W7

Name: Mohammed AL Shuaili

Date: 26/2/2025

Consider the following frequency response for a transmission line.

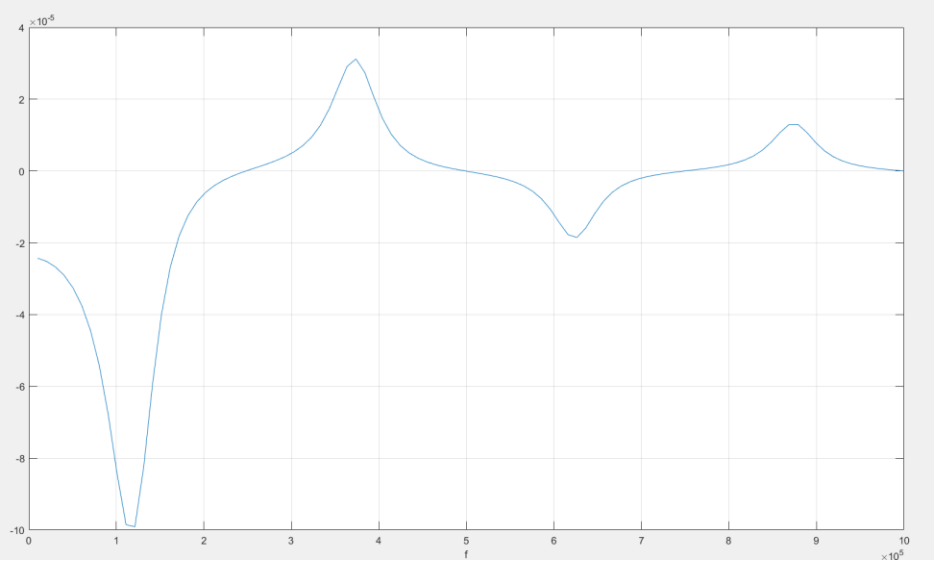

Figure 1 : shows the frequency response of the exact solution.

This is modelled over 100 frequency points using the following parameters:

$$R = 0.1, l = 400m, L = 2.5\ x10^{-7}, C = 1x10^{-10}, v_{in} = 30\ , f_{max} = 1x10^{6}$$

Generating rational approximation using the first 15 points will result in with $w(1) = 0$:

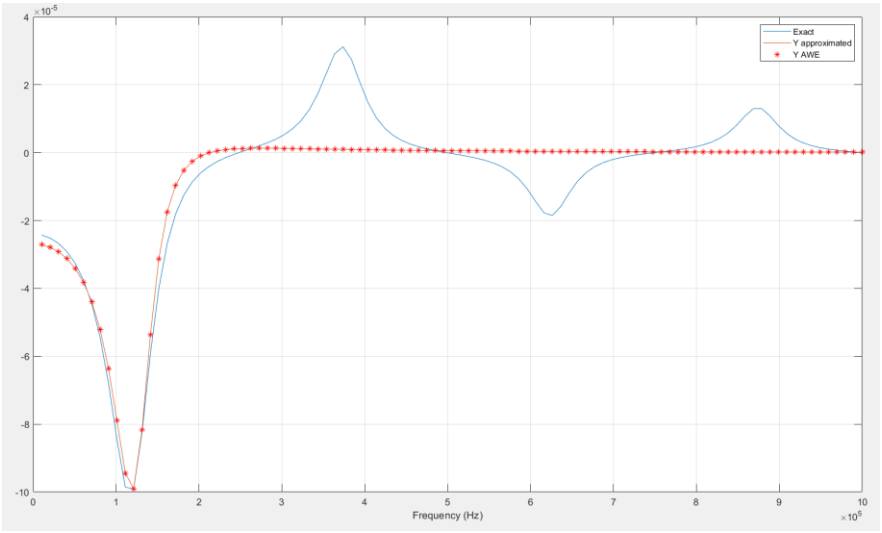

Figure 2 compares the exact responses to AWE and Y Rational approximation.

The poles are:

$$p_1 = -1.7680 + 7.8673i, p_2 = -1.7680 - 7.8673i \; x10^5$$

Observations:

For any approximation, when using AWE at $w = 0$, the poles are always the same with different sign.

For Example, consider a second model where the exact values are used from (15:25) as in the following code:

```
[H2,num,deno] = generate_yp2(real(vo(15:25)),imag(vo(15:25)),w(15:25));
[A,B,C,D] = create_state_space(num,deno);
[poles2, moments2] = AWE_poles(A,B,C,D,w(1));
```

where $w(1) = 0$, the poles are,

$$p_{12} = -0.1275 \pm 1.1137i \; x10^6$$

This is without evaluating the second approximation as the difference between the exact and the first model at (15:25).

With H_diff:

$$p_{12} = -0.3378 \pm 1.3363i \; x10^6$$

AWE adjusted to be evaluated about s_o

```
function [h_impulse, h_s, y_step, t] = AWE2(A, B, C, D, w, input, time)
    t = linspace(0, time, 250);
    q = length(B);
    num_moments = 2 * q;
    s0 = 1i * w;
    moments = zeros(1, num_moments);
    [r, c] = size(C);
    if r ~= 1
        C = C';
    end
    for k = 1:num_moments
        moments(k) = (-1)^(k-1) * C * (s0 * eye(size(A)) - A)^-(k) * B;
    end
    moments(1) = moments(1) + D;  % Include D in the zeroth moment

    approx_order = q;

    % Construct moment matrix and vector for denominator coefficients
    moment_matrix = zeros(approx_order);
    Vector_c = -moments(approx_order+1 : 2*approx_order)';
    for i = 1:approx_order
        moment_matrix(i, :) = moments(i : i + approx_order - 1);
    end

    % Solve for denominator coefficients
    b_matrix = moment_matrix \ Vector_c;
    poles_unshifted = roots([b_matrix; 1]);  % Unshifted poles (s' = s - s0)

    % Compute residues using unshifted poles
    V = zeros(approx_order);
    for i = 1:approx_order
```

```
        for j = 1:approx_order
            V(i, j) = 1 / (poles_unshifted(j))^(i-1);
        end
    end
    A_diag = diag(1 ./ poles_unshifted);
    r_moments = moments(1:approx_order);
    residues = -A_diag \ (V \ r_moments(:));d

    % Shift poles to s-plane
    poles = poles_unshifted + s0;

    % Impulse response using shifted poles
    h_impulse = zeros(size(t));
    for i = 1:approx_order
        h_impulse = h_impulse + residues(i) * exp(poles(i) * t);
    end

    % Transfer function in s-domain
    h_s = @(s) sum(residues ./ (s - poles), 1);

    % Step response using recursive convolution
    y_step = zeros(size(t));
    y = zeros(length(poles), 1);
    for n = 2:length(t)
        dt = t(n) - t(n-1);
        exp_term = exp(poles * dt);
        for i = 1:length(poles)
            y(i) = residues(i) * (1 - exp_term(i))/(-poles(i)) * input + exp_term(i) * y(i);
        end
        y_step(n) = sum(y);
    end
end
```

in this code the poles are shifted to the origin after calculating moments and residues which gives much accurate results depending on the number of models and the transmission line parameters.

- Complex frequency hoping

Analyse the poles at different frequency and find the dominant poles. Consider 4 hopes.

w	Poles
AWE_$W(1) = 0$, Y_W(1:15)	$-1.7680 \pm 7.8673i \; x10^5$
AWE_$W(16) = w_0$, Y_W(16:25)	-0.4033 + 1.2386i 0.0798 + 1.0928i $x10^6$
AWE_$W(26) = w_0$, Y_W(26:35)	-0.0939 + 2.2961i -0.0458 + 1.5426i $x10^6$
AWE_$W(36) = w_0$, Y_W(36:45)	-0.0222 + 2.5013i -0.0035 + 2.2141i $x10^6$

Chose the dominant poles using overlapping.

Observations:

1. There is only one pole that's not stable in the second model.
2. There is no obvious overlap between these poles.
3. This suggests that these poles might be not enough for Pole Convergence.

- Adjust AWE to include larger number of moments and using L and M of the Pade approximation table in order to obtain 20-30 moments to capture more poles and residues near s_0

The following code implement AWE with different value of L and M to achieve better accuracy, choosing M to be 6 and $L = M$ as in the paper.

```
clear
clc
% test AWE
A = [-2, 1, 0, 0; 1, -2, 1, 0; 0, 1, -2, 1;0, 0, 1, -1];
B = [1; 0; 0; 0];
C = [1; 0; 0; 0];
D = 0;
sys = ss(A,B,C',D);
t = linspace(0, 2, 1000);
[y1,t1]=step(sys,t);
t_end = 2;
wo = 0;
M = 7;
[h_s, y_step, t]= AWE_CFH(A, B, C, D, M,wo, 1, 2);
plot(t,y_step,t1,y1)
xlabel('time s')
ylabel('H(t)')
title(['Impulse response at w= ', num2str(wo)]);
grid on
RMSE = sqrt(sum(abs(y_step-y1').^2)/length(y1));
```

where AWE_CFH is

```
function [h_s, y_step, t]= AWE_CFH(A, B, C, D, M, w0, input, t)
    L=M;
    t = linspace(0, t, 1000);
    num_moments = L + M + 1; % Moments from m₀ to m_{L+M}
    s0 = 1i * w0; % Complex frequency point
    moments = zeros(1, num_moments);

    % Ensure C is a row vector
    [r, ~] = size(C);
    if r ~= 1
        C = C';
    end

    % Compute moments m₀ to m_{L+M}
    for k = 1:num_moments
        moments(k) = (-1)^(k-1) * C * (s0 * eye(size(A)) - A) ^-(k) *(B);
    end
    moments(1) = moments(1) + D; % Add the term D to m₀
```

```
    approx_order = M;

    % Construct moment matrix and vector for denominator coefficients
    moment_matrix = zeros(approx_order);
    Vector_c = -moments(approx_order+1 : 2*approx_order)';
    for i = 1:approx_order
        moment_matrix(i, :) = moments(i : i + approx_order - 1);
    end

    % Solve for denominator coefficients
    b_matrix = moment_matrix \ Vector_c;
    poles_unshifted = roots([b_matrix; 1]);  % Unshifted poles (s' = s - s0)

    % Compute residues using unshifted poles
    V = zeros(approx_order);
    for i = 1:approx_order
        for j = 1:approx_order
            V(i, j) = 1 / (poles_unshifted(j))^(i-1);
        end
    end
    A_diag = diag(1 ./ poles_unshifted);
    r_moments = moments(1:approx_order);
    residues = -A_diag \ (V \ r_moments(:));

    % Shift poles to s-plane
    poles = poles_unshifted + s0;

    % Impulse response using shifted poles
    h_impulse = zeros(size(t));
    for i = 1:approx_order
        h_impulse = h_impulse + residues(i) * exp(poles(i) * t);
    end

    % Transfer function in s-domain
    h_s = @(s) sum(residues ./ (s - poles), 1);

    % Step response using recursive convolution
    y_step = zeros(size(t));
    y = zeros(length(poles), 1);
    for n = 2:length(t)
        dt = t(n) - t(n-1);
        exp_term = exp(poles * dt);
        for i = 1:length(poles)
            y(i) = residues(i) * (1 - exp_term(i))/(-poles(i)) * input + exp_term(i) * y(i);
        end
        y_step(n) = sum(y);
    end

end
```

even though the code faced computational rounding (Results may be inaccurate. RCOND = 7.904153e-21'), the results of the new AWE with an adjustable number of moments (by choosing M and L) gives better results with RMSE of $5.5023x10^{-12}$ compared to $4.8606x10^{-11}$.

Main issues are causing inaccurate modelling of the transmission line.

1. Rounding issues that when computing the moments, poles and residues for higher value of M.
2. The generated second order rational approximation of a frequency of interest is not accurate, so trying to generate a model out of inaccurate model might be the main issue.

However, if the order is increased to achieve better approximation, AWE faces more rounding issues.

Example of the second point.

Let's consider the first model generated in Figure 2, the error associated with this model is 1.7435 RMSE. This is computed by comparing the exact impulse to the rational approximation using the same frequencies used to generate the rational expression. Hence, AWE will not be accurate compared to the exact solution as it's using an inaccurate model. If a third order of the rational approximation is used the error will be reduced to 0.0060 RMSE but will face computational rounding issues in AWE. In addition, Using the code for AWE2 above, the unit step response with enough number of models (12 – 18) gives an error of (3.2– 1.5) RMSE when evaluating around w' as the beginning of each model.

- Consider AWE_CFH and compare the poles to find the dominant poles.

w	Poles
AWE_$W(1) = 0$, Y_W(1:15)	0.5011 + 1.4753i 0.5011 - 1.4753i -0.8983 + 0.0000i 0.7086 + 0.0000i -0.1697 + 0.7764i -0.1697 - 0.7764i $x10^6$
AWE_$W(16) = w_0$, Y_W(16:25)	-0.2745 - 1.1706i 0.2172 + 0.0800i 0.5094 + 0.6723i 0.0733 + 1.3918i -0.2745 + 1.1706i -0.0000 + 0.8568i $x10^6$
AWE_$W(26) = w_0$, Y_W(26:35)	-0.1682 - 1.9510i -0.1682 + 1.9510i -0.4358 + 1.2898i 0.1559 + 0.9660i 0.3246 + 1.5468i -0.0000 + 1.4280i $x10^6$
AWE_$W(36) = w_0$, Y_W(36:45)	-0.0950 - 2.3562i 0.4129 + 1.5867i -0.2827 + 1.9575i -0.0950 + 2.3562i 0.2866 + 2.1243i -0.0000 + 1.9992i $x10^6$

How to pick the dominant poles, a summery of CFH (What I understood so far):

1. **Pole Confirmation via Overlap**:
    - Poles detected from different hops are compared. If the same pole (within a tolerance *error*) is found in two or more hops, it is marked as accurate.
2. **Radius of Accuracy**:
    - For each hop, the distance to the farthest confirmed pole defines a radius of accuracy ($R_{accuracy}$).
    - All poles within $R_{accuracy}$ of the hop are considered valid (even if not explicitly confirmed by another hop).
3. **Dominance Criteria**:
    - Proximity to Imaginary Axis: Poles closer to the $j\omega$-axis dominate transient/frequency responses (e.g., ringing, crosstalk).
    - Large Residues: Poles with larger residues k_j in (3) are prioritized, as they have stronger influence on the output.
4. **Rejection of Spurious/Non-Dominant Poles**:
    - Poles outside the radius of accuracy or not confirmed by multiple hops are discarded.
    - Right-half plane (unstable) poles are rejected for stable systems.

Binary Search Strategy (Section III.C):

- Midpoints are adaptively chosen to minimize hops while ensuring coverage of critical regions (e.g., sharp resonances).
- Example: If no common poles are found between s=0 and $s=j\omega_{max}$, a midpoint hop at s=$\frac{j\omega_{max}}{2}$ is added.

Once this is confirmed along with other issues, the code can be generated and automated and the above example can be used.

Plotting THz transmission line using the following parameter:

$$C = 1e-10F, \qquad L = 250e-9H, \qquad R = 1200Ohms, \qquad l = 150e-6m$$

1. I will be using the exact solution to get approximation and from now on this model will be used for validation.

Step response with 10V input:

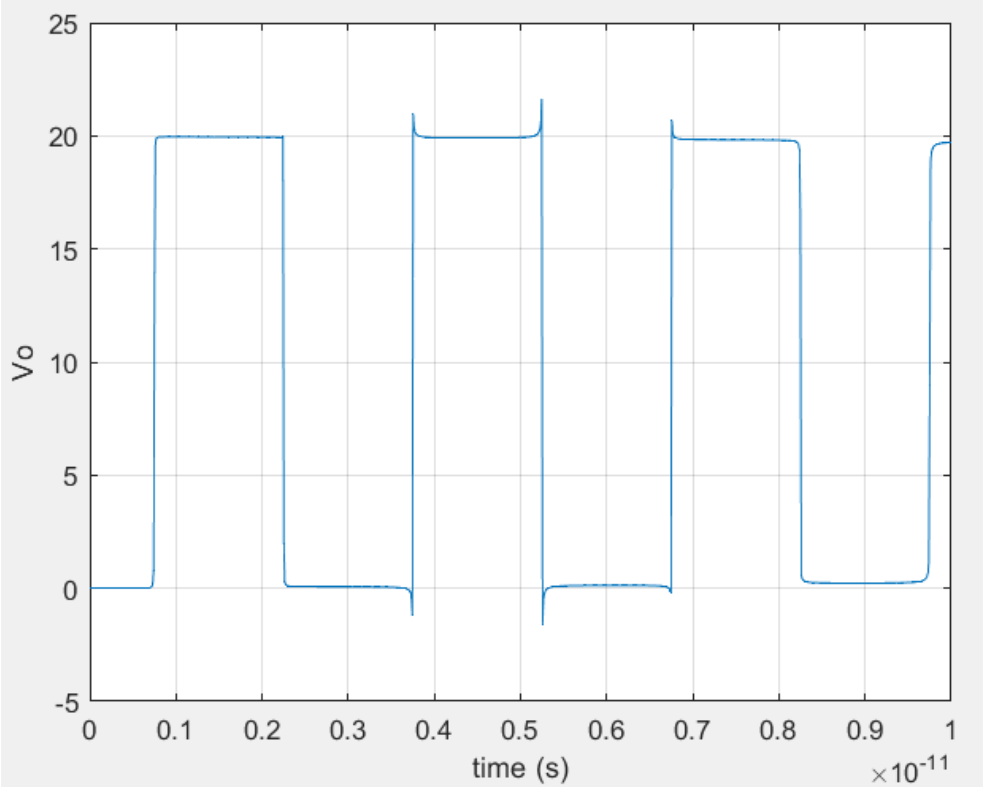

Figure 3 : shows the step response of the THz transmission line.

Sinewave with a frequency of 100GHz.

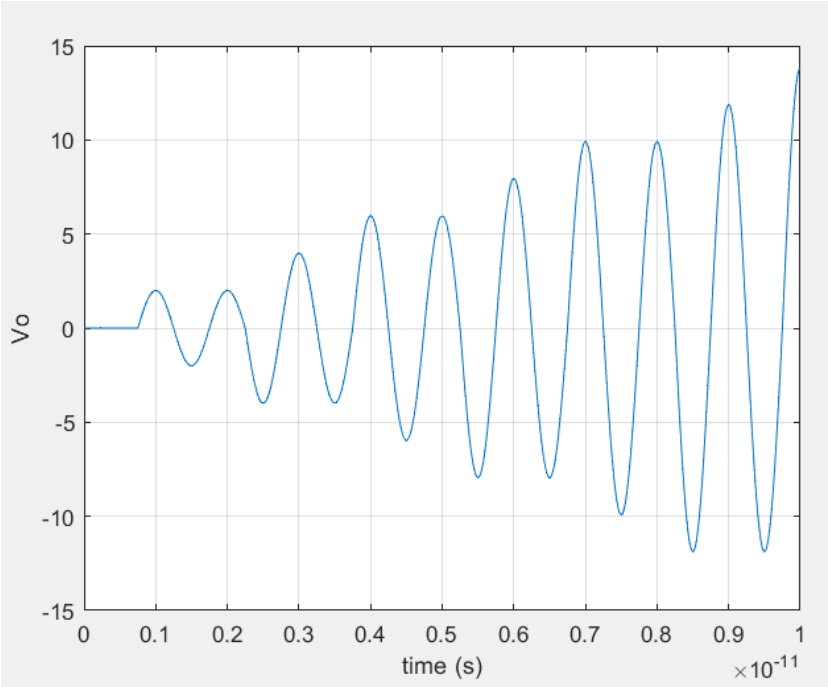

Figure 4 shows the response due to sine wave with 100 GHz frequency.

This plot shows a growing oscillation which is unexpected, not sure why. The following code is used:

```
clear
clc
R = 1200;          % Resistance per unit length (Ω/m)
L = 250e-9;        % Inductance per unit length (H/m)
C = 1e-10;         % Capacitance per unit length (F/m)
Rs = 10;
G = 0;
l = 150e-6;        % Length of the transmission line
f_max = 100e10;    % Maximum frequency (100 GHz)
%f = linspace(0, f_max, 100);
w = 2*pi*f_max;
s = 1i*w;
vs_sine = @(s) w./(s.^2 + w^2);% Laplace transform of sin(wt)
% Transfer function (exact solution)
vo = @(s) 1 ./ (cosh(l .* sqrt((R + L.*s) .* (G + C.*s))));
vo_step = @(s) 10./s.* vo(s);
vo_sine = @(s) vs_sine(s).*vo(s);
time =10e-12;
%[y_step,t] = niltcv(vo_step,time,'pp');
[y_sine,t] = niltcv(vo_sine,time,'pp');
% Plot the frequency response
%plot(f / 1e9, 20*log10(abs(vo(s)))); %20*log10 for scaling
plot(t, y_sine)
xlabel('time (s)');
ylabel('Vo');
grid on;
```

Pulse with a rise time from 0 to 1V of 1ps and the same fall time and it stays at 1V for 5ps using trapezoidal pulse.

Laplace Transform of the Trapezoidal Pulse:

A trapezoidal pulse that:

- Rises linearly from 0 V to 1 V over 1 ps,
- Stays at 1 V for 5 ps,
- Falls back to 0 V over 1 ps,
  can be built from ramps and exponentials in the Laplace domain.

Time-Domain Definition

$$v_{pulse}(t) = \begin{Bmatrix} 0, & t < 0 \\ linear\ rise, & 0 \le t < 1ps \\ 1, & 1ps \le t < 6ps \\ linear\ fall, & 6ps \le t < 7ps \\ 0, & t \ge 7ps. \end{Bmatrix}$$

Total duration: 7 ps = 1 ps (rise) + 5 ps (high) + 1 ps (fall).

Constructing It via Ramp Functions as follows:

$$ramp(t) = tu(t)\,, in\ laplace\ domin \frac{1}{s^2}.$$

A linear rise and fall from 0 to 1 over $[0,\ T_r]$ can be written as

$$\frac{1}{T_r}\ [ramp(t) - ramp(t - T_r)]$$

Hence, for rise time T_r and peak duration T_P

$$v_{pulse}(t) =$$

$$\frac{1}{T_r}[ramp(t) - ramp(t - T_r)] - \frac{1}{T_r}\left[ramp\left(t - \left(T_r + T_p\right)\right) - ramp\left(t - \left(2T_r + T_p\right)\right)\right]$$

Using $\mathcal{L}\{ramp(t-a)\} = \frac{e^{-as}}{s^2}$, we get:

$$V_{pulse}(s) = \frac{1}{T_r s^2}[1 - e^{-T_r s}] - \frac{1}{T_r s^2}\left[e^{(-T_r + T_p)s} - e^{(-2T_r + T_p)s}\right].$$

The following output is obtained:

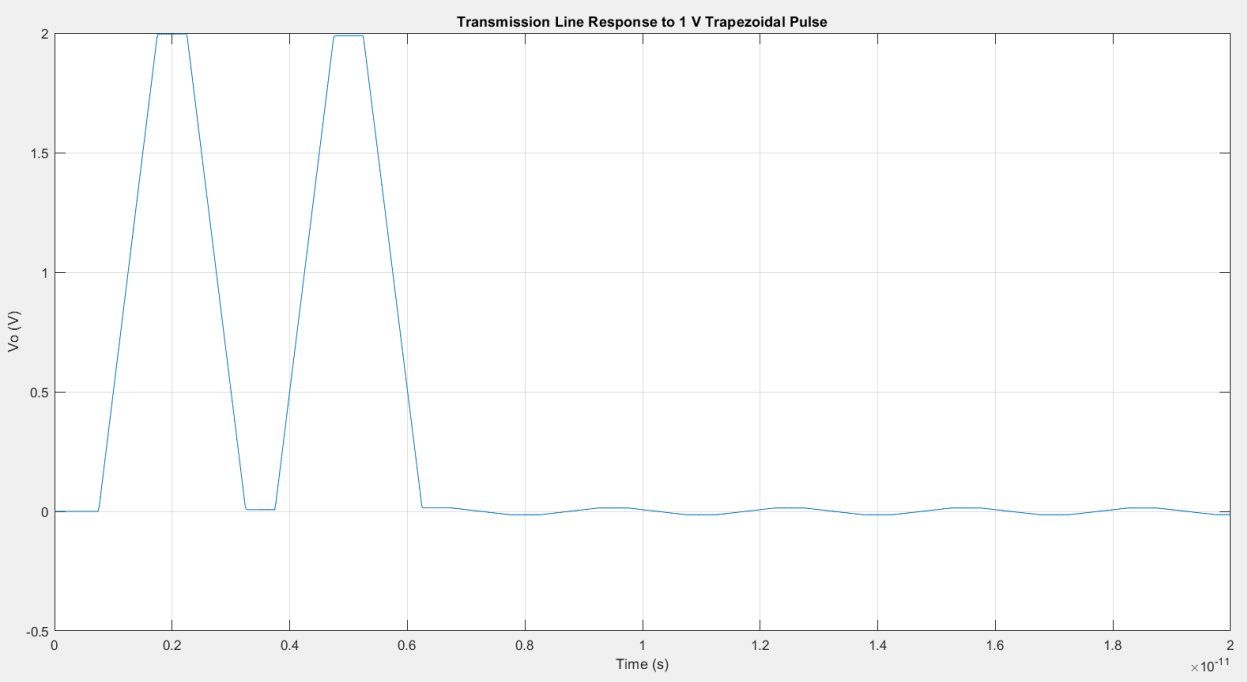

Figure 5 shows the response due to pulse input of 1 amplitude for 5 ps.

The following code is used to obtain Figure 5.

```
clear
clc
R = 1200;          % Ω/m
L = 250e-9;        % H/m
C = 1e-10;         % F/m
G = 0;             % S/m
l = 150e-6;        % m  (line length)

f_max = 100e10;    % 100 GHz
w = 2*pi*f_max;
s = 1i*w;

% Transfer function of the line
vo = @(s) 1 ./ cosh( l .* sqrt( (R + L.*s) .* (G + C.*s) ) );

%Define the trapezoidal pulse in the Laplace domain

Tr = 1e-12;  % 1 ps rise/fall
Tp = 5e-12;  % 5 ps high
Amp = 1;     % 1 V amplitude

% Laplace transform of the trapezoid
Vpulse = @(s) (Amp./(Tr*s.^2)).*(1 - exp(-Tr.*s))- (Amp./(Tr.*s.^2)).*(exp(-(Tr+Tp).*s) - exp(-
(2*Tr+Tp).*s));

vo_pulse = @(s) Vpulse(s) .* vo(s);

% Choose a simulation time that covers the entire 7 ps pulse
Tsim = 20e-12;

% Compute inverse Laplace transform numerically
```

```
[y_pulse, t] = niltcv(vo_pulse, Tsim, 'pp');

plot(t, y_pulse)
xlabel('Time (s)')
ylabel('Vo (V)')
title('Transmission Line Response to 1 V Trapezoidal Pulse')
grid on
```