ECE 802 - 603: Microwave and Milimeter Waves: Homework 1

Michael Saybolt

3/29/2016

Learning how to use LaTeX!

Contents

1 De-Embedding Data

De-embedding data will take measured S-parameters of a system and give us the S-parameters of the DUT without the error induced by the connection to the device. Once measured data is acquired for the DUT, and control measurements for an open connection (no DUT), a short (to ground), and through (a section of line), the S-parameters of the error, or error boxes can be calculated. From there, the inverse of the error box is multiplied by the S-parameters of the DUT, and then by the non-inverted S-parameters of the error box, assuming that the error boxes are symmetric. The result is the de-embedded S-parameters of the DUT. The following explains this procedure in greater detail.

1.1 Error Box S-parameters

The procedure for getting the S-parameters of the error box is found around/on p. 201 of Pozar's Microwave Engineering. Since $S_{11} = S_{22}$, and $S_{12} = S_{21}$, the equations are combined and S_{11} and S_{12} for the error box are obtained:

$$S_{11_{Error}} = \frac{S_{11_{Short}} - 2 * S_{11_{Through}} + S_{11_{Open}}}{S_{11_{Open}} - 2 * S_{12_{Through}} - S_{11_{Short}}} \tag{1}$$

$$S_{12_{Error}} = \sqrt{S_{12_{Through}} * (1 - S_{11_{Error}}^2)} \tag{2}$$

$$S_{21_{Error}} = S_{12_{Error}} \tag{3}$$

$$S_{22_{Error}} = S_{11_{Error}} \tag{4}$$

1.2 De-embedding without MATLAB RF Toolbox

The first time around, it was not known that MATLAB has an RF toolkit that does many of the things in this assignment. Since 90% of the time was spent doing it that way, the works are still presented.

1.2.1 Functions

A MATLAB function was made to import .s2p files into an array type in MATLAB. It should be noted that the *.s2p import function had trouble with a s2p file so the files had to be modified in Notepad and the headers removed. It appeared that the *importdata* function in MATLAB only supports one type of comment line, and having lines that started with "" as well as "#" resulted in confusion and data loss.

Listing 1: *.s2p Import

```
function [ S11, S21, S12, S22, freq ] = ECE802_S2Pread( ...
    fileName )
%Read full S2P matrix
%Note: make modified files with headers it knows how to deal ...
    with
%      - More than one type of "comment" lines seems to ...
    confuse importdata

[A, delimiterOut, headerlinesOut] = importdata(fileName);

%import each column of data part of struct into ...
    vectors/matricies
freq = A.data(:,1);

S11(:,1) = A.data(:,2);
S11(:,2) = A.data(:,3);

S21(:,1) = A.data(:,4);
S21(:,2) = A.data(:,5);

S12(:,1) = A.data(:,6);
S12(:,2) = A.data(:,7);

S22(:,1) = A.data(:,8);
S22(:,2) = A.data(:,9);
end
```

More functions were made to convert S-parameters to ABCD parameters in order to perform the de-embedding operations on them, and then to convert ABCD back to S-parameters.

Listing 2: S-parameters to ABCD

```
function [ ABCD ] = StoABCD( S, Z0 )
% StoABCD convert S matrix to ABCD parameters
% V1.0
% [A, B; C, D] = [S11, S12; S21, S22]

ABCD(1,1) = ((1+S(1,1))*(1-S(2,2))+S(1,2)*S(2,1))/(2*S(2,1));
ABCD(1,2) = Z0*((1+S(1,1)*(1+S(2,2))-S(1,2)*S(2,1))/(2*S(2,1)));
ABCD(2,1) = ...
    1/Z0*(((1-S(1,1))*(1-S(2,2))-S(1,2)*S(2,1))/(2*S(2,1)));
ABCD(2,2) = (((1-S(1,1))*(1+S(2,2))+S(1,2)*S(2,1))/(2*S(2,1)));

end
```

Listing 3: ABCD to S-parameters

```
function [ S ] = ABCDtoS( ABCD, Z0 )
% ABCDtoS convert ABCD mat to Spara
% V1.0
% [S11, S12; S21, S22] = [A, B; C, D]

```

```
    % reassign for readability and ease of coding
    A = ABCD(1,1); B = ABCD(1,2); C = ABCD(2,1); D = ABCD(2,2);

    S(1,1) = (A + B/Z0 - C*Z0 - D)/(A + B/Z0 + C*Z0 + D);
    S(1,2) = (2*(A*D - B*C))/(A + B/Z0 + C*Z0 + D);
    S(2,1) = 2/(A + B/Z0 + C*Z0 + D);
    S(2,2) = (-A + B/Z0 - C*Z0 + D)/(A + B/Z0 + C*Z0 + D);

    end
```

1.2.2 Main Code

These functions are called repeatedly inside the main body of the code. Most of it runs inside a giant *for* loop that does the de-embedding operation on a 2x2 matrix for each frequency. These are assembled at the end back into the 2x2xN matrix for plotting along the frequency slice.

Listing 4: De-embedding MATLAB Code

```
%802 deembedding shit draft
%folderName = [pwd,'\De-Embedding Data']

clear; close all;

% LDR Data
[DiodeS11, DiodeS21, DiodeS12, DiodeS22, freq] = ...
    ECE802_S2Pread('diode_mod.s2p');
[OpenS11, OpenS21, OpenS12, OpenS22] = ...
    ECE802_S2Pread('open_mod.s2p');
[ResistorS11, ResistorS21, ResistorS12, ResistorS22] = ...
    ECE802_S2Pread('resistor_mod.s2p');
[ShortS11, ShortS21, ShortS12, ShortS22] = ...
    ECE802_S2Pread('short_mod.s2p');
[ThroughS11, ThroughS21, ThroughS12, ThroughS22] = ...
    ECE802_S2Pread('through_mod.s2p');

Z0 = 50;

%Whole program is contained in for loop and runs for each ...
    frequency
for ii = 1:length(freq)
%Convert used data from mag<phase to re+j*im
%DiodeS(ii,1,1) = magphase2cplex(DiodeS11(ii,1), DiodeS11(ii,2))
DiodeS(ii,1,1) = ...
    db2magPwr(DiodeS11(ii,1))*exp(db2magPwr(DiodeS11(ii,2))*sqrt(-1));
DiodeS(ii,2,1) = ...
    db2magPwr(DiodeS21(ii,1))*exp(db2magPwr(DiodeS21(ii,2))*sqrt(-1));
DiodeS(ii,1,2) = ...
    db2magPwr(DiodeS12(ii,1))*exp(db2magPwr(DiodeS12(ii,2))*sqrt(-1));
DiodeS(ii,2,2) = ...
    db2magPwr(DiodeS22(ii,1))*exp(db2magPwr(DiodeS22(ii,2))*sqrt(-1));

ResistorS(ii,1,1) = ...
    db2magPwr(ResistorS11(ii,1))*exp(db2magPwr(ResistorS11(ii,2))*sqrt(-1));
```

```
25      ResistorS(ii,2,1) = ...
            db2magPwr(ResistorS21(ii,1))*exp(db2magPwr(ResistorS21(ii,2))*sqrt(-1));
26      ResistorS(ii,1,2) = ...
            db2magPwr(ResistorS12(ii,1))*exp(db2magPwr(ResistorS12(ii,2))*sqrt(-1));
27      ResistorS(ii,2,2) = ...
            db2magPwr(ResistorS22(ii,1))*exp(db2magPwr(ResistorS22(ii,2))*sqrt(-1));
28
29      ThroughS(ii,1,1) = ...
            db2magPwr(ThroughS11(ii,1))*exp(db2magPwr(ThroughS11(ii,2))*sqrt(-1));
30      ThroughS(ii,2,1) = ...
            db2magPwr(ThroughS21(ii,1))*exp(db2magPwr(ThroughS21(ii,2))*sqrt(-1));
31      ThroughS(ii,1,2) = ...
            db2magPwr(ThroughS12(ii,1))*exp(db2magPwr(ThroughS12(ii,2))*sqrt(-1));
32      ThroughS(ii,2,2) = ...
            db2magPwr(ThroughS22(ii,1))*exp(db2magPwr(ThroughS22(ii,2))*sqrt(-1));
33
34      OpenS(ii,1,1) = ...
            db2magPwr(OpenS11(ii,1))*exp(db2magPwr(OpenS11(ii,2))*sqrt(-1));
35      OpenS(ii,2,1) = ...
            db2magPwr(OpenS11(ii,1))*exp(db2magPwr(OpenS11(ii,2))*sqrt(-1));
36      OpenS(ii,1,2) = ...
            db2magPwr(OpenS11(ii,1))*exp(db2magPwr(OpenS11(ii,2))*sqrt(-1));
37      OpenS(ii,2,2) = ...
            db2magPwr(OpenS11(ii,1))*exp(db2magPwr(OpenS11(ii,2))*sqrt(-1));
38
39      ShortS(ii,1,1) = ...
            db2magPwr(ShortS11(ii,1))*exp(db2magPwr(ShortS11(ii,2))*sqrt(-1));
40      ShortS(ii,2,1) = ...
            db2magPwr(ShortS11(ii,1))*exp(db2magPwr(ShortS11(ii,2))*sqrt(-1));
41      ShortS(ii,1,2) = ...
            db2magPwr(ShortS11(ii,1))*exp(db2magPwr(ShortS11(ii,2))*sqrt(-1));
42      ShortS(ii,2,2) = ...
            db2magPwr(ShortS11(ii,1))*exp(db2magPwr(ShortS11(ii,2))*sqrt(-1));
43
44      Test(ii) = ...
            db2magPwr(ResistorS11(ii,1))*exp(db2magPwr(ResistorS11(ii,2))*sqrt(-1));
45      %Test(ii,1,1) = ...
            db2magPwr(ResistorS11(ii,1))*exp(db2mag(ResistorS11(ii,2))*sqrt(-1));
46      %Test(ii,2,1) = ...
            ResistorS21(ii,1)*exp(ResistorS21(ii,2)*sqrt(-1));
47
48      % Solve:
49      % Analytical Work: (P201 Pozar)
50      % T11 = S11 + S22(S12^2/(1-S22^2)), S11 = S12
51      % T12 = (S12^2/(1-S22^2))
52      %
53      % T11 = S11 + S22*T12 = S22(1+T12)
54      % ==>  S22 = T11/(1+T12)
55      %
56      % S12^2 = T12*(1-S22^2) = T12(1-(T11/(1+T12))^2)
57      % ==> S12 = sqrt(T12(1-(T11/(1+T12))^2)) = sqrt(T12(1-S22^2))
58
59      % Get Error box S param
60      ErrorS(ii,1,1) = ...
            (ShortS(ii,1,1)-2*ThroughS(ii,1,1)+OpenS(ii,1,1))/ ...
61      (OpenS(ii,1,1)-2*ThroughS(ii,1,2)-ShortS(ii,1,1));
62      ErrorS(ii,1,2) = sqrt(ThroughS(ii,1,2)*(1-ErrorS(ii,1,1)^2));
```

```
63      ErrorS(ii,2,1) = ErrorS(ii,1,2);
64      ErrorS(ii,2,2) = ErrorS(ii,1,1);
65
66      % Convert all shit to ABCD to manipulate
67      %When drawing data from one 2x2 slice of n x m x p ...
            ubermatrix, use squeeze to
68      %form the 2x2 (otherwise is a row of 4 elements for some ...
            stupid reason)
69      %ResistorABCD(ii,:,:) = StoABCD(squeeze(ResistorS(ii,:,:)),Z0);
70      ResistorABCD(ii,:,:) = s2a(squeeze(ResistorS(ii,:,:)));
71      DiodeABCD(ii,:,:) = StoABCD(squeeze(DiodeS(ii,:,:)),Z0);
72      ErrorABCD(ii,:,:) = StoABCD(squeeze(ErrorS(ii,:,:)),Z0);
73      %%% ASK DR CHAHAL WHY USE INVERSE AGAIN (see p202 Pozar) %%%
74      ErrorDBCA(ii,:,:) = [ErrorABCD(ii,1,1) ErrorABCD(ii,1,2); ...
75      ErrorABCD(ii,2,1) ErrorABCD(ii,2,2)];
76
77      %De-embed
78      %ResistorDemABCD(ii,:,:) = squeeze(ErrorABCD(ii,:,:))\ ...
79      %squeeze(ResistorABCD(ii,:,:))/squeeze(ErrorABCD(ii,:,:));
80      ResistorDemABCD(ii,:,:) = inv(squeeze(ErrorABCD(ii,:,:)))* ...
81      squeeze(ResistorABCD(ii,:,:))*inv(squeeze(ErrorDBCA(ii,:,:)));
82
83
84      %DiodeDemABCD(ii,:,:) = squeeze(ErrorABCD(ii,:,:))\ ...
85      %squeeze(DiodeABCD(ii,:,:))/squeeze(ErrorABCD(ii,:,:));
86      DiodeDemABCD(ii,:,:) = inv(squeeze(ErrorABCD(ii,:,:)))* ...
87      squeeze(DiodeABCD(ii,:,:))*inv(squeeze(ErrorDBCA(ii,:,:)));
88
89      % Convert back to S parameters
90      %ResistorDemS(ii,:,:) = ...
            ABCDtoS(squeeze(ResistorDemABCD(ii,:,:)),Z0);
91      ResistorDemS(ii,:,:) = a2s(squeeze(ResistorDemABCD(ii,:,:)));
92      DiodeDemS(ii,:,:) = ABCDtoS(squeeze(DiodeDemABCD(ii,:,:)),Z0);
93      end
94
95      % Plot
96      plottr(freq, ResistorS11(:,1),'Frequency (Hz)', '|S11|', ...
            'Resistor S11(nonimbed, straight mag from file)')
97      %plottr(freq, ...
            mag2dbPwr(sqrt(real(Test(:)).^2+imag(Test(:)).^2)), ...
            'Frequency (Hz)', '|S11|', 'Resistor S11(test)')
98      plottr(freq, mag2dbPwr(abs(Test(:))), 'Frequency (Hz)', ...
            '|S11|', 'Resistor S11(nonimbed, db->mag,mag->db)')
99
100     %plottr(freq, abs(Test(:,2,1)), 'Frequency (Hz)', '|S21|', ...
            'Resistor S21(test)')
101     %Spara resistor
102     plottr(freq, mag2dbPwr(abs(ResistorDemS(:,1,1))), 'Frequency ...
            (Hz)', '|S11|', 'Resistor S11')
103     plottr(freq, mag2dbPwr(abs(ResistorDemS(:,1,2))), 'Frequency ...
            (Hz)', '|S12|', 'Resistor S12')
104     plottr(freq, abs(ResistorDemS(:,2,1)), 'Frequency (Hz)', ...
            '|S21|', 'Resistor S21')
105     plottr(freq, abs(ResistorDemS(:,2,2)), 'Frequency (Hz)', ...
            '|S22|', 'Resistor S22')
106
107     %Spara diode
```

```
108     plottr(freq, abs(DiodeDemS(:,1,1)), 'Frequency (Hz)', ...
            '|S11|', 'Diode S11')
109     plottr(freq, abs(DiodeDemS(:,1,2)), 'Frequency (Hz)', ...
            '|S12|', 'Diode S12')
110     plottr(freq, abs(DiodeDemS(:,2,1)), 'Frequency (Hz)', ...
            '|S21|', 'Diode S21')
111     plottr(freq, abs(DiodeDemS(:,2,2)), 'Frequency (Hz)', ...
            '|S22|', 'Diode S22')
112
113     % Export data for ADS
114     filename = 'ResistorSpara.xlsx';
115     xlswrite(filename,[freq,abs(ResistorDemS(:,1,1)),angle(ResistorDemS(:,1,1)), ...
            ...
116     abs(ResistorDemS(:,1,2)), angle(ResistorDemS(:,1,2)), ...
            abs(ResistorDemS(:,2,1)), ...
117     angle(ResistorDemS(:,2,1)), abs(ResistorDemS(:,2,2)), ...
            angle(ResistorDemS(:,2,2))]);
118
119     filename = 'DiodeSpara.xlsx';
120     xlswrite(filename,[freq,abs(DiodeDemS(:,1,1)),angle(DiodeDemS(:,1,1)), ...
            ...
121     abs(DiodeDemS(:,1,2)), angle(DiodeDemS(:,1,2)), ...
            abs(DiodeDemS(:,2,1)), ...
122     angle(DiodeDemS(:,2,1)), abs(DiodeDemS(:,2,2)), ...
            angle(DiodeDemS(:,2,2))]);
```

1.2.3 Postprocessing with Excel and Python

The output of the MATLAB code is in .xls format for Microsoft Excel. It needs to be converted back to .s2p for importing into ADS to make an equivalent model.

First, the file is opened in Excel, and saved as a comma delimited file, *.csv.

A Python script was made by Chris Oakley to convert *.csv to *.s1p for some other application called "csv_to_s1p.py". It was modified a bit to accept more inputs and re-branded as "csv_to_s2p.py". This version of the script also accepts user input through the command prompt to specify the input and output filenames, however the working directory still must be specified in the file. Below is the code.

Listing 5: Python Based *.csv to *.s2p Converter

```
1   # -*- coding: utf-8 -*-
2   """
3   Created on Mon Feb 15 10:58:19 2016
4
5   @author: oakleych
6   @s2p_modder: sayboltm
7   csv_to_s2p.py
8   note: Crude but it works
9   """
10
11  import os.path
12
```

```
13      def getNumLines(fname, skiprows):
14      lines = 0
15
16      if os.path.isfile(fname):
17      f = open(fname, 'r')
18
19      for n in range(skiprows):
20      f.readline()
21
22      for line in f:
23      if line[0] == 'E':
24      break
25      else:
26      lines += 1
27
28      f.close()
29
30      return lines
31
32      num_ports = 1
33
34
35      #Set this to the directory where the data is stored
36      #basedir = r"M:\SiCRFPAPER\spara/"
37      basedir = r"C:\Users\Mike\Dropbox\Documents\MS EE\SS16\ECE ...
            802 - 603 (MWndMM Ciruits)\HW\HW1/"
38      print('Present working directory is:\n', basedir)
39
40      #Input file name
41      #csv_name = 'ResistorSpara.csv'
42      print('\nInput file to get converted including .csv extension:')
43      csv_name = input()
44
45      #Output file name
46      #out_name = 'ResistorSpara.s2p'
47      print('\nInput desired output file name, again with extension:')
48      out_name = input()
49
50      # Note the 8 at the end that skips some lines
51      data_lines = getNumLines(basedir + csv_name, 8)
52
53      csv_file = open(basedir + csv_name, 'r')
54
55      #Number of comment lines in CSV file
56      comment_rows = 5
57
58      comment_lines = ''
59      freq_data = []
60      mag_data11 = []
61      phase_data11 = []
62      mag_data12 = []
63      phase_data12 = []
64      mag_data21 = []
65      phase_data21 = []
66      mag_data22 = []
67      phase_data22 = []
68
```

```
69      # Suck up the comments for use later
70      for i in range(comment_rows):
71      comment_lines += csv_file.readline()
72
73      #discard empty line
74      tmp = csv_file.readline()
75
76      #discard BEGIN line
77      tmp = csv_file.readline()
78
79      #Get header information
80      hdr_line = csv_file.readline()
81      hdr = hdr_line.split(',')
82
83      #Get frequency units
84      freq_unit = hdr[0].split('(')
85      freq_unit = freq_unit[1].split(')')
86      freq_unit = freq_unit[0]
87
88      #Get channel name and type
89      ch_info = hdr[1]
90      ch_info = ch_info.split('(')
91      ch_name = ch_info[0]
92      ch_unit = ch_info[1].split(')')
93      ch_unit = ch_unit[0]
94
95      #Change case if not set properly
96      if ch_unit == 'DB':
97      ch_unit = 'dB'
98
99      #Read in data from CSV file
100     for i in range(data_lines):
101     line = csv_file.readline()
102     line = line.split(',')
103
104     freq_data.append(line[0])
105     mag_data11.append(line[1])
106     phase_data11.append(line[2])
107     mag_data12.append(line[3])
108     phase_data12.append(line[4])
109     mag_data21.append(line[5])
110     phase_data21.append(line[6])
111     mag_data22.append(line[7])
112     phase_data22.append(line[8])
113
114
115     csv_file.close()
116
117     out_file = open(basedir + out_name, 'w+')
118
119     #Output format line to write data
120     format_line = '# ' + freq_unit + ' S ' + ch_unit + ' R 50' + ...
            '\n'
121
122     #Include channel name
123     snp_comment = '!S2P File: ...
            Measurements:(justKiddingNotChannelname) ' + ch_name + '\n'
```

```
124     comment_lines += snp_comment
125     #can put moar shit here if gather moar channel names
126
127     #Write out comments and format information
128     out_file.writelines(comment_lines) #restore the comments
129     out_file.writelines(format_line) #print the format stuff ...
            with units/whatever
130
131     #Write data
132     for i in range(len(freq_data)):
133     line_out = freq_data[i] + ' ' + mag_data11[i] + ' ' + ...
            phase_data11[i] + ' ' + mag_data12[i] + ' ' + ...
            phase_data12[i] + ' ' + mag_data21[i] + ' ' + ...
            phase_data21[i]+ ' ' + mag_data22[i] + ' ' + ...
            phase_data22[i]# + '\n'
134     out_file.writelines(line_out)
135
136
137     out_file.close()
138
139     print('Operation completed successfully.')
```

The script outputs the *.s2p file ready to import into ADS!

1.3 De-embedding with MATLAB RF Toolbox

This was much easier, and the second (current) version was able to be done without a single *for* loop! Data is imported into "rfdata" objects through built-in functions designed for *.s2p files. No manual editing is necessary. Error box S-parameters are obtained as described in **??**.

S-parameters are extracted, and the error box and DUT S-parameters are fed into a de-embed function, part of the RF Toolbox. Conversion from S-parameters to ABCD is done inside the function, however the dB conversion only goes one way, so $20 * log10(x)$ must be applied to convert back to dB before plotting. This entire process was much more simple. Below is the code.

Listing 6: De-embedding with RF Toolbox

```
1     %ECE802Deembedder3
2     % IT works and is simple!
3
4     clear; close all;
5
6     Diode = read(rfdata.data, 'diode.s2p');
7     Resistor = read(rfdata.data, 'resistor.s2p');
8     Open = read(rfdata.data, 'open.s2p');
9     Short = read(rfdata.data, 'short.s2p');
10    Through = read(rfdata.data, 'through.s2p');
11
12    %DiodeS = sparameters(Diode);
13    [DiodeS, freq]  = extract(Diode, 'S_parameters');
14    ResistorS = extract(Resistor, 'S_parameters');
15    OpenS = extract(Open, 's_parameters');
```

```
16      ShortS = extract(Short, 's_parameters');
17      ThroughS = extract(Through, 's_parameters');
18
19      % Get Error box S param
20      ErrorS(1,1,:) = ...
            (ShortS(1,1,:)-2.*ThroughS(1,1,:)+OpenS(1,1,:))./ ...
21      (OpenS(1,1,:)-2.*ThroughS(1,2,:)-ShortS(1,1,:));
22      ErrorS(1,2,:) = sqrt(ThroughS(1,2,:).*(1-ErrorS(1,1,:).^2));
23      ErrorS(2,1,:) = ErrorS(1,2,:);
24      ErrorS(2,2,:) = ErrorS(1,1,:);
25
26      ResistorDemS = deembedsparams(ResistorS, ErrorS, ErrorS);
27      DiodeDemS = deembedsparams(DiodeS, ErrorS, ErrorS);
28
29      % Plot stuff
30      % Note that, for some reason matlab is not psychic and ...
            assumes your data
31      % are not power measurements so it does 20log10 instead of ...
            10log10.  Or
32      % maybe Saran wrap was wrong and it should be 20log10. Dunno.
33      plottr(freq, mag2db(squeeze(abs(ResistorS(1,1,:)))), ...
            'Frequency (Hz)', '|S11|', 'Resistor S11')
34      hold on
35      plot(freq, mag2db(squeeze(abs(ResistorDemS(1,1,:)))),  'g')
36      legend('Original', 'Deembedded')
37
38      plottr(freq, mag2db(squeeze(abs(ResistorS(1,2,:)))), ...
            'Frequency (Hz)', '|S12|', 'Resistor S12')
39      hold on
40      plot(freq, mag2db(squeeze(abs(ResistorDemS(1,2,:)))),  'g')
41      legend('Original', 'Deembedded')
42
43      plottr(freq, mag2db(squeeze(abs(ResistorS(2,1,:)))), ...
            'Frequency (Hz)', '|S21|', 'Resistor S21')
44      hold on
45      plot(freq, mag2db(squeeze(abs(ResistorDemS(2,1,:)))),  'g')
46      legend('Original', 'Deembedded')
47
48      plottr(freq, mag2db(squeeze(abs(ResistorS(2,2,:)))), ...
            'Frequency (Hz)', '|S22|', 'Resistor S22')
49      hold on
50      plot(freq, mag2db(squeeze(abs(ResistorDemS(2,2,:)))),  'g')
51      legend('Original', 'Deembedded')
52
53
54      plottr(freq, mag2db(squeeze(abs(DiodeS(1,1,:)))), 'Frequency ...
            (Hz)', '|S11|', 'Diode S11')
55      hold on
56      plot(freq, mag2db(squeeze(abs(DiodeDemS(1,1,:)))),  'g')
57      legend('Original', 'Deembedded')
58
59      plottr(freq, mag2db(squeeze(abs(DiodeS(1,2,:)))), 'Frequency ...
            (Hz)', '|S12|', 'Diode S12')
60      hold on
61      plot(freq, mag2db(squeeze(abs(DiodeDemS(1,2,:)))),  'g')
62      legend('Original', 'Deembedded')
63
```

```
64      plottr(freq, mag2db(squeeze(abs(DiodeS(2,1,:)))), 'Frequency ...
            (Hz)', '|S21|', 'Diode S21')
65      hold on
66      plot(freq, mag2db(squeeze(abs(DiodeDemS(2,1,:)))),  'g')
67      legend('Original', 'Deembedded')
68
69      plottr(freq, mag2db(squeeze(abs(DiodeS(2,2,:)))), 'Frequency ...
            (Hz)', '|S22|', 'Diode S22')
70      hold on
71      plot(freq, mag2db(squeeze(abs(DiodeDemS(2,2,:)))),  'g')
72      legend('Original', 'Deembedded')
73
74      %write(ResistorDemS, 'demres.s2p')
75
76      % Export to Excel, save as .csv, then use Python to convert ...
            to .s2p
77      filename = 'ResistorSpara.xlsx';
78      xlswrite(filename,[freq,mag2db(squeeze(abs(ResistorDemS(1,1,:)))), ...
            squeeze(angle(ResistorDemS(1,1,:))), ...
79      mag2db(squeeze(abs(ResistorDemS(1,2,:)))), ...
            squeeze(angle(ResistorDemS(1,2,:))), ...
            mag2db(squeeze(abs(ResistorDemS(2,1,:)))), ...
80      squeeze(angle(ResistorDemS(2,1,:))), ...
            mag2db(squeeze(abs(ResistorDemS(2,2,:)))), ...
            squeeze(angle(ResistorDemS(2,2,:)))]);
81
82      filename = 'DiodeSpara.xlsx';
83      xlswrite(filename,[freq,mag2db(squeeze(abs(DiodeDemS(1,1,:)))), ...
            squeeze(angle(DiodeDemS(1,1,:))), ...
84      mag2db(squeeze(abs(DiodeDemS(1,2,:)))), ...
            squeeze(angle(DiodeDemS(1,2,:))), ...
            mag2db(squeeze(abs(DiodeDemS(2,1,:)))), ...
85      squeeze(angle(DiodeDemS(2,1,:))), ...
            mag2db(squeeze(abs(DiodeDemS(2,2,:)))), ...
            squeeze(angle(DiodeDemS(2,2,:)))]);
```