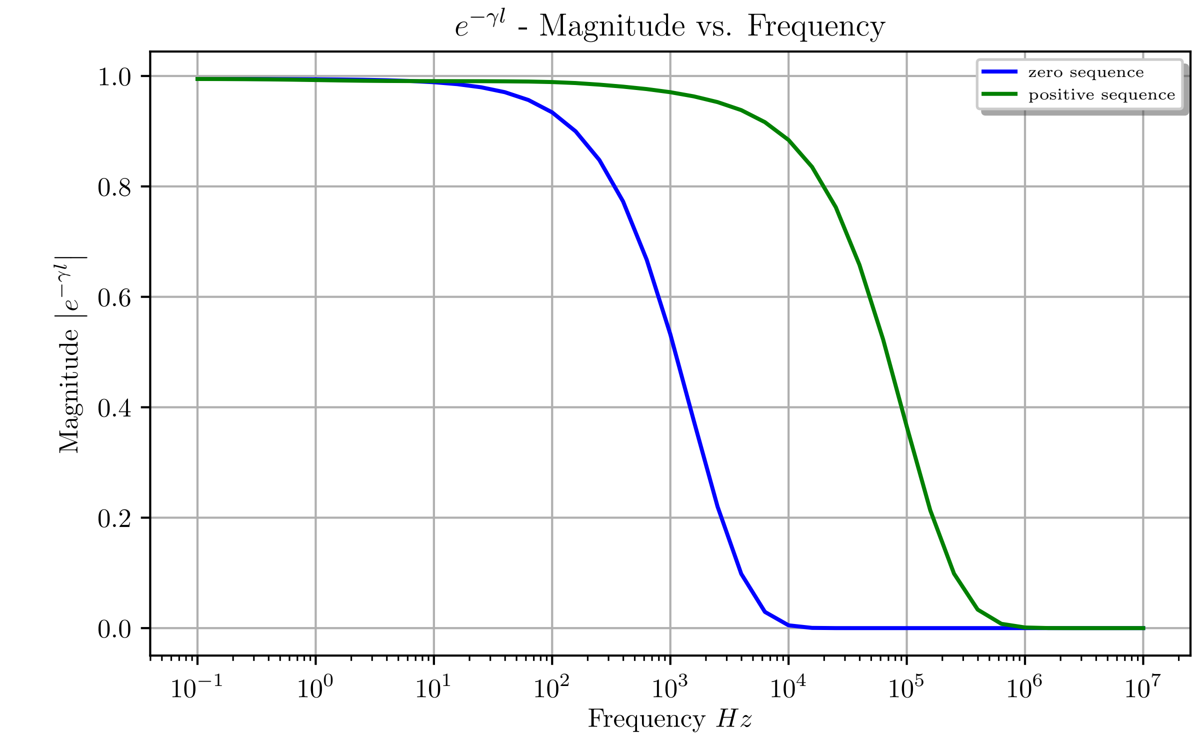


Figure 1: Characteristic Impedance (Z_c) Magnitude and Phase vs. Frequency Plots



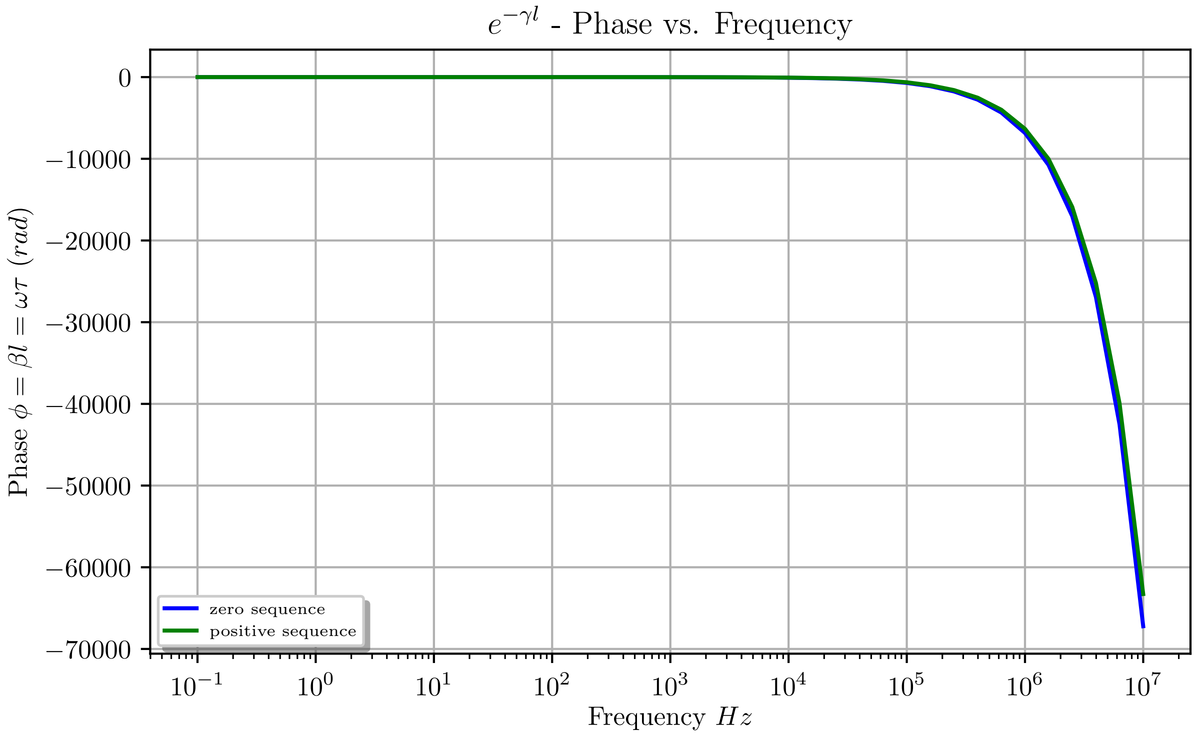


Figure 2: Propagation Function $e^{-\gamma(\omega)l}$ Magnitude and Phase vs. Frequency Plots

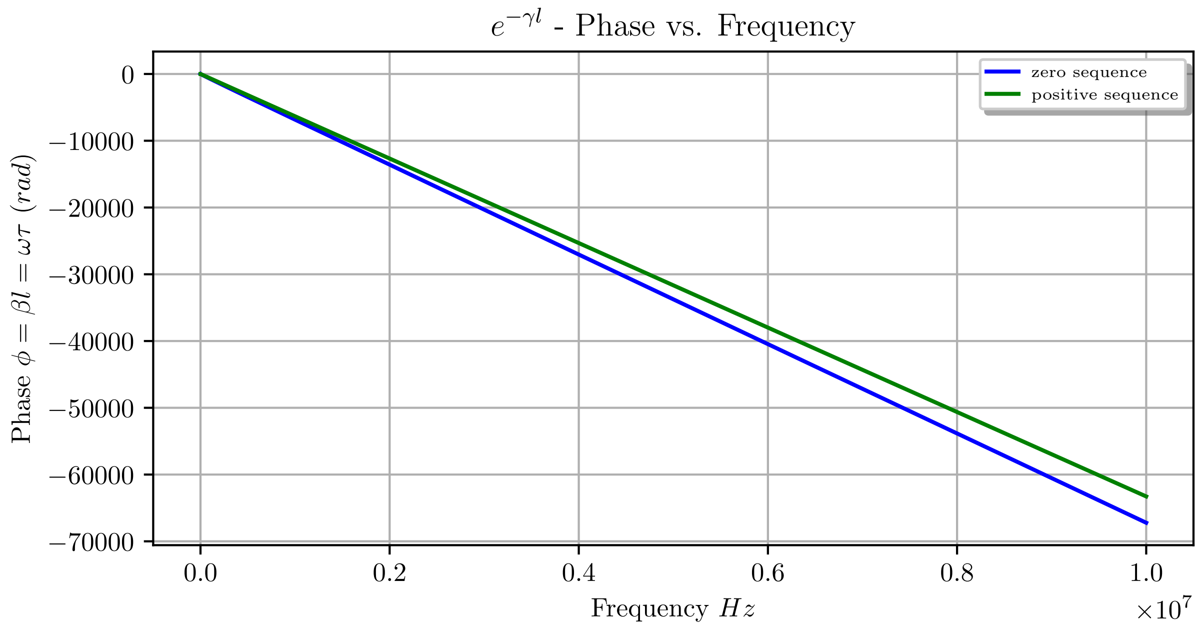


Figure 3: Propagation Function $e^{-\gamma(\omega)l}$ Magnitude and Phase vs. Linear Frequency Plot

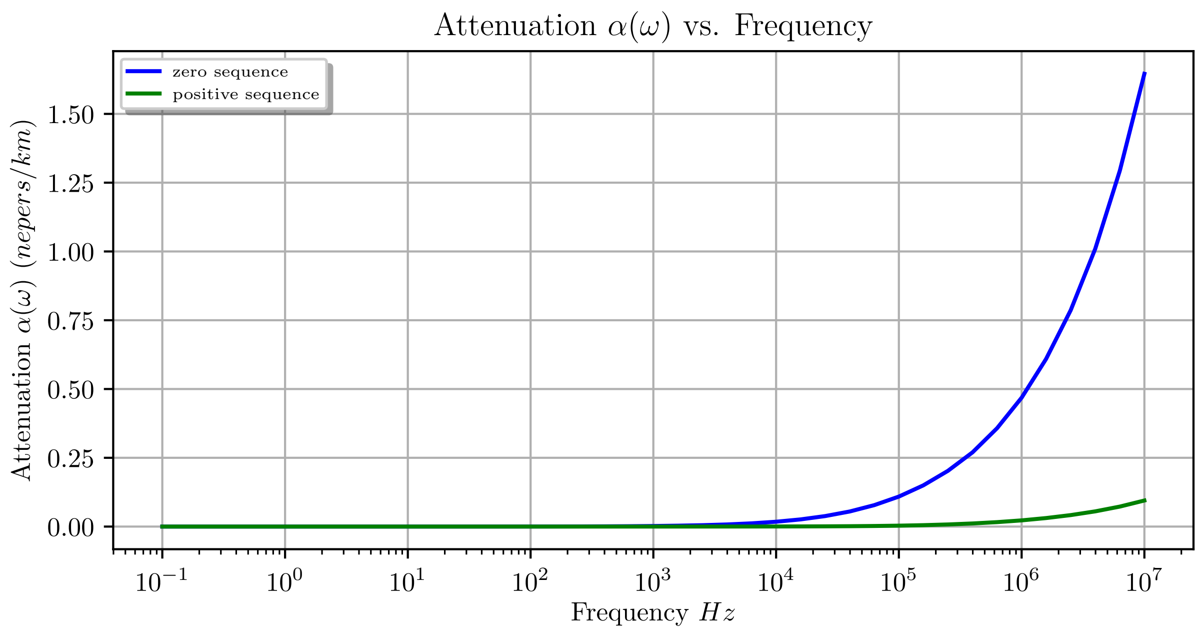


Figure 4: Attenuation $\alpha(\omega)$ vs. Frequency Plot

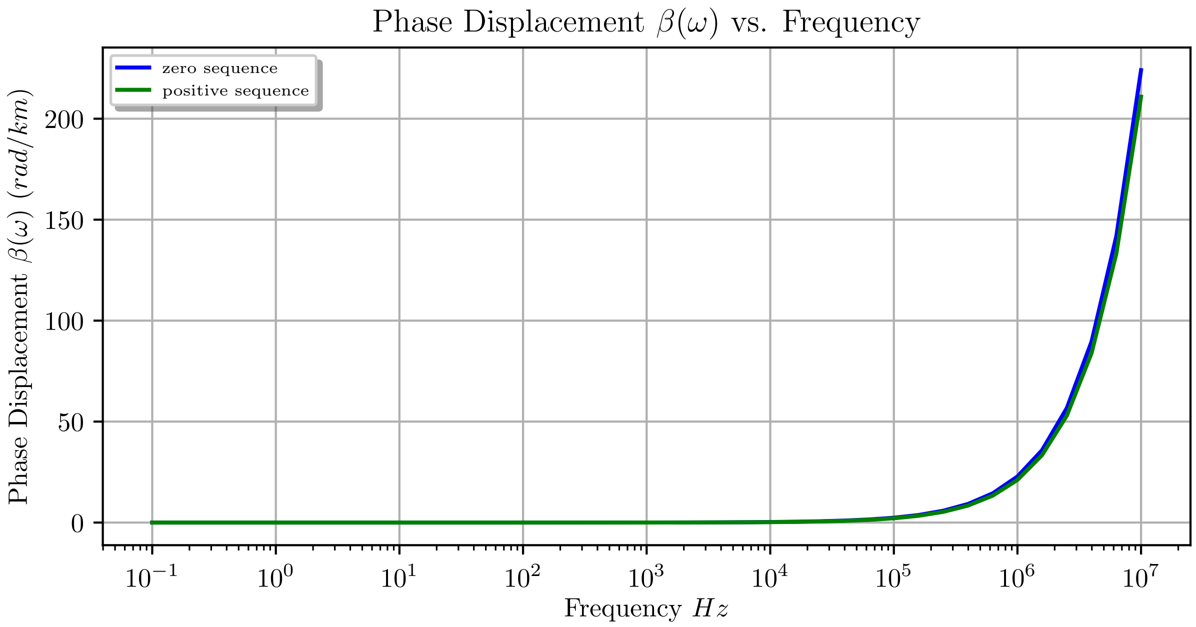


Figure 5: Phase Displacement $\beta(\omega)$ vs. Frequency Plot

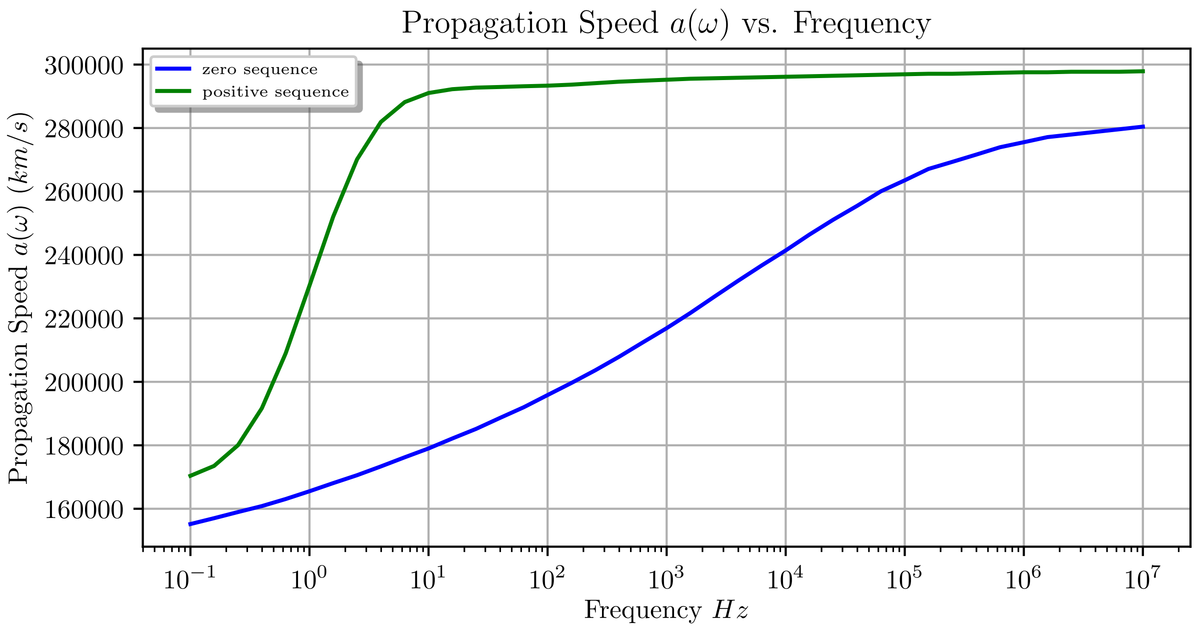


Figure 6: Propagation Speed $a(\omega)$ vs. Frequency Plot

4 Conclusion

There are distinct differences between the zero and positive sequences of the characteristic impedance $Z_c(\omega)$ and propagation function $e^{-\gamma(\omega)l}$. These differences likely arise from path the return current takes. In the zero sequence, the current returns via the ground, and in the positive sequence, the return conductor is another conductor.

For the characteristic impedance, we see that the magnitude of the zero sequence is always higher than that of the positive sequence. This seems intuitive because there would likely be greater losses when the current path is back through ground instead of another conductor. The fact that the two conductors are likely closer together will have an effect on their mutual capacitance and inductance which contributes to their wave propagating properties. At lower frequencies, the magnitude of the characteristic impedance is quite large, which gives a clue as to why power is transmitted in alternating current at 60 Hz, which minimizes the losses on the line.

If we ignore the low frequencies in the phase plot, we see that the zero sequence (return through ground) also has a negative reactance (capacitive) component to it for a large range of its frequencies, whereas the positive sequence (return through another conductor) is almost exclusively dominated by a resistive component. It seems that for 50/60 Hz (the two most common power transmission frequencies), the approximation of using a purely real impedance with no reactances is an appropriate one as the reactive components are small in the frequency dependent model.

When observing the propagation function's magnitude, we see two similar shapes at two different frequencies. The magnitude here represents the attenuation of the wave propagating along the line, and we see that until we hit high frequencies, the attenuation is minimal. Specifically, at 50/60Hz we see very little attenuation, which once more shows how appropriate those frequencies are for power delivery. Eventually, both magnitudes drop off completely to zero as they approach higher frequencies.

The zero and positive sequence propagation function phases represent an almost linear increase in the negative range with different slopes. The phase was plotted with frequency on a linear scale to make this evident, in Figure 3. This makes sense, as the phase of the propagation function can also be represented by $\omega\tau$, where τ is the time it takes for the wave to propagate across the line. Since the propagation speed remains in the same order of magnitude across all frequencies (and very close numerically at higher frequencies), the difference in phase is negligible, manifesting in this linear change. This shows once more that a constant value for τ is a decent approximation for a transmission line model.

Observing the propagation speed in Figure 6, we notice that the wave does not travel at the same velocity across all frequencies. This is because the transmission line has frequency-dependent components that define the velocity of the wave traveling across the transmission line. Concretely, our data contains a value for the inductance per unit length that varies with frequency. Since wave velocity is approximately equal to $\frac{1}{\sqrt{L'C'}}$, we see that our velocity will change with frequency as we are computing the velocity with different values of L' at every frequency data point.

This assignment is a good demonstration of how an ideal transmission line model is able to approximate the frequency dependent model. However, there are likely many cases where a frequency dependent model would be necessary to accurately model edge cases, especially in the lower frequencies. It would be interesting to model the transmission line using these parameters and observe an impulse travel across the transmission line and observe the different behaviours when the line is shorted, opened, and matched.

5 Code Listings and Data

5.1 Python Code Listing

The following is the code written in Python to perform the calculations derived for this homework assignment as well as generate the plots used in this report. The transmission line parameters were converted to a CSV file which is read in by this Python script.

```python
import argparse
import csv
import matplotlib
import matplotlib.ticker as tck
import matplotlib.pyplot as plt
import numpy as np

 # Matplotlib export settings
matplotlib.use('pgf')
import matplotlib.pyplot as plt
matplotlib.rcParams.update({
    'pgf.texsystem': 'pdflatex',
    'font.size': 10,
    'font.family': 'serif',  # use serif/main font for text elements
    'text.usetex': True,     # use inline math for ticks
    'pgf.rcfonts': False     # don't setup fonts from rc parameters
})

# Main function
def main(args):

    C_zero = 7.5240e-03 * 1e-6 # Farads/km
    C_pos = 1.2027e-02 * 1e-6 # Farads/km
    G_zero = 2.0000e-08 # Mhos/km
    G_pos = 2.0000e-08 # Mhos/km
    length = 300 # km

    FREQ_INDEX = 0
    R_ZERO_INDEX = 1
    L_ZERO_INDEX = 2
    R_POS_INDEX = 3
    L_POS_INDEX = 4

    MAGNITUDE_INDEX = 0
    PHASE_INDEX = 1

    # prepopulate data with a list of five empty lists
    data = [[] for i in range(5)]

    # Read in PSCAD .CSV data
    print('***␣Opening␣assignment␣4␣CSV␣data␣file...')
    with open('data_assign04.csv') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        line_count = 0
        # Read in row data
        for row in csv_reader:
            if line_count == 0:
                print('Column␣names␣are:␣' + ',␣'.join(row))
                line_count += 1
            else:
                data[FREQ_INDEX].append(float(row[0]))
                data[R_ZERO_INDEX].append(float(row[1])) # Ohms/km
                data[L_ZERO_INDEX].append(float(row[2]) * 1e-3) # Henries/km
                data[R_POS_INDEX].append(float(row[3])) # Ohms/km
                data[L_POS_INDEX].append(float(row[4]) * 1e-3) # Henries/km
                line_count += 1
        # Figure out when break switched
        print('Processed␣' + str(line_count) + '␣lines.')

    num_data_points = len(data[FREQ_INDEX])

    # Prepare values for Z(w) magnitude and phase
    impedance_zero = [[],[]]
    impedance_pos = [[],[]]
    for index in range(num_data_points):
        omega = 2*np.pi*data[FREQ_INDEX][index]
        impedance_zero_val = np.sqrt((data[R_ZERO_INDEX][index] + (1j*omega*data[
            L_ZERO_INDEX][index]))/(G_zero + (1j*omega*C_zero)))
        impedance_pos_val =  np.sqrt((data[R_POS_INDEX][index] +  (1j*omega*data[L_POS_INDEX
            ][index])) /(G_pos +  (1j*omega*C_pos)))
        # print("F: " + str(data[FREQ_INDEX][index]))
        # print("Omega: " + str(omega))
        # print("R_0: " + str(data[R_ZERO_INDEX][index]))
        # print("L_0: " + str(data[L_ZERO_INDEX][index]))
        # print("C_0: " + str(C_zero))
        # print("G_0: " + str(G_zero))
        # print("R_+: " + str(data[R_POS_INDEX][index]))
        # print("L_+: " + str(data[L_POS_INDEX][index]))
        # print("C_+: " + str(C_pos))
        # print("G_+: " + str(G_pos))
        # print("Zc_0: " + str(impedance_zero_val))
        # print("Zc_0 mag: " + str(np.absolute(impedance_zero_val)))
        # print("Zc_+: " + str(impedance_pos_val))
        # print("Zc_+ mag: " + str(np.absolute(impedance_pos_val)))
        impedance_zero[MAGNITUDE_INDEX].append(np.absolute(impedance_zero_val))
        impedance_zero[PHASE_INDEX].append(np.angle(impedance_zero_val))
        impedance_pos[MAGNITUDE_INDEX].append(np.absolute(impedance_pos_val))
        impedance_pos[PHASE_INDEX].append(np.angle(impedance_pos_val))
        print("\r\n")

    # Prepare values for propagation function magnitude and phase as well
    # Prepare values for attenuation alpha(w) (nepers/km)
    # Prepare values for phase displacement beta(w) (radians/km)
    # Prepare values for propagation speed a(w) (km/s)
    propagation_zero = [[],[]]
    propagation_pos = [[],[]]
    attenuation_zero = []
    attenuation_pos = []
    phase_zero = []
    phase_pos = []
    propagation_speed_zero = []
    propagation_speed_pos = []
    for index in range(num_data_points):
        omega = 2*np.pi*data[FREQ_INDEX][index]
        gamma_zero = np.sqrt((data[R_ZERO_INDEX][index] + 1j*omega*data[L_ZERO_INDEX][index
            ])*(G_zero + 1j*omega*C_zero))
        gamma_pos = np.sqrt((data[R_POS_INDEX][index] + 1j*omega*data[L_POS_INDEX][index])*(
            G_pos + 1j*omega*C_pos))
        # propagation function magnitude and phase
        propagation_zero[MAGNITUDE_INDEX].append(np.absolute(np.exp(-1*gamma_zero*length)))
        propagation_zero[PHASE_INDEX].append(-np.imag(gamma_zero)*length)
        propagation_pos[MAGNITUDE_INDEX].append(np.absolute(np.exp(-1*gamma_pos*length)))
        propagation_pos[PHASE_INDEX].append(-np.imag(gamma_pos)*length)
        # attenuation (real component of gamma) (nepers/km)
        attenuation_zero.append(np.real(gamma_zero))
        attenuation_pos.append(np.real(gamma_pos))
        # phase displacement (imaginary component of gamma) (radians/km)
        phase_zero.append(np.imag(gamma_zero))
        phase_pos.append(np.imag(gamma_pos))
        # propagation speed (omega/phase_displacement) (km/s)
        propagation_speed_zero.append(omega/phase_zero[-1])
        propagation_speed_pos.append(omega/phase_pos[-1])
        # propagation_speed_zero.append(1/np.sqrt(data[L_ZERO_INDEX][index]*C_zero))
        # propagation_speed_pos.append(1/np.sqrt(data[L_POS_INDEX][index]*C_pos))
```

```
# Plots for publication
legend_font_size = 6
# Plot Z(w) magnitude and phase
fig, ax = plt.subplots(2)
ax[0].plot(data[FREQ_INDEX], impedance_zero[MAGNITUDE_INDEX], color='b', label='zero␣
    sequence')
ax[0].plot(data[FREQ_INDEX], impedance_pos[MAGNITUDE_INDEX], color='g', label='positive␣
    sequence')
ax[0].set(xlabel='Frequency␣$Hz$', ylabel='Magnitude␣($\Omega/km$)', title='$Z_c$␣-␣
    Magnitude␣vs.␣Frequency')
ax[0].grid()
ax[0].set_xscale('log')
ax[0].legend(loc='best', prop={'size':legend_font_size}, fancybox=True, shadow=True)
ax[1].plot(data[FREQ_INDEX], impedance_zero[PHASE_INDEX], color='b', label='zero␣
    sequence')
ax[1].plot(data[FREQ_INDEX], impedance_pos[PHASE_INDEX], color='g', label='positive␣
    sequence')
ax[1].yaxis.set_major_formatter(tck.FormatStrFormatter('%1.1f␣$\pi$'))
ax[1].yaxis.set_major_locator(tck.MultipleLocator(base=1/5))
ax[1].set(xlabel='Frequency␣$Hz$', ylabel='Phase␣($rad$)', title='$Z_c$␣-␣Phase␣vs.␣
    Frequency')
ax[1].grid()
ax[1].set_xscale('log')
ax[1].legend(loc='best', prop={'size':legend_font_size}, fancybox=True, shadow=True)
fig.set_size_inches(6.5,8)
fig.tight_layout()
fig.savefig('zc_magnitude_phase_plots.pgf')
fig.savefig('zc_magnitude_phase_plots.png')

# Plot propagation function magnitude and phase
fig, ax = plt.subplots(2)
ax[0].plot(data[FREQ_INDEX], propagation_zero[MAGNITUDE_INDEX], color='b', label='zero␣
    sequence')
ax[0].plot(data[FREQ_INDEX], propagation_pos[MAGNITUDE_INDEX], color='g', label='
    positive␣sequence')
ax[0].set(xlabel='Frequency␣$Hz$', ylabel=r'Magnitude␣$\left|e^{-\gamma{}l}\right|$',
    title='$e^{-\gamma{}l}$␣-␣Magnitude␣vs.␣Frequency')
ax[0].grid()
ax[0].set_xscale('log')
ax[0].legend(loc='best', prop={'size':legend_font_size}, fancybox=True, shadow=True)
ax[1].plot(data[FREQ_INDEX], propagation_zero[PHASE_INDEX], color='b', label='zero␣
    sequence')
ax[1].plot(data[FREQ_INDEX], propagation_pos[PHASE_INDEX], color='g', label='positive␣
    sequence')
ax[1].set(xlabel='Frequency␣$Hz$', ylabel=r'Phase␣$\phi{}=\beta{}l=\omega{}\tau$␣($rad$)
    ', title='$e^{-\gamma{}l}$␣-␣Phase␣vs.␣Frequency')
ax[1].grid()
ax[1].set_xscale('log')
ax[1].legend(loc='best', prop={'size':legend_font_size}, fancybox=True, shadow=True)
fig.set_size_inches(6.5,8)
fig.tight_layout()
fig.savefig('prop_magnitude_phase_plots.pgf')
fig.savefig('prop_magnitude_phase_plots.png')

# Plot propagation function magnitude and phase (no long for frequency)
fig, ax = plt.subplots(1)
ax.plot(data[FREQ_INDEX], propagation_zero[PHASE_INDEX], color='b', label='zero␣sequence
    ')
ax.plot(data[FREQ_INDEX], propagation_pos[PHASE_INDEX], color='g', label='positive␣
    sequence')
ax.set(xlabel='Frequency␣$Hz$', ylabel=r'Phase␣$\phi{}=\beta{}l=\omega{}\tau$␣($rad$)',
    title='$e^{-\gamma{}l}$␣-␣Phase␣vs.␣Frequency')
ax.grid()
ax.legend(loc='best', prop={'size':legend_font_size}, fancybox=True, shadow=True)
fig.set_size_inches(6.5,3.5)
fig.tight_layout()
fig.savefig('prop_phase_plot_nolog.pgf')
fig.savefig('prop_phase_plot_nolog.png')
```

```
    # Plot attenuation (real component of gamma) (nepers/km)
    fig, ax = plt.subplots()
    ax.plot(data[FREQ_INDEX], attenuation_zero, color='b', label='zero␣sequence')
    ax.plot(data[FREQ_INDEX], attenuation_pos, color='g', label='positive␣sequence')
    ax.set(xlabel='Frequency␣$Hz$', ylabel=r'Attenuation␣$\alpha{}(\omega)$␣$(nepers/km)$',
        title=r'Attenuation␣$\alpha{}(\omega)$␣vs.␣Frequency')
    ax.grid()
    ax.set_xscale('log')
    ax.legend(loc='best', prop={'size':legend_font_size}, fancybox=True, shadow=True)
    fig.set_size_inches(6.5,3.5)
    fig.tight_layout()
    fig.savefig('attenuation_plots.pgf')
    fig.savefig('attenuation_plots.png')

    # Plot phase displacement beta(w) (radians/km)
    fig, ax = plt.subplots()
    ax.plot(data[FREQ_INDEX], phase_zero, color='b', label='zero␣sequence')
    ax.plot(data[FREQ_INDEX], phase_pos, color='g', label='positive␣sequence')
    ax.set(xlabel='Frequency␣$Hz$', ylabel=r'Phase␣Displacement␣$\beta{}(\omega)$␣$(rad/km)$
        ', title=r'Phase␣Displacement␣$\beta{}(\omega)$␣vs.␣Frequency')
    ax.grid()
    ax.set_xscale('log')
    ax.legend(loc='best', prop={'size':legend_font_size}, fancybox=True, shadow=True)
    fig.set_size_inches(6.5,3.5)
    fig.tight_layout()
    fig.savefig('phase_displacement_plots.pgf')
    fig.savefig('phase_displacement_plots.png')

    # Plot propagation speed a(w) (km/s)
    fig, ax = plt.subplots()
    ax.plot(data[FREQ_INDEX], propagation_speed_zero, color='b', label='zero␣sequence')
    ax.plot(data[FREQ_INDEX], propagation_speed_pos, color='g', label='positive␣sequence')
    ax.set(xlabel='Frequency␣$Hz$', ylabel=r'Propagation␣Speed␣$a(\omega)$␣$(km/s)$', title=
        r'Propagation␣Speed␣$a(\omega)$␣vs.␣Frequency')
    ax.grid()
    ax.set_xscale('log')
    ax.legend(loc='best', prop={'size':legend_font_size}, fancybox=True, shadow=True)
    fig.set_size_inches(6.5,3.5)
    fig.tight_layout()
    fig.savefig('propagation_speed_plots.pgf')
    fig.savefig('propagation_speed_plots.png')

if __name__ == '__main__':
    # the following sets up the argument parser for the program
    parser = argparse.ArgumentParser(description='Assignment␣4␣solution␣generator')

    args = parser.parse_args()

    main(args)
```