Student: Jacob Watts

Course: ECE 3710

Subject: Lab 4, 1-port VNA calibration

Date: March 6, 2023

1 Introduction

In this exercise we will plot the input impedance versus position. We will be using a Network Analysis 8510 TRL Calibration Equipment from Hewlett-Packard. We will be using special connectors to properly calibrate the equipment. The antenna that I will be testing is a amateur radio antenna dual band.

$$Z_in(l) = Z_0\frac{Z_L + jZ_0tanBl}{Z_0 + jZ_LtanBl} \tag{1}$$

After we will reconstruct data of the return loss plot of a amateur radio dual band antenna.

2 Plot

Below are the required plots.

2.1 Return Loss Plot Network Analyzer

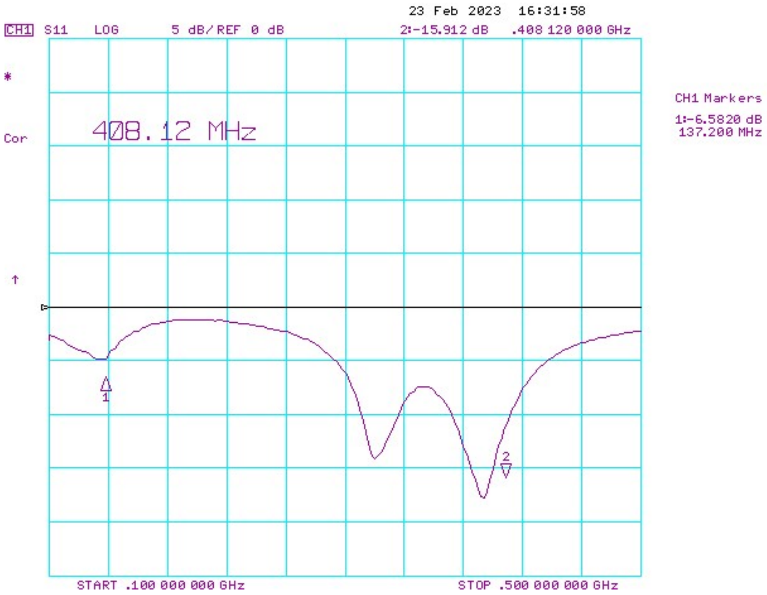

Figure 1: Return Loss Plot Network Analyzer

2.2 Return Loss Plot of Dual Band Antenna

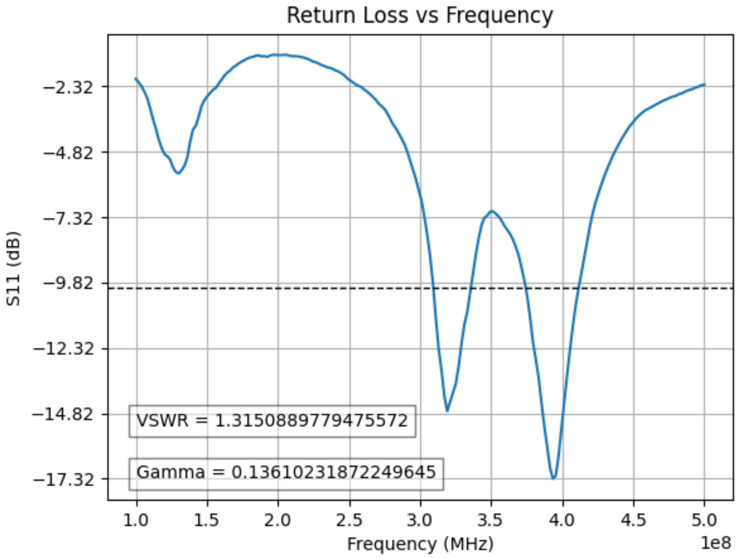

Figure 2: Return Loss Plot

2.3 VSWR of Dual Band Antenna

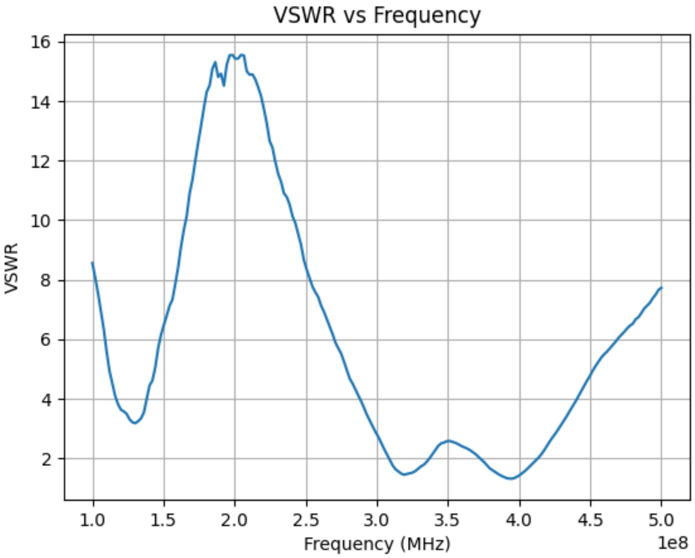

Figure 3: VSWR of Dual Band Antenna

3 Python

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sys

if len(sys.argv) < 2:
    print("Usage:")
    print("    $ python %s <data>" %
          sys.argv[0])
    sys.exit(0)
filename = sys.argv[1]

def read_data(filename):
    #read line 6 from the data, read only the number values
    with open(filename) as f:
        for i, line in enumerate(f):
            if i == 6:
                seg = line.split()
                break
    start_freq = float(seg[1])
    stop_freq = float(seg[2])
    num_points = int(seg[3]) -1

    #reads data from file converts to numpy
    real_str = pd.read_csv(filename, skiprows=9,skipfooter=2, sep=',', usecols=[0], engine='python')
    img_str = pd.read_csv(filename, skiprows=9,skipfooter=2, sep=',', usecols=[1], engine='python')
    real = np.array(real_str,dtype=float)
    img = np.array(img_str,dtype=float)
    #convert to 1d array
    real = real.flatten()
    img = img.flatten()
    return real, img, start_freq, stop_freq, num_points

def s11_computation(real, img):
    s11 = np.sqrt(np.power(real,2) + np.power(img,2))
    return s11

def rl_computation(s11):
    db = 20*np.log10(s11)
    return db

def vswr_computation(s11):
    vswr = (1+s11)/(1-s11)
    return vswr

def fig1_data():
    # number of samples
    ns = 1000
    lmbda = 1
    L = 1 * lmbda
    gamma = np.array([(1/5)])
    points = np.linspace(0, L, ns)
    Z_l = 75
    Z_0 = 50
    Z_in = Z_0*((Z_l + Z_0*1j*np.tan(2*np.pi*points))/(Z_0 + Z_l*1j*np.tan(2*np.pi*points)))
    plt.grid(True)
    #title name needs to have the symbol for ohm in it
    plt.title('R = 50 ohm-Term line, Z_0 = 50 ohm')
    plt.xlabel('Lambda ( )')
    plt.ylabel('Z_in(ohm))')
    #x axis go in steps of .1
    plt.xticks(np.arange(0, 1.1, step=0.1))
    plt.ylim(-10, 110)
    #y axis go in steps of 10
    plt.yticks(np.arange(0, 110, step=10))
    plt.plot(points, Z_in)
    #save figure to docs folder outside of src
    plt.savefig("fig1.png")
    plt.show()

def fig2_plt(db, gamma, vswr, start_f, end_f, points):
    #find the value of the minimum value in the array
    min_value_db = np.amin(db)
    #create title
    plt.title('Return Loss vs Frequency')
    #turn on legend for each line
    plt.xlabel('Frequency (MHz)')
    plt.ylabel('S11 (dB)')
    #y axis go in steps of 10
    #plt.ylim(-25,0)
    plt.grid()
    plt.plot(np.linspace(start_f,end_f,points),db)
    #plt.xlim(2,3)
    #y axis go in steps of .5
    plt.yticks(np.arange(min_value_db, 0, step=2.5))
    #x axis go in steps of .1
    #plt.xticks(np.arange(, 3.1, step=0.1))
    #add a horizontal line at -10 dB dotted line
    plt.axhline(y=-10, color='black', linestyle='--', linewidth=1)
    #find the idx position where the plot crosses -10 dB
    idx = np.argwhere(np.diff(np.sign(db + 10))).flatten()

    #add gamma and vswr in a box
    plt.text(start_f+10, min_value_db, 'Gamma = ' + str(gamma), bbox=dict(facecolor='white', alpha=0.5))
    plt.text(start_f+10, min_value_db+2, 'VSWR = ' + str(vswr), bbox=dict(facecolor='white', alpha=0.5))
    plt.savefig(filename[:-3] + ".png")

    plt.show()

def find_gamma(db):
    return np.power(10,np.amin(db)/20)

def find_vswr(gamma):
    return (1+gamma)/(1-gamma)

def main():
    print('running plotting program')
    #fig1_data()
    real, img, start_f, end_f, points = read_data(filename)
    print('start frequency: ', start_f)
    print('end frequency: ', end_f)
    print('number of points: ', points)

    s11 = s11_computation(real, img)
    db = rl_computation(s11)
    gamma_value = find_gamma(db)
    vswr = find_vswr(gamma_value)
    fig2_plt(db,gamma_value,vswr, start_f, end_f, points)

if __name__ == "__main__":
    main()
```