EMTI

Name :- V. Krishna Sec :- 'C'

Registration Number :- RA2411004010159 Batch :- 01

Case Study Questions

1.) ans :- Electromagnetic Interference (EMI) is a Common issue in EV's, because of the close interaction between high power Circuits, motors, sensors and Control units. To maintain proper functioning and realibility, the vehicle's design must ensure EMC, meaning that all systems,

a) Shielding :- Enclose sensitive modules and Cables in Conductive Shields (or) metal housings to prevent unwanted EM radiation.

b) Proper Grounding and Bonding :- Creates a low-impedance ground network to control current return paths & minimize Voltage.

c) Cable management :- Physically separate high-current cables (from motors (or) inverters) from signal lines.

d. Filtering :- Installing EMI filters, ferrite beads, & decoupling capacitors on power and Communication lines to suppress high frequency.

e. Optimized PCB Design :- Implement solid ground planes, short trace lengths and differential routing to reduce cross-talk.

f. Component placement :- Keep high-frequency Components like inverters and switching regulators away from sensitive control Circuits.

g) Software Filtering :- Using digital filters, redundant signal and error checking algorithms to correct (or) ignore noisy signals

h) Use of Common mode Chokes :- Placing Common-mode Chokes on data and power lines helps block unwanted high-frequency noise.

i) Proper power supply Recoupling :- Using local bypass Capacitors near IC stabilizes voltage and prevents power supply fluctuations.

8, 3

In space, satellites face extreme Electromagnetic Environments like solar flares, cosmic radiation and signals from other satellites can all disrupt communication. Ensuring EMC is essential to maintain stable and uninterrupted data transmission.

a) Radiation shielding :- Enclose sensitive circuits in Conductive (or) Composite materials to block harmful high energy.

b) Frequency management :- Allocating Communication frequencies carefully to avoid overlap with other signals and using frequency

c) Filtering and Grounding :- High quality filters and well planned grounding networks help remove unwanted noise

d) Redundant systems :- Implementing backup transponders antenna and power Circuits ensures Continuous operation.

e) Error Correction Codes [ECC] :- Advanced Coding techniques like Reed-Solomon (or) Turbo Codes allow data recovery even if part of the signal is corrupted.

f) Antenna Design:-
Directional, high-gain antennas focus the signal beam towards Earth, minimizing pickup of stray radiation from space.

g) Thermal and EMI-hard packaging:-
Proper enclosure design prevents radiation induced noise and temperature variations from affecting signal.

h) Active noise Calculations:-
Some systems use DSP to identify and subtract noise pattern from the received signal.

i) Regular Calibration and monitoring:-
Continuously monitor signal integrity and make necessary adjustments through ground Control.

j) Use of Shielded Cables & Connectors:-
Employ shielded harnesses and Connectors within the satellite to prevent internal EMI Coupling between subsystem.

Q.3) A lossless 30 m long transmission line with Zo=50Ω is established between two ground stations which operate at 2 MHz. The line is terminated with a load ZL = 60+j40Ω. If u = 0.6c on the line. Write a Scilab code to plot the reflection coefficient (Γ), standing wave ratio (S) and input impedance in smith chart.

Source Code:

```
// Scilab script for Q3
// Plots: |Gamma| vs distance, SWR vs distance, and Smith-chart (Gamma plane)

clc;
clear;
close;

// Given
Z0 = 50;              // ohms
ZL = 60 + %i*40;      // load
f = 2e6;              // Hz
c = 3e8;              // m/s
u = 0.6 * c;          // propagation velocity
L = 30;               // line length in meters

// Derived
lambda = u / f;
beta = 2 * %pi / lambda;

// Reflection coefficient at load
Gamma_load = (ZL - Z0) / (ZL + Z0);
Gamma_load_mag = abs(Gamma_load);
```

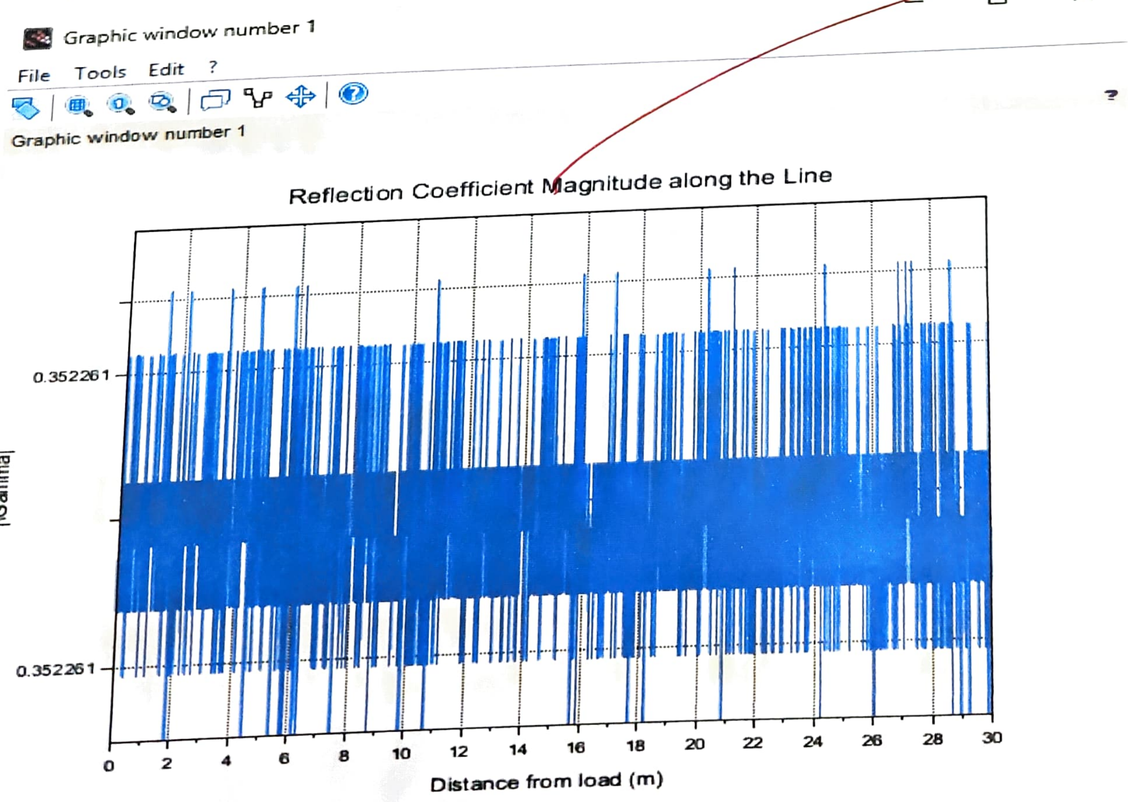
1241100. MAVD....

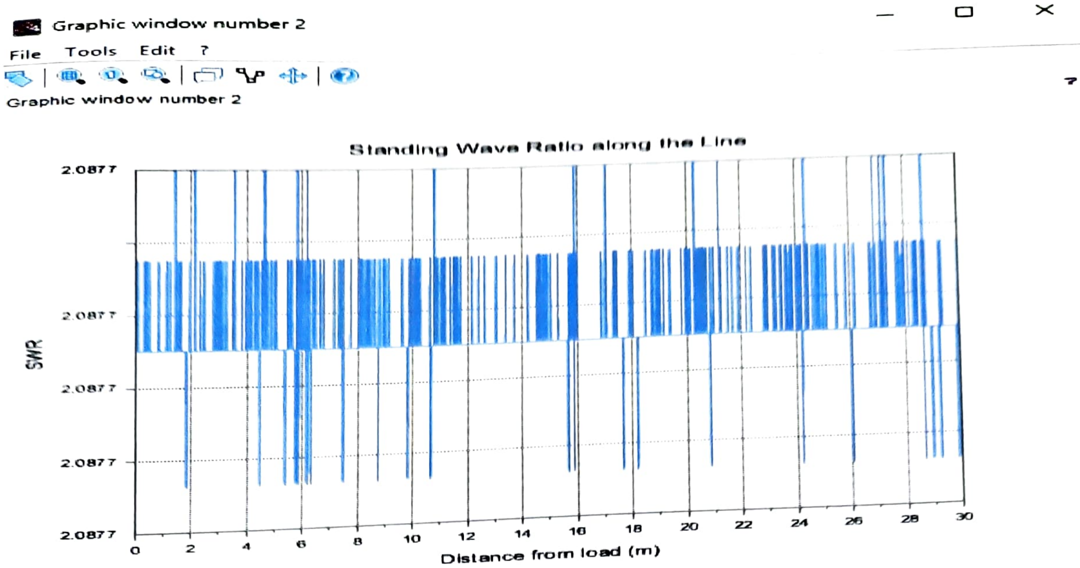
```
Gamma_load_phase_deg = atan(imag(Gamma_load)/real(Gamma_load)) * 180
/ %pi; // approximate angle
SWR_load = (1 + Gamma_load_mag) / (1 - Gamma_load_mag);

// Print numeric checks
disp("Gamma at load: " + string(Gamma_load));
disp(" |Gamma| at load: " + string(Gamma_load_mag));
disp(" Phase (deg) approx: " + string(Gamma_load_phase_deg));
disp(" SWR at load: " + string(SWR_load));

// Sample points along line (0 -> L)
d = 0:0.01:L; // meter resolution (adjust if needed)

// Input impedance along line (lossless)
Zin = Z0 * (ZL + %i*Z0 .* tan(beta .* d)) ./ (Z0 + %i*ZL .* tan(beta .* d));

// Reflection coefficient along line
Gamma_d = (Zin - Z0) ./ (Zin + Z0);

// Plot |Gamma| along the line
figure(1);
plot(d, abs(Gamma_d));
xlabel("Distance from load (m)");
ylabel("|\Gamma|");
xtitle("Reflection Coefficient Magnitude along the Line");
xgrid();

// Plot SWR along the line
```

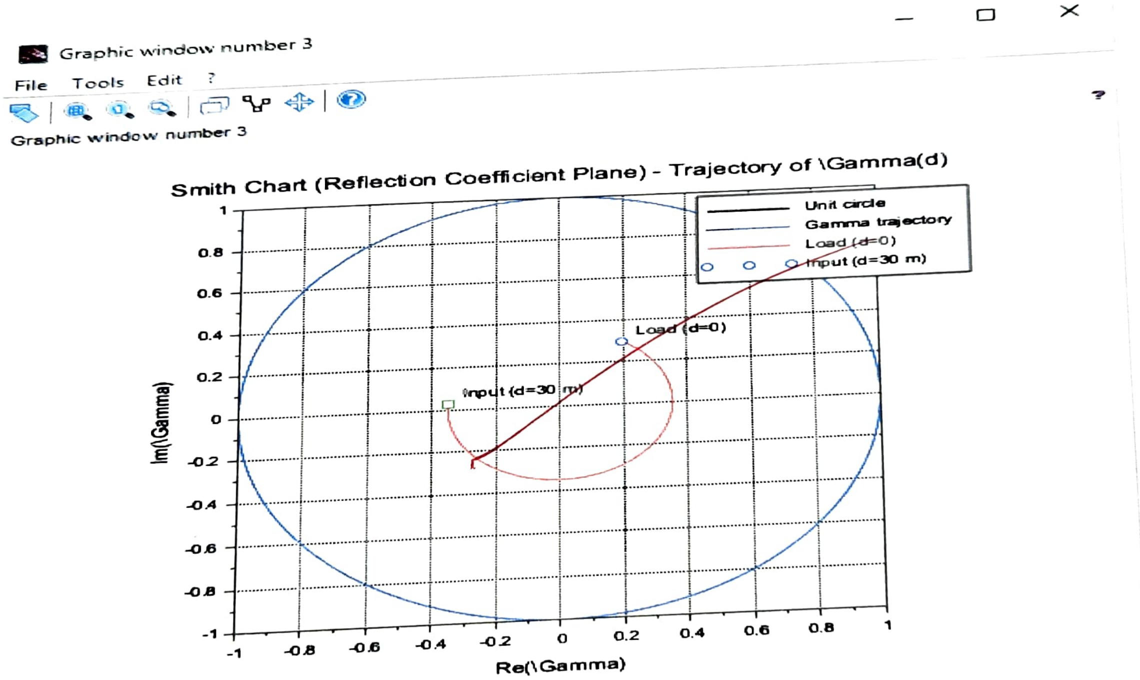
```
SWR_d = (1 + abs(Gamma_d)) ./ (1 - abs(Gamma_d));
figure(2);
plot(d, SWR_d);
xlabel("Distance from load (m)");
ylabel("SWR");
xtitle("Standing Wave Ratio along the Line");
xgrid();

// Smith chart (Gamma plane)
// Unit circle
theta = 0:0.01:2*%pi;
cx = cos(theta);
cy = sin(theta);

// Prepare points for overlay: Gamma trajectory
realG = real(Gamma_d);
imagG = imag(Gamma_d);

// Compute Gamma at input end (d = L) for annotation
Gamma_input = (Zin($) - Z0) / (Zin($) + Z0); // Zin($) = last element
// (Gamma_load already computed for d=0)

// Plot unit circle and Gamma trajectory on same axes
figure(3);
plot(cx, cy, 'k'); // unit circle
// Overlay trajectory and key points in one plot call
plot(cx, cy, realG, imagG, 'r-', real(Gamma_load), imag(Gamma_load), 'bo',
real(Gamma_input), imag(Gamma_input), 'gs');
```

```
xlabel("Re(\Gamma)");
ylabel("Im(\Gamma)");
xtitle("Smith Chart (Reflection Coefficient Plane) - Trajectory of \Gamma(d)");
xgrid();
// keep equal axis scale
a = gca();
a.isoview = "on";
legend(["Unit circle", "Gamma trajectory", "Load (d=0)", "Input (d=30 m)"], 1);

// Optionally annotate numeric values near points
// (Simple text annotations)
xstring(real(Gamma_load)+0.03, imag(Gamma_load), "Load (d=0)");
xstring(real(Gamma_input)+0.03, imag(Gamma_input), "Input (d=30 m)");
```

Output:







Scilab 2026.0.0 Console

File Edit Control Applications ?

Scilab 2026.0.0 Console

```
"Gamma at load: 0.1970803+%i*0.2919708"
" |Gamma| at load: 0.3522607"
" Phase (deg) approx: 55.98065"
" SWR at load: 2.0876619"

-->
```

Q.4) In a satellite base station, a load of 100 + j150 Ω is connected to a 75Ω lossless line. Write a Scilab code to plot the Reflection coefficient (Γ), SWR value and input impedance (Zin) at 0.4λ from the load.

Source Code:

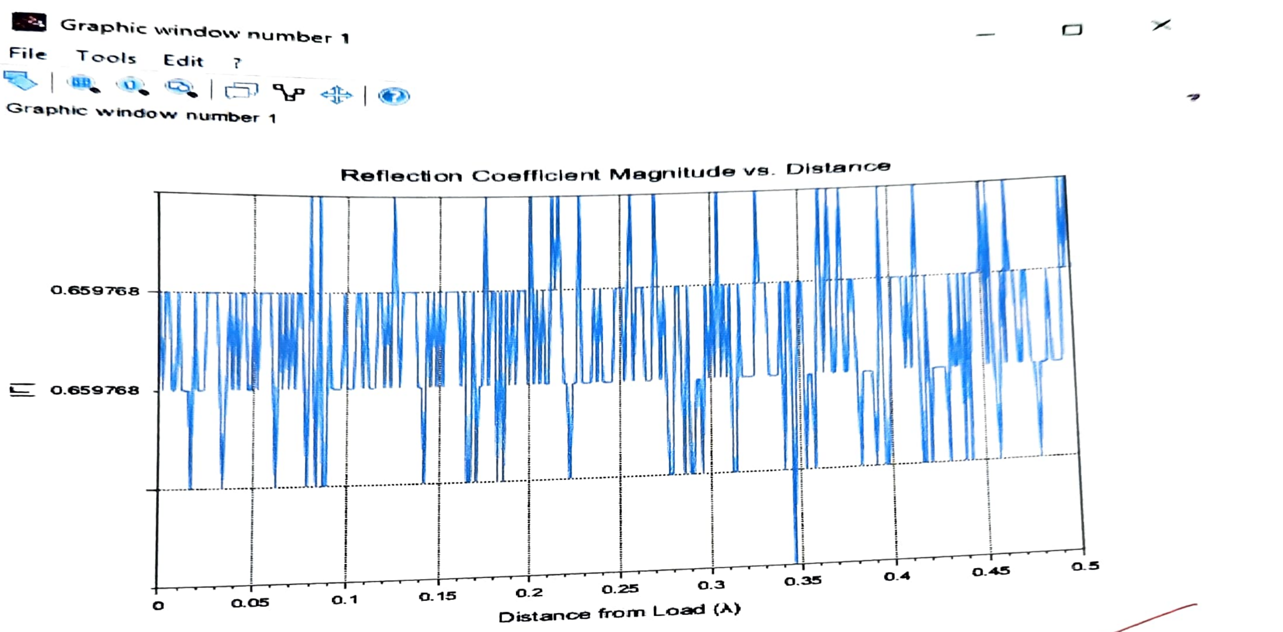
```
// Q4 - Electromagnetic Theory and Interference
// Satellite base station simulation
// Compute and plot Reflection Coefficient, SWR, and Zin at 0.4λ

clc;
clear;
close;

// Given data
Z0 = 75;              // Characteristic impedance (ohms)
ZL = 100 + %i*150;    // Load impedance (ohms)
lambda = 1;           // Normalized wavelength (unit value)
d = 0.4 * lambda;     // Distance from load (in wavelengths)

// Reflection Coefficient at Load
Gamma_L = (ZL - Z0) / (ZL + Z0);
Gamma_mag = abs(Gamma_L);
Gamma_phase_deg = atan(imag(Gamma_L)/real(Gamma_L)) * 180 / %pi;

// Standing Wave Ratio
SWR = (1 + Gamma_mag) / (1 - Gamma_mag);

// Input Impedance at distance d = 0.4λ
```

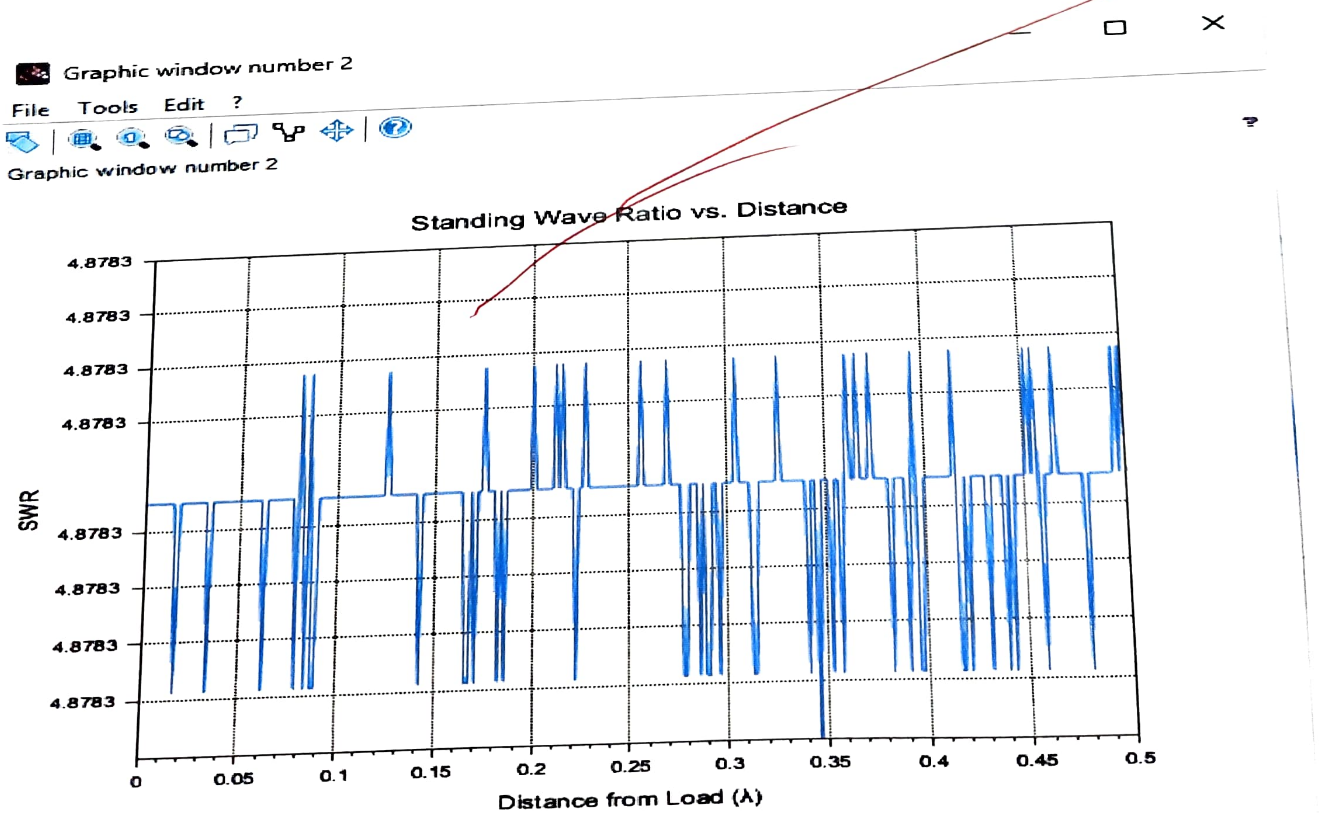
```
beta = 2 * %pi / lambda;            // Phase constant (rad/m)
Zin = Z0 * (ZL + %i*Z0 * tan(beta*d)) / (Z0 + %i*ZL * tan(beta*d));

// Display results
disp("--------------------------------------------------");
disp("Reflection Coefficient (Γ_L): " + string(Gamma_L));
disp("|Γ_L| : " + string(Gamma_mag));
disp("Phase of Γ_L (degrees): " + string(Gamma_phase_deg));
disp("Standing Wave Ratio (SWR): " + string(SWR));
disp("Input Impedance at 0.4λ (Zin): " + string(Zin));
disp("--------------------------------------------------");

// For visualization, sweep along 0 to 0.5λ for Γ and SWR variation
d_values = linspace(0, 0.5*lambda, 300);
Zin_values = Z0 * (ZL + %i*Z0 .* tan(beta .* d_values)) ./ (Z0 + %i*ZL .*
tan(beta .* d_values));
Gamma_d = (Zin_values - Z0) ./ (Zin_values + Z0);

// Plot |Γ| vs. distance
figure(1);
plot(d_values, abs(Gamma_d));
xlabel("Distance from Load (λ)");
ylabel("|Γ|");
title("Reflection Coefficient Magnitude vs. Distance");
xgrid();

// Plot SWR vs. distance
SWR_d = (1 + abs(Gamma_d)) ./ (1 - abs(Gamma_d));
```

```
figure(2);
plot(d_values, SWR_d);
xlabel("Distance from Load (λ)");
ylabel("SWR");
title("Standing Wave Ratio vs. Distance");
xgrid();

// Smith Chart (Γ plane)
theta = 0:0.01:2*%pi;
cx = cos(theta);
cy = sin(theta);
realG = real(Gamma_d);
imagG = imag(Gamma_d);

figure(3);
plot(cx, cy, 'k'); // unit circle
plot(realG, imagG, 'r-');
xlabel("Re(Γ)");
ylabel("Im(Γ)");
title("Smith Chart (Reflection Coefficient Plane)");
xgrid();
a = gca();
a.isoview = "on";

// Mark important points
plot(real(Gamma_L), imag(Gamma_L), 'bo');
Gamma_input = (Zin - Z0) / (Zin + Z0);
```

```
plot(real(Gamma_input), imag(Gamma_input), 'gs');
xstring(real(Gamma_L)+0.03, imag(Gamma_L), "Load (d=0)");
xstring(real(Gamma_input)+0.03, imag(Gamma_input), "Input (d=0.4λ)");
legend(["Unit circle", "Γ trajectory", "Load", "Input"], 1);
```

Output:

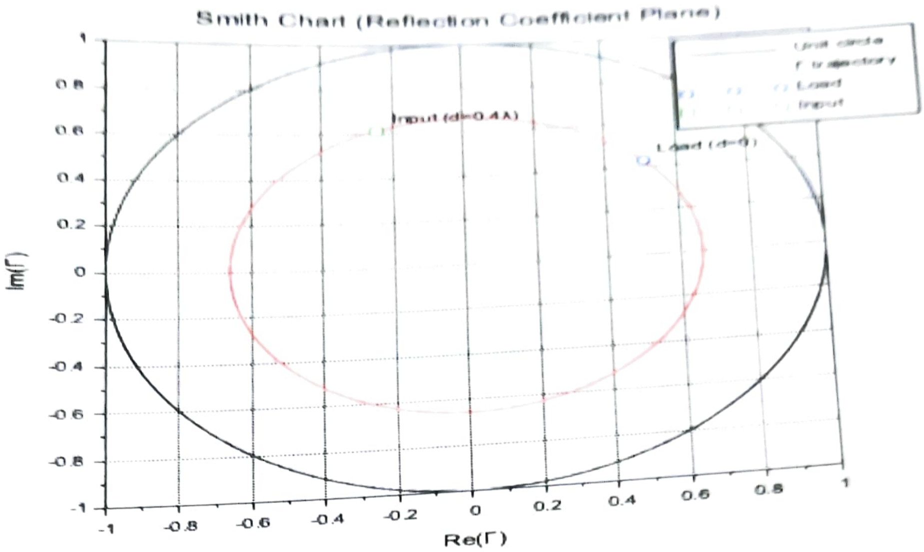




Graphic window number 3

File Tools Edit ?

Graphic window number 3



Scilab 2026.0.0 Console

File Edit Control Applications ?

Scilab 2026.0.0 Console

```
"--------------------------------------------------------"
"Reflection Coefficient (Γ_L): 0.5058824+%i*0.4235294"
"|Γ_L| : 0.6597682"
"Phase of Γ_L (degrees): 39.936383"
"Standing Wave Ratio (SWR): 4.8783458"
"Input Impedance at 0.4λ (Zin): 21.964531+%i*47.60816"
"--------------------------------------------------------"

-->
```