CHAPTER

6

COMPUTER-AIDED ANALYSIS OF LINEAR CIRCUITS

6.1 Y-MATRIX APPROACH

While S-parameters are the usual parameters used to describe most RF and microwave circuit measurements, it is particularly convenient for computer-aided linear analysis to use the admittance, Y, matrix representation for the linear analysis. This chapter will describe a general formalism which is the foundation of most linear computer-adied-design (CAD) systems. In the discussion to follow circuits will be described in terms of circuit elements with associated parameter values and a nodal "net list" which shows how the elements are connected to form the circuit. An example of a generic representation of a net list is shown in Figure 6.1 together with a schematic of the circuit. For modern CAD systems the circuit description can usually be entered in a graphics representation of the schematic. In this case the software first converts the graphical representation from the schematic circuit into a net list representation of the circuit. For purposes of this chapter the net list will be considered as the starting point for describing the circuit.

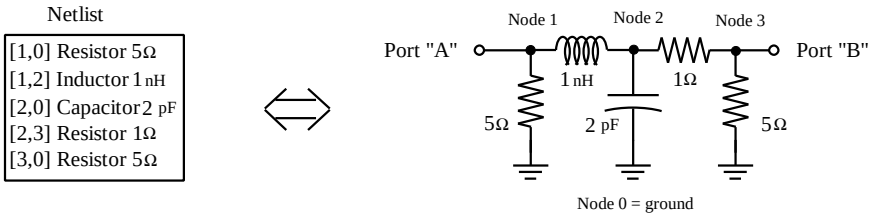


Figure 6.1 An example of a net list representation and it equivalent schematic of a circuit

For the circuit illustrated in Figure 6.1 three nodes are identified. A *node is defined as a junction between two or more elements.* Two ports are also identified by the letters "A" and "B." *A port signifies a circuit node that can be connect to the outside world*. For example a source might be connected to port "A" designating node "1" and a load to port "B" designating node "3." In general the node numbers do not have to be sequential as long as they uniquely represent each junction of the circuit. This means, also, that the node list order can be shuffled, i.e., there is no formal preferred order when listing the elements. Of course the analyst may prefer an order such as listing elements sequentially from left to right. Alternative net list constructions for the same circuit are shown in figure 6.2

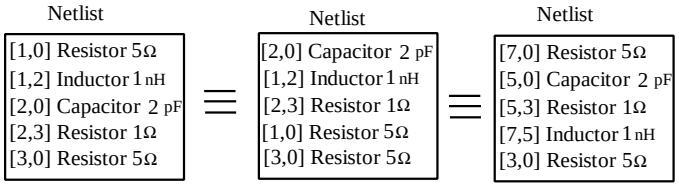


Figure 6.2 Different Net Lists representations of the same circuit

Note that in the example circuit of figure 6.3 node 2 is not designated as a port meaning that it is an *internal node* and will not be connected to the outside world. Nodes designated by ports are referred to as *external nodes* meaning that they can be connected to the outside world. The first step in analyzing the circuit is to consider that voltage sources are connected to each node and the resulting currents are measured.

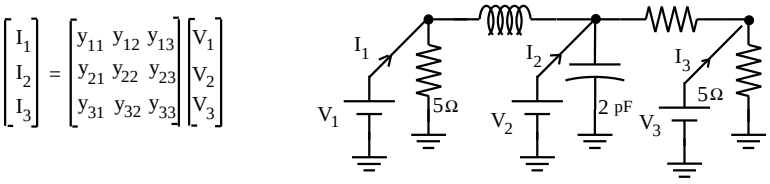
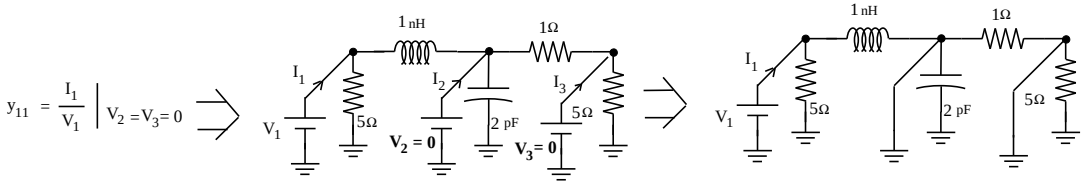


Figure 6.3 With voltages connected to every node the resulting current is described by the Y-matrix equation

The elements of the Y-matrix are equal to

$$y_{11} = \left.\frac{I_1}{V_1}\right|_{V_2=V_3=0} \qquad y_{12} = \left.\frac{I_1}{V_2}\right|_{V_1=V_3=0}, \ldots$$

From the circuit y_{11} is found by shorting nodes 2 and 3 to ground (making $V_2 = V_3 = 0$), applying a voltage to node 1 and observing the current into node 1. This is illustrated in figure 6.4



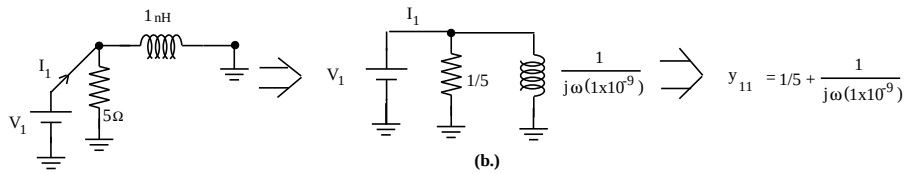


Figure 6.3 Determination of y_{11} by grounding all other nodes and observing the current to voltage ratio

In summary a driving point admittance, y_{11}, is the sum of all the element admittances connected to node 1. In a similar manner

$$y_{22} = jw(2\times10^{-12}) + \frac{1}{jw(1\times10^{-9})} + 1$$

since when nodes 1 and 3 are grounded the inductance, capacitance, and 1 Ω resistance are in parallel and the admittance is the sum of their individual element admittance values. Also,

$$y_{33} = 1 + \frac{1}{5}$$

is found when nodes 1 and 2 are grounded. This completes the determination of the diagonal terms of the circuits admittance matrix. The off diagonal terms are also easily found. For example the transfer admittance term y_{12} is found by grounding nodes 1 and 3 and applying a voltage at node 2 and observing the current I_1 which is the current in the grounded node. This is illustrated in figure 6.5

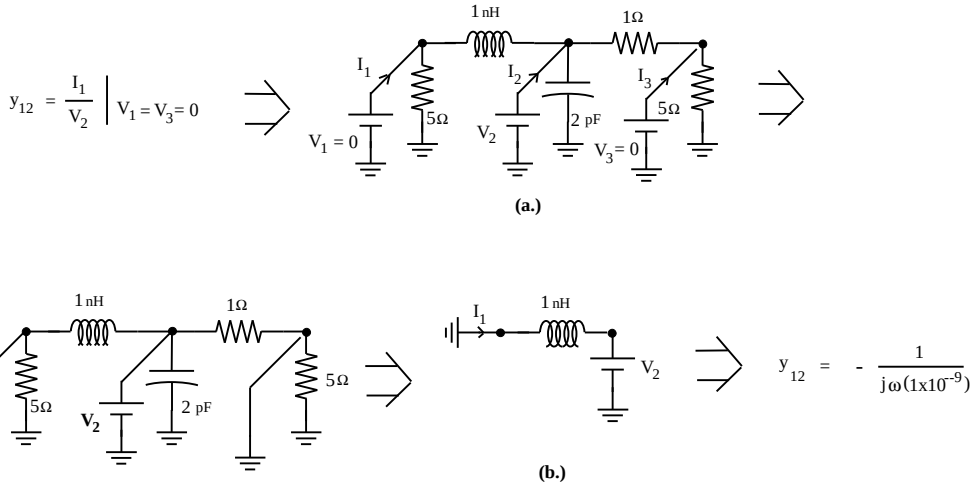


Figure 6.5 Illustration of y_{12} determination with nodes "1' and "3" grounded.

The results

$$y_{12} = -\frac{1}{jw(1\times10^{-9})}$$

is the sum of the admittances connected between nodes 1 and 2 with a negative sign. The negative sign is due to the assumed direction for currents, i.e., defined as being positive when going into the element. The transfer admittance term

$$y_{13} = 0$$

since there are no elements connecting nodes 1 and 3. The transfer admittance term

$$y_{23} = -1.$$

In a similar manner the three remaining terms can be found to be

$$y_{21} = -\frac{1}{jw(1\times10^{-9})},\ y_{31} = 0\text{, and } y_{32} = -1.$$

The Y-matrix for the above circuit is therefore

$$\mathbf{Y} = \begin{bmatrix} \frac{1}{5} + \frac{1}{jw(1\times10^{-9})} & -\frac{1}{jw(1\times10^{-9})} & 0 \\ -\frac{1}{jw(1\times10^{-9})} & jw(2\times10^{-12}) + \frac{1}{jw(1\times10^{-9})} + 1 & -1 \\ 0 & -1 & 1+\frac{1}{5} \end{bmatrix}$$

These terms could be written immediately by observing that all of the elements in the circuit are reciprocal and thus the Y-matrix is a symmetric matrix with off-diagonal terms being pairwise equal. Computation of the Y-matrix elements can be immediately written down by applying the following two rules:

The Diagonal Terms (y_{ii}) is the sum of element admittances connected to the node "i"

The Off-Diagonal Terms (y_{ij}) is the *negative* sum of element admittances connected between node "i" and node "j."

Assuming an operating frequency of 1 GHz permits one to determine the numerical values for the Y matrix, i.e.

$$\mathbf{Y} = \begin{bmatrix} .2 - j.0398 & j.0398 & 0 \\ j.0398 & 1 - j.0105 & -1 \\ 0 & -1 & 1.2 \end{bmatrix}$$

This characterizes the circuit assuming that sources and load combinations could be connected to any of the nodes. The circuit can be more specifically characterized since it is known that one node,"2," is an internal node and will have no current entering from the outside, $I_2 \equiv 0$. Knowing that the ports are connected to node 1 and node 3 it should be possible to characterize the circuit in terms of the port voltages and currents since that is the only interaction that the circuit will have with it environment. The achievement of this begins by expressing the nodal admittance matrix in terms of internal and external voltages and currents and imposing the requirement that internal nodes have an assoicated zero current. In the specific example this means that the current and voltage relationship using the Y matrix woud be

$$\begin{bmatrix} I_1 \\ 0 \\ I_3 \end{bmatrix} = \begin{bmatrix} .2 - j.0398 & j.0398 & 0 \\ j.0398 & 1 - j.0105 & -1 \\ 0 & -1 & 1.2 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}$$

Next the rows of the equations are rearranged so that the internal node current is at the bottom of the current column vector, Y matrix, and the internal voltage is at the bottom of the voltage column vector, i.e.,

$$\begin{bmatrix} I_1 \\ I_3 \\ 0 \end{bmatrix} = \begin{bmatrix} .2 - j.0398 & 0 & j.0398 \\ 0 & 1.2 & -1 \\ j.0398 & -1 & 1 - j.0105 \end{bmatrix} \begin{bmatrix} V_1 \\ V_3 \\ V_2 \end{bmatrix}$$

From the third equation (third row) one sees that

$$j.0398V_1 - V_3 + (1 - j.0105)V_2 = 0$$

or ,

$$V_2 = \frac{(-j.0398V_1 + V_3)}{(1 - j.0105)}$$

and

$$V_2 = (.0004 - j.0398)V_1 + (.9999 + j.0105)V_3$$

which can be substituted into the top two equations,

$$I_1 = (.2 - j..0398)V_1 + j.0398[(.0004 - j.0398)V_1 + (.9999 + j.0105)V_3]V_2$$
$$I_3 = 1.2V_3 - [(.0004 - j.0398)V_1 + (.9999 + j.0105)V_3]$$

$$I_1 = (.2016 - j.0398)V_1 + (.0004 + j.0398)V_3$$
$$I_3 = (.0004 + j.0398)V_1 + (.2001 - j.0105)V_3$$

$$\mathbf{Y}_{reduced} = \begin{pmatrix} .2016 - j.0398 & .0004 + j.0398 \\ .0004 + j.0398 & .2001 - j.0105 \end{pmatrix}$$

$$\mathbf{S}_{reduced} = \begin{pmatrix} -.8304 + j.0299 & .0036 - j.0309 \\ .0036 - j.0309 & -.8243 - j.0090 \end{pmatrix}$$

which agrees with the results in example 3.3.1 and example 3.4.1. In summary the technique involved the following steps: (1.) Form the nodal admittance matrix, Y, where driving point terms, y_{ii} is found by adding the admittances of all elements connected to the node, i, and where the transfer terms y_{ij} is found by the negative sum of the admittances of the elements connecting the node, i and j. (2.) the rows and columns of the Y matrix are rearranged so that internal nodes are located at the lower right of the matrix, (3.) the Y matrix is reduced by substituting the internal equations to reduce the Y matrix so that it includes only the external nodes, i.e., ports.

One additional genralization can be seen by considering the Y matrix for the inductor element. The element Y matrix is illustrated in figure 6.6.

$$\mathbf{Y}_{inductor} = \begin{bmatrix} \frac{1}{j\omega L} & -\frac{1}{j\omega L} \\ -\frac{1}{j\omega L} & \frac{1}{j\omega L} \end{bmatrix}$$

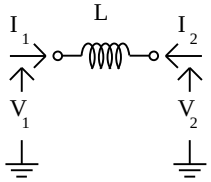


Figure 6.6 The matrix for an inductor.

Noting that the negative sign is part of the off diagonal term for the inductor admittance matrix means that the formation of the nodal Y matrix uses the element matrix components appropriate to the nodal matrix entry, i.e., when computing a driving point value then the driving point value from the element matrix is used and when computing a transfer value for the nodal matrix then the transfer component from the element matrix is used. The negative sign observed before is automatically taken care of. Therefore, the nodal Y matrix can be formed by successively adding the components of the element matrix to the appropriate node numbers. If an elements with its own matrix is connected between nodes i and j and has not yet been accounted for then it is added to the existing nodal matrix with y_{11} of the element being added to Y_{ii} of the nodal matrix, and y_{12} of the element being added to Y_{ij} of the nodal matrix, and so forth. This is summarized as follows.

$$\mathbf{Y}_{element} = \begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{pmatrix} \Rightarrow \mathbf{Y}_{nodal} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & Y_{ii} + y_{11} & \cdot & Y_{ij} + y_{12} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & Y_{ji} + y_{21} & \cdot & Y_{jj} + y_{22} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

Recognition of this facilitates a straight forward computer implimentation of a linear circuit simulator. A library of elements produces the element admittance elements. One accumulates these values in a nodal matrix according to above proceedure. Finally the nodal matrix is reduced to include only the nodes which are assoicated with ports.

$$\mathbf{I} = \begin{pmatrix} \mathbf{I_e} \\ \mathbf{I_i} \end{pmatrix} = \begin{pmatrix} \mathbf{Y_{ee}} & \mathbf{Y_{ei}} \\ \mathbf{Y_{ie}} & \mathbf{Y_{ii}} \end{pmatrix} \begin{pmatrix} \mathbf{V_e} \\ \mathbf{V_i} \end{pmatrix} = \mathbf{V}$$

$$\begin{pmatrix} \mathbf{I_e} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{Y_{ee}} & \mathbf{Y_{ei}} \\ \mathbf{Y_{ie}} & \mathbf{Y_{ii}} \end{pmatrix} \begin{pmatrix} \mathbf{V_e} \\ \mathbf{V_i} \end{pmatrix}$$

$$\mathbf{I_e} = \mathbf{Y_{ee}}\mathbf{V_e} + \mathbf{Y_{ei}}\mathbf{V_i}$$
$$\mathbf{0} = \mathbf{Y_{ie}}\mathbf{V_e} + \mathbf{Y_{ii}}\mathbf{V_i}$$

$$\mathbf{V_i} = -\mathbf{Y}_{ii}^{-1}\mathbf{Y_{ie}}\mathbf{V_e}$$

$$\mathbf{I_e} = \left(\mathbf{Y_{ee}} - \mathbf{Y_{ei}}\mathbf{Y}_{ii}^{-1}\mathbf{Y_{ie}}\right)\mathbf{V_e}$$

$$\boxed{\mathbf{Y}_{reduced} = \mathbf{Y_{ee}} - \mathbf{Y_{ei}}\mathbf{Y}_{ii}^{-1}\mathbf{Y_{ie}}}$$

6.2 LINEAR CAD EXAMPLE

To reinforce the principles of the previous section a relatively generic linear simulator is illustrated using the matrix based language MATLAB. The approach will consist of using a text file, called a *design file*, to enter net-list information about the circuit. Circuit elements are represented by key words or their abreviations in the design file. As each line of the netlist is read the elements activate a function to up date a node Y matrix which will accumulate admittance values. In the net list the analyst can use arbitrary integers to designate nodes not necessarily in sequential order. Node "0" is the required designator for ground. An example of a deisgn file structure is shown in Figure 6.7 .

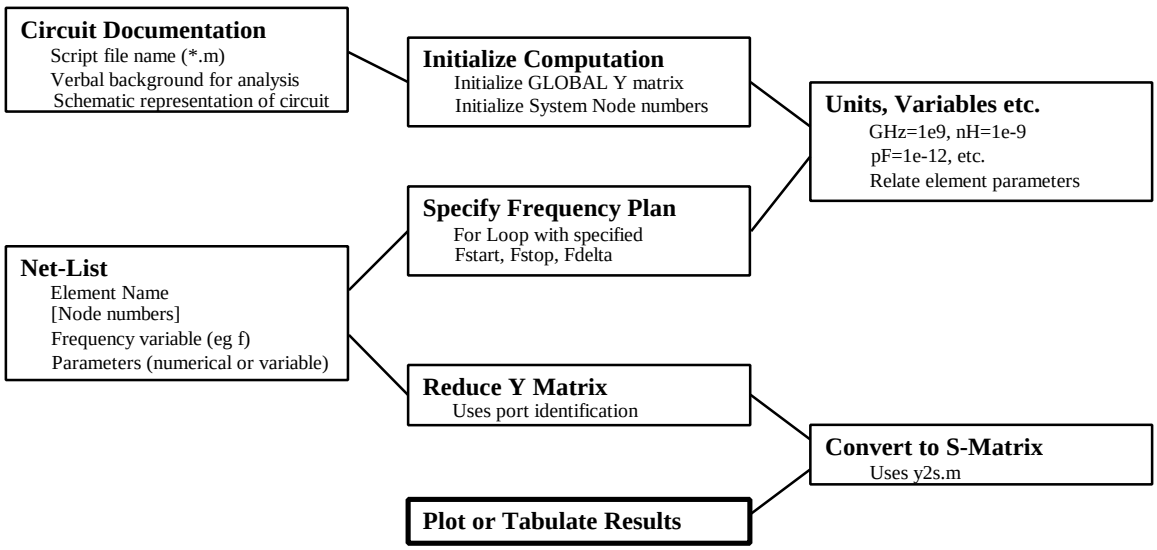


Figure 6.7 Script File Steps

```
%RESCKT1.M
%Name: M.L.Edwards
%Purpose:  Illustrate MATLAB Linear Simulation with Resistive Network
%
%%                 -------------- R=11 --------------
%                 |                                  |
%                 |                                  |
%    Port A >--- 3 ---- R=Rs --- (-2) ---- R=Rs --- 1 ---< Port B
%                 |                 |                |
%              R=7.7              R=4.9            R=6.6
%                 |                 |                |
%                Gnd               Gnd              Gnd
%                (0)               (0)              (0)
%%                      [Rs=5.2]
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

Initlize   %clears all variables from previous anaylsis
%Variables
Rs=5.2;
%Units
GHz=1e9;

%Frequency Sweep
Fstart=1*GHz; Fstop=5*GHz; Fdelta=1*G
for f=Fstart : Fdelta : Fstop

%Netlist
  Res([3,0],7.7);    %Node order irrevalent for reciprocal 2-port elements
  Res([3,-2],Rs);    %negative integers ok & valuable for parameter
  Res([3,1],11);
  Res([0,-2],4.9);   %ground"0" can be first node
  Res([1,-2],Rs);
  Res([1,0],6.6);

Yout=Reduce([3,1]);         %Reduces Y matrix to two port with A=3 & B=1
                            %and clear Global Y and NodeNum matrices
Sout=Y2S(Yout,1/50);        %Creates Scattering Matrix of S-parameters

Ytable=mktable(Ytable, f/GHz, Yout);
Stable=mktable(Stable, f/GHz, Sout);

end                         %Repeat Analysis for next frequency

%Print\Plot results
Ytable   %prints table since no ";" at end of line
Stable
```

Figure 6.8 An example script file for analyzing a linear circuit described by a net-list

The functions called in the script file in figure 6.9 are described next.

```
%script file initlize.m
%clears local and global variables
clear; clear global; clear all
```

```
function Y=Res(Nodes,Resist)
%Function to update global Y matrix with a resistor
%
Nodechk(Nodes,2,1); %Error Check, Nodechk(Array of Nodes, # of nodes, zerosok=1)
Yres=[1,-1;-1,1]/Resist;   %Element Model:  Y matrix for resistor
global YGLOBAL;            %UpdateGlobal Ymatrix
Sysnodes=Nodenum(Nodes);   %Convert to SysNodes
YGLOBAL=UpdateY(Yres,YGLOBAL,Sysnodes);   %End of function
```

```
function Y=Nodechk(N, k,zerosok)
%File to check node number entries: nodechk.m
% Nodechk(Nodes,number of nodes, zerosok) where zerosok=1 if true

if N==[], error('***Nodes missing***'),end
if length(N) ~= k, error(['***Number nodes should equal ',num2str(k)],' ***'), end

if zerosok,
   if find(N~=0)==[], error('***No non-zero nodes***'), end
 else
   if nnz(N)~=length(N), error('***Gnd, "0," not allowed as node number***'), end
end;

nonzero=N(find(N~=0));
T=(nonzero'*ones(1,length(nonzero))==ones(length(nonzero),1)*nonzero);
if any(sum(T)>1), error('***Repeated non-zero nodes not allowed***'), end;
```

```
function sysnodes=Nodenum(n)
global GLOBALNODES;GN=GLOBALNODES;
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
%T=Transfer matrix between n and GN==>n repeated columns
%compared with GN repeated as rows.  No available match causes T to
%have a zero row determining GNnew which is added to update GN
%Transfer matrix recalculated with update GN to get sysnodes
% ========================================================
if GN==[]; GN=n(find(n~=0)); end ;                    %(1)
T=n'*ones(1,length(GN)+1)==ones(length(n),1)*[0,GN];  %(2)
GN=[GN,n(all(T'==0))] ;                               %(3)
T=n'*ones(1,length(GN)+1)==ones(length(n),1)*[0,GN] ; %(4)
sysnodes=[0:length(GN)]*T';                           %(5)
GLOBALNODES=GN;    % end of function                  %(6)
%========================================================
%Footnotes:
%-----------
%(1)  At first call of function Globalnode is empty
% Therefore, Non zero input nodes start off list
%
%Next several notes illustrated by example
%     assume n=[2 0 4] and GN=[1 4 3 5] at start of function

%(2)  n'*ones(1,length(GN)+1)
%    ( 2 )              ( 2     2     2     2     2 )
%    ( 0 )*(1 1 1 1 1) = ( 0     0     0     0     0 )
%    ( 4 )              ( 4     4     4     4     4 )
%
%    ones(length(n),1)*[0,GN]
%    (1)              (0     1     4     3     5 )
%    (1)*(0 1 4 3 5) = (0     1     4     3     5 )
%    (1)              (0     1     4     3     5 )
%


%               function sysnodes=Nodenum(n) Cont'd
```

```
%         (0  0  0  0  0) <==detects that 2 not in list
%     T = (1  0  0  0  0)
%         (0  0  1  0  0)
%
%(3)
%     all(T'==0)   =>  [1 0 0]
%     n(all(T'==0))  n( [1 0 0]) = 2
%     [GN,n(all(T'==0))] => [[1 4 3 5],[2]] = [1 4 3 5 2]
%
%(4)
%    ( 2   2   2   2   2   2)    (0   1   4   3   5   2)
%    ( 0   0   0   0   0   0) == (0   1   4   3   5   2)
%    ( 4   4   4   4   4   4)    (0   1   4   3   5   2)
%
%         (0  0  0  0  0  1) <==2 detected in update GN list
%     T = (1  0  0  0  0  0)
%         (0  0  1  0  0  0)
%
%(5)  sysnodes = [0 1 2 3 4 5]*transpose(T) = [5 0  2]
%(6)  GLOBALNODES = [1 4 3 5 2]
```

```
function Ynew=UpdateY(Yelem,Yold,Nodes)
%% Ynew=UpdateY(Yelem,Yold,Nodes)
%function to update global Y matrix, Yold, with values
%from the element Y matrix--works for any size(Yelem)
%   Note: lenght(Nodes) = size(Yelem)
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Ynew=Yold;       % Initialize Ynew for update
%Increase size of Yold if new nodes being added
if max(Nodes)>max(size(Yold)); Yold=Add(Yold,zeros(max(Nodes))); end
NonZero=Nodes(Nodes~=0);
ElemNode=find(Nodes~=0);
Ynew(NonZero,NonZero)=Yold(NonZero,NonZero)+Yelem(ElemNode,ElemNode);
```

```
function Yreduce=reduce(nodes)
%====================check nodes for errors ========================
global GLOBALNODES;GN=GLOBALNODES;
if GN==[]; error('*** No element nodes in circuit ***'), end ;
if nodes==[]; error('*** No ports in reduce statement ***'), end ;
T=nodes'*ones(1,length(GN)+1)==ones(length(nodes),1)*[0,GN];
if any(T(:,1)); error(['*** Gnd, "0," not allowed as ',...
   'port in "reduce" statement ***']), end
if ~all(any(T')); error(['*** Node number(s) " ',...
    num2str(nodes(find(~any(T')))),...
    ' " not in circuit netlist***']), end
T(:,1)=[]; Duplicate=(sum(T)>1);
if any(Duplicate); error(['*** Mulitple node numbers "',...
    num2str(GN(Duplicate)),'" in "reduce" statement ***']),end
%=====================================================================
N=Nodenum(nodes);  %N=system nodes
%function to reduce global Y-matrix to an N-port Y-matrix where N<=number of nodes
%Need to move rows and columns so that external nodes at upper left and internal nodes
%lower right.  At completion clears Global Y matrix and Global Node Counter
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
%Example Y=Reduce([2,3])
%Note:
%   (y11 y12 y13)     (y22 y23 y21)
%   (y21 y22 y23)  => (y32 y33 y31)
%   (y31 y32 y33)     (y12 y13 y11)
%       [Old]            [New]
%         Shuffle Function  Shifts row and column so internal node lower left
%%then
%         (y22 y23)
%    Yee= (y32 y33)  and  Yii=(y11)
%
%                            (y21)
%    Yie=(y12 y13)   and    Yei=(y31)
%
%    Yreduce=Yee-Yei*inv(Yii)*Yie
%               function Yreduce=reduce(nodes) Cont'd
```

```
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
global YGLOBAL
Ynew=schuffle(YGLOBAL,N);  % routine shuffle matrix

%numports=length(N)
sz=size(YGLOBAL);
Yee=Ynew(1:length(N),1:length(N));
Yii=Ynew(length(N)+1:sz,length(N)+1:sz);
Yei=Ynew(1:length(N),length(N)+1:sz);
Yie=Ynew(length(N)+1:sz,1:length(N));
Yreduce=add(Yee,-Yei*inv(Yii)*Yie); %using add( , ) takes care of Yii=[] case
YGLOBAL=[];    %Clear YGLOBAL for additional circuit calulations
GLOBALNODES=[];  %Clears node counter(relates user nodes to program nodes)
```

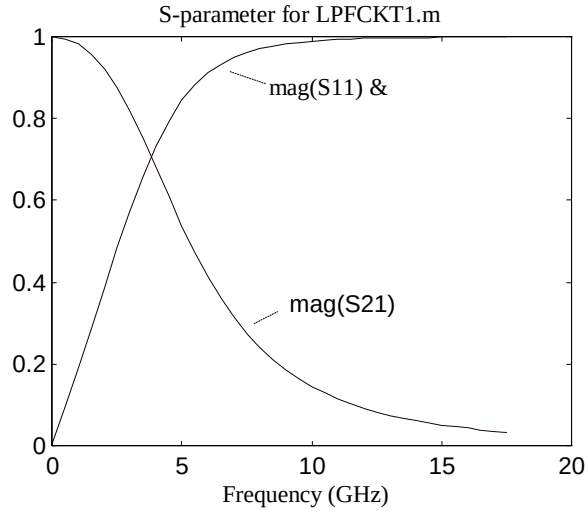
```
function Yout=schuff(Yin,N)
%functions schuffles rows and columns of Matrix Yin so that rows and columns contained
%in vector N are moved to upper left
%-----------------------------------------------------------
Seq=[1:size(Yin)] ; % Seq=sequence & Ncomp=deleted Sequence [Seq=N + Ncomp]
Ncomp=Seq(all( [N,0]'*ones(1,length(Seq)) ~= ones(length(N)+1,1)*Seq));
%"0" added so that "all" logic will also work for length(N)=1 case
Yout(Seq,Seq)=Yin([N,Ncomp],[N,Ncomp]);    %end function
```

```
function S=Y2S(Y,Yo)
%converts Y matrix to Scattering Matrix

% - - -Identity matrix with dimension of Y - - -
I=eye(size(Y));
S=(Yo*I-Y)/(Yo*I+Y); %Matrix division==mult by inverse
```

```
function NewTable=MkTable(OldTable, F, NewData)

% This function adds a row to the bottom of the OldTable.
% The new row consists of the frequency F as the first element,
% and the elements in NewData taken row by row.

% Make N the transpose of the new matrix NewData
N=NewData';    % This transpose conjugate all complex numbers.

% Take the transpose of the column form of N
O=N(:)';       % This second transpose conjugate back all complex numbers.

% Add variable F to the beginning of new row of data
Newline=[F, O];

% Add new row to the oldtable
NewTable=[OldTable; Newline];
```

6.3 CIRCUIT SCRIPT FILES AND RESULTS

```
%RLCCKT3.M
%Name: M.L.Edwards
%Purpose:  Illustrate MATLAB Linear Simulation for RLC Networkin Chap 5.3
%
%
%    Port A >--- 1 --- L=1   ---- 2 ----  L=1   --- 3 ---< Port B
%                                  |
%                                  |
%                                C=2
%                                  |
%                                  |
%                                Gnd
%                                (0)
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Initlize   % clears all variables from previous anaylsis
%Variables (Optional)

%Units
GHz=1e9;
nH=1e-9;
pF=1e-12;

%Stimulation (single frequency or sweep)
fstart=.01*GHz; fdelta=.5*GHz; fstop=18*GHz;
for f=fstart:fdelta:fstop

%Netlist

  IND([1,2],1*nH,f);
  IND([2,3],1*nH,f);
  CAP([2,0],2*pF,f);

Yout=Reduce([1,3]);        %Reduces Y matrix to two port with A=1 & B=3
Sout=Y2S(Yout,1/50);       %Creates Scattering Matrix of S-parameters
Ytable=mktable(Ytable, f/GHz, Yout);
Stable=mktable(Stable, f/GHz, Sout);

end

%Ytable;
%Stable;

plot(Stable(:,1),abs(Stable(:,2)),'r',Stable(:,1),abs(Stable(:,4)),'g')
xlabel('Frequency (GHz)');
ylabel('mag(S11) & mag(S21)');
title('S-parameter for LPFCKT1.m')
```

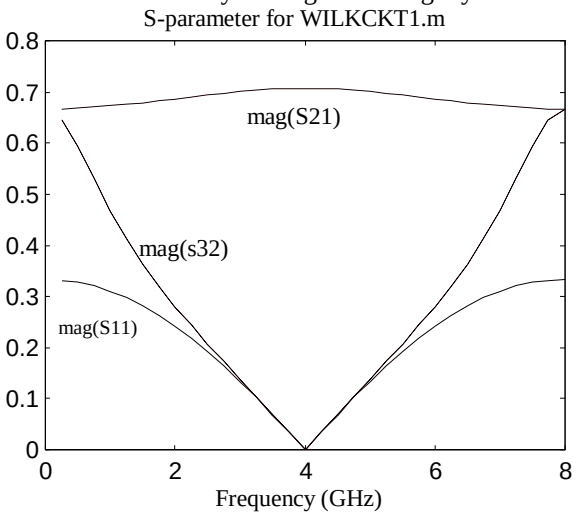


```
%Wilkckt.M
%Name: M.L.Edwards
%Purpose:  Illustrate MATLAB Linear Simulation with Wilkenson Coupler
%
%                     Z=70.7
%                     E=90deg,
%                     f=4GHz
%                     ------
%                  --| Tlin |---(2)-----< Port B
%                 |   ------     |
%                 |              |
%      Port A >--(1)            R=100
%                 |              |
%                 |   ------     |
%                  --| Tlin |---(3)-----< PortC
%                     ------
%                     Z=70.7,
%                     E=90deg,
%                     f=4GHz
%
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Initlize   %clears all variables from previous anaylsis
%Variables (Optional)
%Units
GHz=1e9;Deg=pi/180;

Fstart=.25; Fstop=8; Fdelta=.25;
%Frequency Sweep
for f=Fstart : Fdelta : Fstop,

%Netlist
Tlin([1,2],70.7,90*Deg,4*GHz,f*GHz);
Tlin([1,3],70.7,90*Deg,4*GHz,f*GHz);
RES([2,3],100);

Yout=Reduce([1,2,3]);   %Reduces Y matrix to two port with A=3 & B=2
Sout=Y2S(Yout,1/50);    %Creates Scattering Matrix of S-parameters

Ytable=mktable(Ytable, f, Yout);
Stable=mktable(Stable, f, Sout);
end

plot(Stable(:,1),abs(Stable(:,2)),'r',Stable(:,1),abs(Stable(:,4)),'g')
hold on
plot(Stable(:,1),abs(Stable(:,7)),'m')
xlabel('Frequency (GHz)');ylabel('mag(S11,S21,S32)');
title('S-parameter for WILKCKT1.m')
```

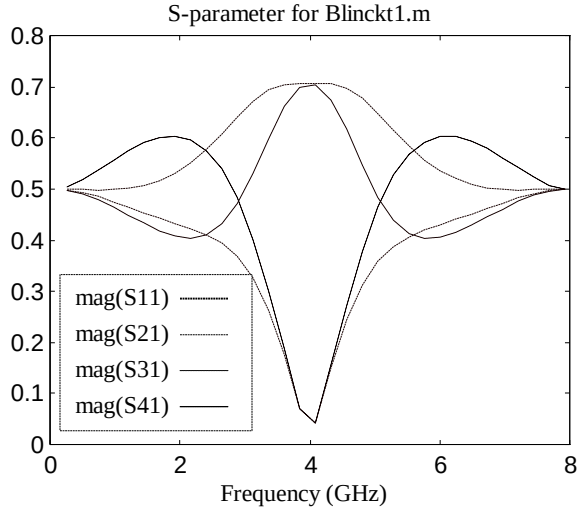
Warning: Matrix is close to singular or badly scaled.

Results may be inaccurate. RCOND = 4.328983e-017

[Due to 8 GHz since lines $\lambda/2$. Can eliminate by making fdleta=slightly different than .25, say 24



```
%Blinckt1.M
%Name: M.L.Edwards
%Purpose:  Illustrate MATLAB Linear Simulation with Branchline Coupler
%                     Z=35.35
%                     E=90deg,
%                     f=4GHz
%                     ------
%                (4) --| Tlin |--(3)-----< Port B
%                 |   ------      |
%     Z=50        ----             ----
%  E=90 @ 4GHz  |Tlin|           |Tlin|  Z=50,E=90 @ 4 GHz
%                ----             ----
%                 |   ------      |
%      Port A >--(1)--| Tlin |---(2)-----< PortC
%                     ------
%                     Z=35.35,
%                     E=90deg,
%                     f=4GHz
% * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Initlize   %clears all variables from previous anaylsis
%Variables (Optional)
%Units
GHz=1e9;
Deg=pi/180;

Fstart=.25; Fstop=8; Fdelta=.24;

%Frequency Sweep
for f=Fstart : Fdelta : Fstop,

%Netlist
Tlin([1,2],35.35,90*Deg,4*GHz,f*GHz);
Tlin([2,3],50,90*Deg,4*GHz,f*GHz);
Tlin([3,4],35.35,90*Deg,4*GHz,f*GHz);
Tlin([1,4],50,90*Deg,4*GHz,f*GHz);

Yout=Reduce([1,2,3,4]);   %Reduces Y matrix to two port with A=3 & B=2
Sout=Y2S(Yout,1/50);    %Creates Scattering Matrix of S-parameters

Ytable=mktable(Ytable, f, Yout);
Stable=mktable(Stable, f, Sout);

end

plot(Stable(:,1),abs(Stable(:,2)),'r',Stable(:,1),abs(Stable(:,3)),'g')
hold on
plot(Stable(:,1),abs(Stable(:,4)),'m',Stable(:,1),abs(Stable(:,5)),'b')
xlabel('Frequency (GHz)');
ylabel('mag(S11,S21,S31,S41)');
title('S-parameter for Blinckt1.m')
```



6.4 APPENDIX: ELEMENT LIBRARY

```
function Y=Cap(Nodes,C,f)
%Function to update global Y matrix with
%an capacitor
%
% - -Error Check, Bell, and Message Routine Here-
Nodechk(Nodes,2,1); % Nodechk(Array of Nodes, # of nodes, zerosok=1)
%
% - - - - - - Element Model - - - - - - - - - - -

Ycap=[1,-1;-1,1]*(i*2*pi*f*C);  %Y matrix for inductor
                                 %cross terms neg because of
                               %current direction
%
% - - - - - - Convert to SysNodes - - - - - - - -

% - - - -UpdateGlobal Ymatrix- - - - - - - - - - -
global YGLOBAL;
Sysnodes=Nodenum(Nodes);
YGLOBAL=UpdateY(Ycap,YGLOBAL,Sysnodes);
```

```
function Y=Ind(Nodes,L,f)
%Function to update global Y matrix with
%an inductor
%
% - -Error Check, Bell, and Message Routine Here-
Nodechk(Nodes,2,1); % Nodechk(Array of Nodes, # of nodes, zerosok=1)
%
% - - - - - - Element Model - - - - - - - - - - -

Yind=[1,-1;-1,1]/(2*pi*f*L*i);  %Y matrix for inductor
                          %cross terms neg because of
                         %current direction
%
% - - - - - - Convert to SysNodes - - - - - - - -

% - - - -UpdateGlobal Ymatrix- - - - - - - - - - -
global YGLOBAL;
Sysnodes=Nodenum(Nodes);
YGLOBAL=UpdateY(Yind,YGLOBAL,Sysnodes);
```

```
function Y=PRLC(Nodes,R,L,C,f)
%Function to update global Y matrix with
%a parallel resistor, inductor, and capacitor
%
% - -Error Check, Bell, and Message Routine Here-
Nodechk(Nodes,2,1); % Nodechk(Array of Nodes, # of nodes, zerosok=1)
%
% - - - - - - Element Model - - - - - - - - - - -

Yprlc=[1,-1;-1,1]*(1/R+1/(i*2*pi*f*L)+i*2*pi*f*C);


% - - - - - - Convert to SysNodes - - - - - - - -
Sysnodes=Nodenum(Nodes);

% - - - - UpdateGlobal Ymatrix- - - - - - - - - - -
global YGLOBAL;
YGLOBAL=UpdateY(Yprlc,YGLOBAL,Sysnodes);
```

```
function Y=Res(Nodes,Resist)
%Function to update global Y matrix with
%a resistor
%
% - -Error Check, Bell, and Message Routine Here-
Nodechk(Nodes,2,1); % Nodechk(Array of Nodes, # of nodes, zerosok=1)
%
% - - - - - - Element Model - - - - - - - - - - -

Yres=[1,-1;-1,1]/Resist;  %Y matrix for resistor
                          %cross terms neg because of
                         %current direction
%
% - - - - - - Convert to SysNodes - - - - - - - -

% - - - -UpdateGlobal Ymatrix- - - - - - - - - - -
global YGLOBAL;
Sysnodes=Nodenum(Nodes);
YGLOBAL=UpdateY(Yres,YGLOBAL,Sysnodes);
```

```
function Y=SRLC(Nodes,R,L,C,f)
%Function to update global Y matrix with
%a series resistor, inductor, and capacitor
%
% - -Error Check, Bell, and Message Routine Here-
Nodechk(Nodes,2,1); % Nodechk(Array of Nodes, # of nodes, zerosok=1)
%
% - - - - - - Element Model - - - - - - - - - - -

Z=R+i*2*pi*f*L+1/(i*2*pi*f*C);
Ysrlc=[1,-1;-1,1]/Z;


% - - - - - - Convert to SysNodes - - - - - - - -
Sysnodes=Nodenum(Nodes);

% - - - - UpdateGlobal Ymatrix- - - - - - - - - - -
global YGLOBAL;
YGLOBAL=UpdateY(Ysrlc,YGLOBAL,Sysnodes);
```

```
function Y=Tlin(Nodes,CharZ,ElecLength,RefFreq,f)
%Function Tlin(Nodes,CharZ,ElecLength,RefFreq,f)
%to update global Y matrix with
%an ideal transmission line
%
% - -Error Check, Bell, and Message Routine - - -
Nodechk(Nodes,2,1); % Nodechk(Array of Nodes, # of nodes, zerosok=1)
%
% - - - - - - Element Model - - - - - - - - - - -
%
%        Freq dependent Elec Length
Theta=(f/RefFreq)*ElecLength;
%
%         Y matrix for Tlin
Ytlin=i*[-cos(Theta),+1;+1,-cos(Theta)]/(CharZ*sin(Theta));
%
%
% - - - - - - Convert to SysNodes - - - - - - - -
Sysnodes=Nodenum(Nodes);

% - - - -UpdateGlobal Ymatrix- - - - - - - - - - -
global YGLOBAL;
YGLOBAL=UpdateY(Ytlin,YGLOBAL,Sysnodes);
```

```
function Y=Tloc(Nodes,CharZ,ElecLength,RefFreq,f)
%Function Tloc(Nodes,CharZ,ElecLength,RefFreq,f)
%to update global Y matrix with
%an ideal open circuited transmission line
%
% - -Error Check, Bell, and Message Routine - - -
Nodechk(Nodes,2,1); % Nodechk(Array of Nodes, # of nodes, zerosok=1)
%
% - - - - - - Element Model - - - - - - - - - - -
%
%        Freq dependent Elec Length
Theta=(f/RefFreq)*ElecLength;
%
%         Y matrix for Tlin
if tan(Theta)>1e12,
  Ytloc=i*[1,-1;-1,1]*1e12;
else,
  Ytloc=i*[1,-1;-1,1]*tan(Theta)/CharZ;
end;
%
%
% - - - - - - Convert to SysNodes - - - - - - - -
Sysnodes=Nodenum(Nodes);

% - - - -UpdateGlobal Ymatrix- - - - - - - - - - -
global YGLOBAL;
YGLOBAL=UpdateY(Ytloc,YGLOBAL,Sysnodes);
```

```
function Y=Tlsc(Nodes,CharZ,ElecLength,RefFreq,f)
%Function Tlsc(Nodes,CharZ,ElecLength,RefFreq,f)
%to update global Y matrix with
%an ideal short circuited transmission line
%
% - -Error Check, Bell, and Message Routine - - -
Nodechk(Nodes,2,1); % Nodechk(Array of Nodes, # of nodes, zerosok=1)
%
% - - - - - - Element Model - - - - - - - - - - -
%
%        Freq dependent Elec Length
Theta=(f/RefFreq)*ElecLength;
%
% - - - - - - Y matrix for Tlin - - - - - - - - -

if 0<=tan(Theta) & tan(Theta)<1e-12,
   Ytlsc=[1,-1;-1,1]/(i*1e-12*CharZ);
elseif  -1e-12<tan(Theta) & tan(Theta)<=0,
   Ytlsc=-[1,-1;-1,1]/(i*1e-12*CharZ);
else
   Ytlsc=[1,-1;-1,1]/(i*tan(Theta)*CharZ);
end


% - - - - - - Convert to SysNodes - - - - - - - -
Sysnodes=Nodenum(Nodes);

% - - - -UpdateGlobal Ymatrix- - - - - - - - - - -
global YGLOBAL;
YGLOBAL=UpdateY(Ytlsc,YGLOBAL,Sysnodes);
```