The 4th Symposium on
ADVANCED TOPICS IN ELECTRICAL ENGINEERING

ATEE - 2004

November 26, 2004, "Politehnica" University of Bucharest
Faculty of Electrical Engineering

Computer Aided Generation of DC & AC Electric Circuit Tests

Paul Dan Cristea, Rodica Ileana Tuduce, Aurora Rodica Tuduce
"Politehnica" University of Bucharest
Spl. Independentei 313, 77206 Bucharest, Romania,
Phone: +40 - 21- 411 44 37, Fax: +40 - 21- 410 44 14
e-mail:pcristea@dsp.pub.ro

PAPER OUTLINE

1. Problem set description
2. Tree generation
3. Cotree generation
4. Tree plot
5. Graph plot
6. Circuit parameters and variables generation
7. Converting voltage sources to curent sources
8. Introducing controlled sources
9. Web accessibility
10. Conclusions

OBJECTIVES

Design and develop a software able to automatically generate large sets of circuit analysis problems, all with the same general features, but having different topological structures and parameters of the circuits.

Conditions:

• The problems are for use both during the tutorials and for examinations, thus -- despite the inherent risk for an engineering perception of reality -- all parameters and variables describing. the circuits should be integers to facilitate the computational task.

• Problems and solutions should be stored automatically on disk in distinct directories.

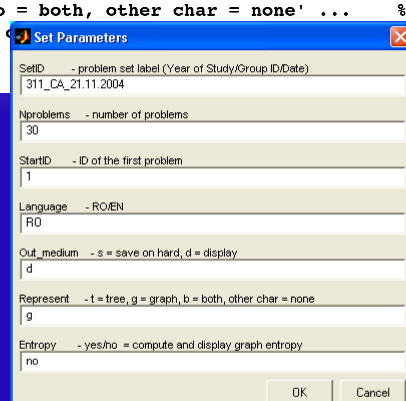
• Files referring to the same problem (text, graphics, etc) will have related labels.

• The system will be developed for making it accessible on the web.

PROBLEM SET DESCRIPTION

Choosing the parameters of the set of AC problems to generate.

```
%                          1            2   3    4    5    6    7
param = query({'311_CA_21.11.2004', '30','1','RO', 'd', 'g', 'no'}, ...
  {
     'SetID          - problem set label (Year of Study/Group ID/Date)', ...   % 1
     'Nproblems      - number of problems', ...                                % 2
     'StartID        - ID of the first problem', ...                           % 3
     'Language       - RO/EN', ...                                             % 4
     'Out_medium     - s = save on hard, d = display' ...                      % 5
     'Represent      - t = tree, g = graph, b = both, other char = none' ...   % 6
     'Entropy        - yes/no  = compute and d                                 % 7
  }, ...
  'Set Parameters');
```



Choosing Variables & Independent Parameters

Circuit Variables & Independent Parameters

Nnodes - number of nodes
4

Nbranches - number of branches
7

```
%                 1    2    3    4    5     6     7     8     9    10    11    12    13
param = query({'4','7','4','4','1', '4', '4', '0', '1', '4', '4', '2', 'Y'}, ...
    {
    'Nnodes        - number of nodes', ...                                            % 1
        'Nbranches     - number of branches', ...                                     % 2
        'I_chord_a_max - maximum absolute value of chord current active components [A]', ...    % 3
        'I_chord_r_max - maximum absolute value of chord current reactive components [A]', ...  % 4
        'R_twig_min    - minimum value of twig resistences [Ohms]', ...               % 5
        'R_twig_max    - maximum value of twig resistences [Ohms]', ...               % 6
        'X_twig_max    - maximum absolute value of twig reactance [Ohms]', ...        % 7
        'E_twig_max    - maximum absolute value of twig Re & Im emf-s [V]', ...       % 8
        'R_chord_min   - minimum value of chord resistences [Ohms]', ...              % 9
        'R_chord_max   - maximum value of chord resistences [Ohms]', ...              %10
        'X_chord_max   - maximum absolute value of chord reactance [Ohms]', ...       %11
        'nJ            - number of branches with current sources', ...                %12
        'CrossLinks    - Y/N - mutual inductances and controlled sources'...          %13
    }, ...
    'Circuit Variables & Independent Parameters');
```

R_chord_max - maximum value of chord resistences [Ohms]
4

X_chord_max - maximum absolute value of chord reactance [Ohms]
4

nJ - number of branches with current sources
2

CrossLinks - Y/N - mutual inductances and controlled sources
Y

OK Cancel

5

Mutual inductive couplings and controlled source parameters

```
if strcmp(lower(CrossLinks), 'y')
    %                  1    2    3    4    5    6    7    8
    param = query({'0', '0', '0', '0', '0', '3', '0', '3
        {
        'nEI - number of current controlled voltage sour                    %1
            'nJU - number of voltage controlled current                     %2
            'nEU - number of voltage controlled voltage                     %3
            'nJI - number of current controlled current                     %4
            'nM  - number of mutual inductive couplings'                    %5
            ['Zta_max - maximum absolute value of transf
                '          Ea + j.Er = (Zta + j.Ztr) (Ia                    %6
            'Ztr_max - maximum absolute value of transfe                    %7
            ['Yta_max - maximum absolute value of transf
                '          Ja + j.Jr = (Yta + j.Ytr) (Ua                    %8
            'Ytr_max - maximum absolute value of transfe                    %9
            ['Aa_max  - maximum absolute value of voltag
                '          Ea + j.Er = (Aa + j.Ar) (Ua +                    %10
            'Ar_max  - maximum absolute value of voltage                    %11
            ['Ba_max  - maximum absolute value of curren
                '          Ja + j.Jr = (Ba + j.Br) (Ia +                    %12
            'Br_max  - maximum absolute value of current                    %13
            'XM_max  - maximum value of mutual inductive                    %14
        }, ...
        'Selection of mutual inductive couplings and cor
```

Selection of mutual inductive couplings and controlle...

nEI - number of current controlled voltage sources E = Zt * I
0

nJU - number of voltage controlled current sources J = Yt * U
0

nEU - number of voltage controlled voltage sources E = A * U
0

nJI - number of current controlled current sources J = B * I
0

nM - number of mutual inductive couplings
0

Zta_max - maximum absolute value of transfer resistance [Ohms]
Ea + j.Er = (Zta + j.Ztr) (Ia + j.Ir)
3

Ztr_max - maximum absolute value of transfer reactance [Ohms]
0

Yta_max - maximum absolute value of transfer conductance [Siemens]
Ja + j.Jr = (Yta + j.Ytr) (Ua + j.Ur)
3

Ytr_max - maximum absolute value of transfer susceptance [Siemens]
0

Aa_max - maximum absolute value of voltage gain active component
Ea + j.Er = (Aa + j.Ar) (Ua + j.Ur)
5

Ar_max - maximum absolute value of voltage gain reactive component
0

Ba_max - maximum absolute value of current gain active component
Ja + j.Jr = (Ba + j.Br) (Ia + j.Ir)
5

Br_max - maximum absolute value of current gain reactive component
0

XM_max - maximum value of mutual inductive reactance [Ohms]
4

OK Cancel

6

CIRCUIT TOPOLOGY

C_nodes_twigs = GenerateTree(Ntwigs, mode)

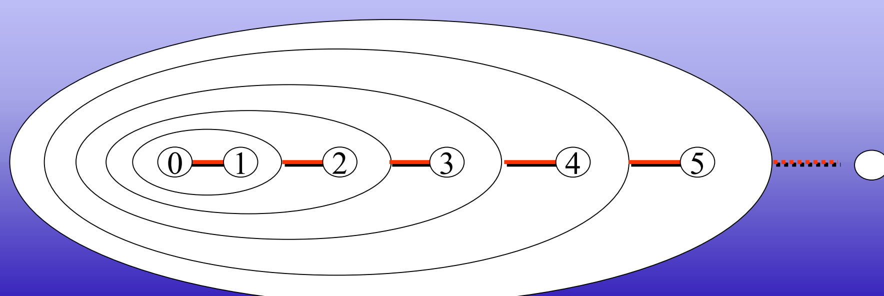
ShowTree(C_nodes_twigs, SetID, k)

C_nodes_chords = GenerateCoTree(C_nodes_twigs, Nchords)

ShowGraphNet(C_nodes_twigs, C_nodes_chords, SetID, k)

C_twigs_chords = EssIncid(C_nodes_twigs, C_nodes_chords)

Tree Generation




```
function C_nodes_twigs = GenerateTree(n, mode)
C_nodes_twigs = zeros(n, n);
rand('state',sum(100*clock));
r = rand(2,n);
c = 2 * ( r(2, :) >= 0.5 ) - 1;
m = 0;
for k = 1:n
   s = ceil( (k-m)* r(1, k) + m-1 );
   f = k;
   if s>0, C_nodes_twigs(s, k) = c(k); end
   C_nodes_twigs(f, k) = - c(k);
   if mode == 's', m = s; end
end
```

Examples

C_nodes_twigs =

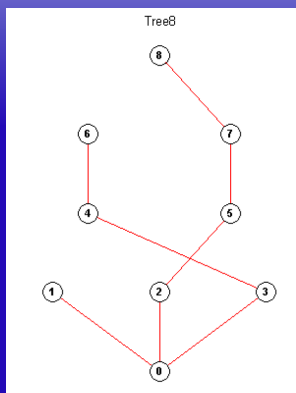
-1	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0
0	0	-1	-1	0	0	0	0
0	0	0	1	0	-1	0	0
0	0	0	0	-1	0	1	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	-1	1
0	0	0	0	0	0	0	-1

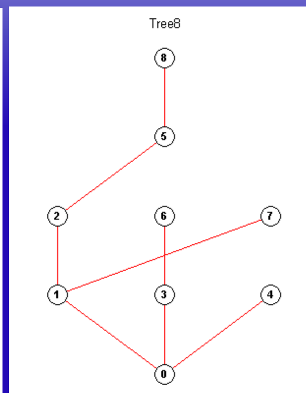
C_nodes_twigs =

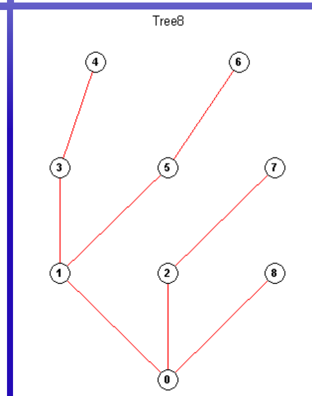
1	-1	0	0	0	0	-1	0
0	1	0	0	-1	0	0	0
0	0	1	0	0	-1	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	-1
0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1

C_nodes_twigs =

-1	0	1	0	1	0	0	0
0	-1	0	0	0	0	1	0
0	0	-1	1	0	0	0	0
0	0	0	-1	0	0	0	0
0	0	0	0	-1	1	0	0
0	0	0	0	0	-1	0	0
0	0	0	0	0	0	-1	0
0	0	0	0	0	0	0	-1





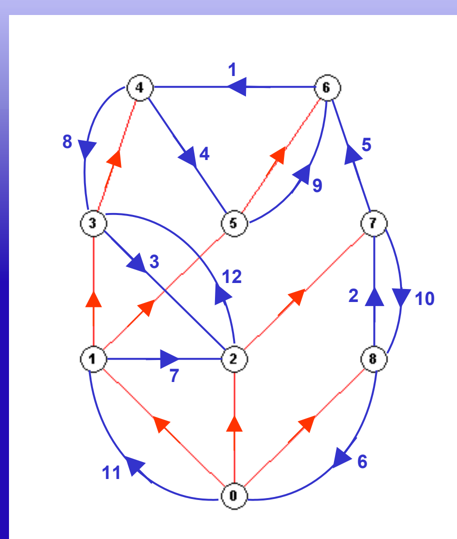


Cotree Generation

Starts from the chosen tree

Chords are introduced between nodes chosen randomly from the class of nodes with the lowest rank (lowest number of connected branches).

This order assures the best connectivity of the circuit for a given number of chords.



Code

```
function C_nodes_chords = GenerateCoTree(C_nodes_twigs, Nchords)

[n,m] = size(C_nodes_twigs);

   if Nchords <= 0,  C_nodes_chords = [ ]; return, end

   C_nodes_chords_ext = zeros(n+1, Nchords);

   rand('state', sum(100*clock)); r = rand(2,Nchords); c = 2 * ( r(2, :) >= 0.5 ) - 1;

   C_nodes_twigs_ext = CompleteCnb(C_nodes_twigs);

   Rank_All_Nodes = sum(abs((C_nodes_twigs_ext)'));

   Chord = 1;

   while Chord <= Nchords

      [Sorted_Rank_All_Nodes, Node_Line] = sort(Rank_All_Nodes);
      Start_Node = Node_Line(1);

       End_Node = Node_Line(2);

      C_nodes_chords_ext(Start_Node, Chord) = c(Chord);
      Rank_All_Nodes(Start_Node) = Rank_All_Nodes(Start_Node) + 1;

      C_nodes_chords_ext(End_Node, Chord) = - c(Chord);
      Rank_All_Nodes(End_Node) = Rank_All_Nodes(End_Node) + 1;

      Chord = Chord + 1;
   end
   C_nodes_chords = C_nodes_chords_ext(2:(n+1), :);
```

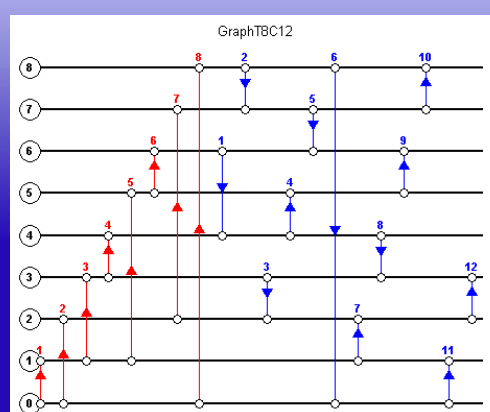

Examples

C_nodes_twigs =

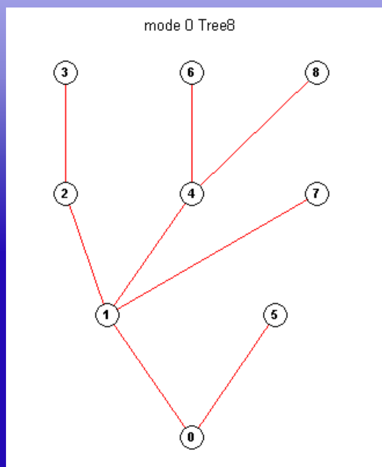
-1	0	1	0	1	0	0	0
0	-1	0	0	0	0	1	0
0	0	-1	1	0	0	0	0
0	0	0	-1	0	0	0	0
0	0	0	0	-1	1	0	0
0	0	0	0	0	-1	0	0
0	0	0	0	0	0	-1	0
0	0	0	0	0	0	0	-1

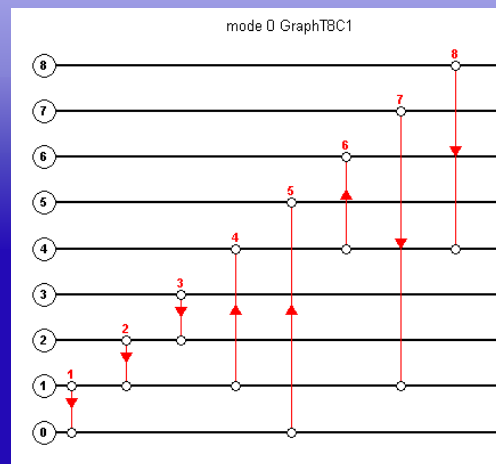
C_nodes_chords =

0	0	0	0	0	0	1	0	0	0	-1	0
0	0	-1	0	0	0	-1	0	0	0	0	1
0	0	1	0	0	0	0	-1	0	0	0	-1
-1	0	0	1	0	0	0	1	0	0	0	0
0	0	0	-1	0	0	0	0	1	0	0	0
1	0	0	0	-1	0	0	0	-1	0	0	0
0	-1	0	0	1	0	0	0	0	1	0	0
0	1	0	0	0	1	0	0	0	-1	0	0

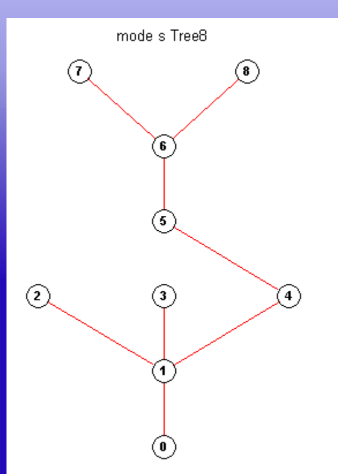


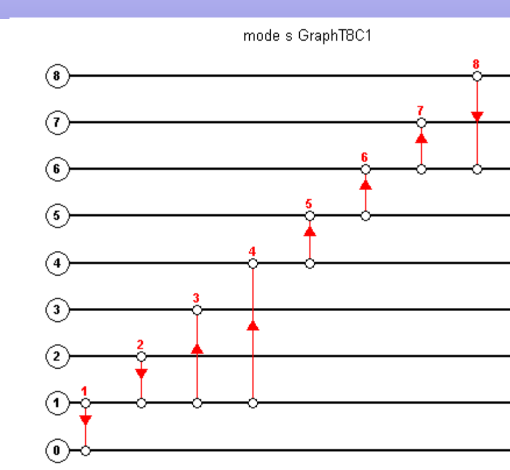
Tree Plot





‘s’ Mode





Code 1

```
function ShowTree(C_nodes_twigs, SetID, k);
% Plots  the tree corresponding to the C_nodes_twigs nodes-twigs incidence matrix
[n,m] = size(C_nodes_twigs);

FigName = 'Tree'; FigType = 'png';
f = sprintf('%s.%s', FigName, FigType);

y = ones(1, n+1);
y(0+1) = 0;
numY = zeros(1, n+1);
numX = zeros(1, n+1);

UAC = [abs(sum(C_nodes_twigs)); abs(C_nodes_twigs) - eye(n)];
[ii, jj] = find(UAC);  % ii(i)-1 - index of node from where starts twig i, ending in node jj(i) = i
               % the direction  might be inversed
for i = 1:n
   y(i+1) = y(ii(i)) + 1;     % y(i+1) ordinate of node i
End

ymax = max(y);

for i = 0:ymax
  numY(i+1) = sum(y == i);
  d = 1/(numY(i+1) + 1);
  kk = find(y == i);
  for j = 1:numY(i+1), x(kk(j)) = d * j; end
end
hFig = figure;
set(hFig, 'tag', strrep(FigName, '_', ' '));
axis([0, 1, 0, ymax+0.25]);
hold on
```


Code 2

```
for i = 1:n
   plot([x(ii(i)), x(i+1)], [y(ii(i)), y(i+1)], '-r');
end


for i = 0:n
   plot(x(i+1), y(i+1), 'ok',...
            'MarkerEdgeColor', [0,0,0], ...
            'MarkerFaceColor', [1,1,1],...
            'MarkerSize',15);

   text('string', num2str(i),'position', [x(i+1), y(i+1)], 'FontSize', 8, ...
       'FontWeight','bold', 'Color', [0, 0, 0], ...
       'HorizontalAlignment', 'Center', 'VerticalAlignment', 'Middle');
end

ha = findobj(hFig, 'type', 'axes');
set(ha, 'Visible', 'off');
ht = title(strrep(FigName, '_', ' '));
set(ht, 'Color', [0 0 0], 'FontAngle', 'normal', 'FontName', 'Helvetica', ...
                                    'FontSize', [10], 'FontUnits', 'points', 'FontWeight', 'normal', ...
                                    'HorizontalAlignment', 'center', 'VerticalAlignment', 'bottom', 'Visible', 'on');
set(hFig, 'name', f);
f = makeUnique(f);
print( hFig, ['-d' FigType], f);
if ~isempty(k), close(hFig); end

return
```

Graph Plot

function ShowGraphNet (C_nodes_twigs, C_nodes_chords, …
SetID, k, ChordsNumbers)

C_nodes_twigs - the tree branches (twigs) to nodes incidence matrix

C_nodes_chords - the co-tree branches (chords) to nodes incidence matrix

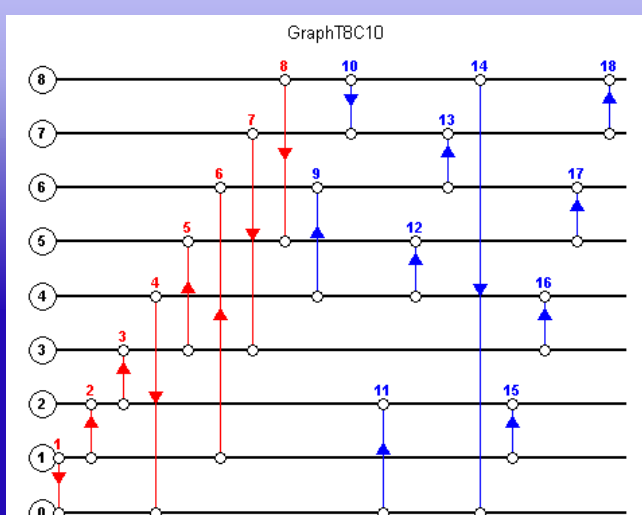
SetID - label (tag) to identify the set of problems

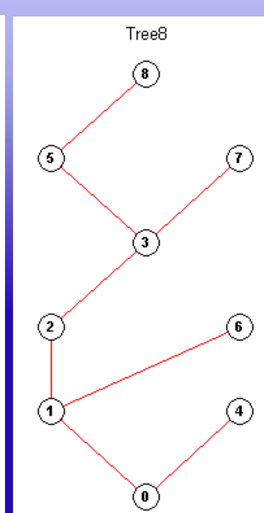
k - numerical identifier of the problem

ChordsNumbers = 0 - chords numbered independently, from 1 to Nchords

ChordsNumbers = 1 - chords numbered in sequence with twigs, from Ntwigs+1 to Ntwigs + Nchords

Examples





CIRCUIT PARAMETERS & VARIABLES

Circuit parameter and variable generation

```
[I, U, Z, E] = CircParam_AC(C_twigs_chords, …
      I_chord_a_max, I_chord_r_max, ...
      R_twig_min,  R_twig_max,  X_twig_max, E_twig_max, ...
      R_chord_min, R_chord_max, X_chord_max);
```

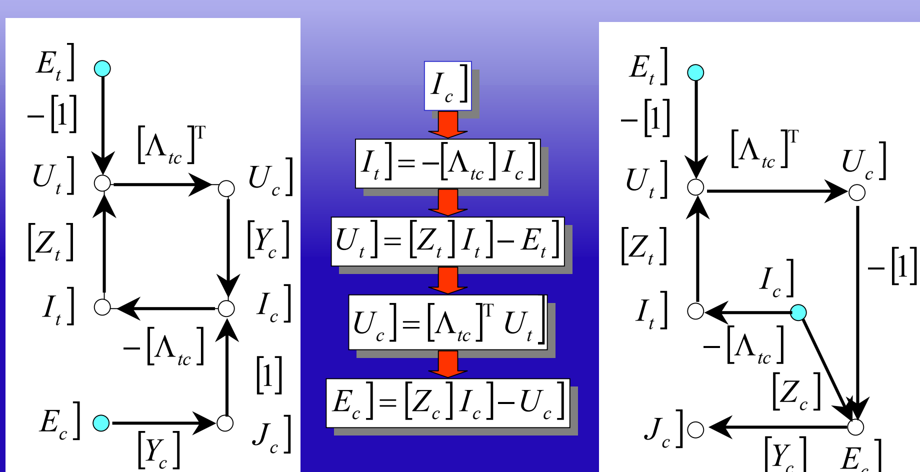
Converting (some) voltage sources to current sources

```
[E, J] = ConvertE2J_AC(E, Z, nJ);
```

Introducing controlled sources

```
[E, J, Zt, Yt, A, B, XM] = ControlledSources_AC(E, J, I, U, Z, ...
        nControl, nEI, nJU, nEU, nJI, nM, ...
        Zta_max, Ztr_max, Yta_max, Ytr_max, Aa_max, Ar_max, …
        Ba_max, Br_max, XM_max);
```


Circuit Parameter and Variable Generation



Global Circuit Variables

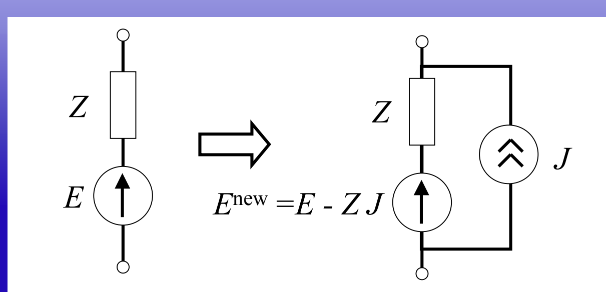
Concatenate the matrices for tree & cotree

$$[U] = \begin{bmatrix} U_t \\ U_c \end{bmatrix} \qquad [I] = \begin{bmatrix} I_t \\ I_c \end{bmatrix} \qquad [E] = \begin{bmatrix} E_t \\ E_c \end{bmatrix} \qquad [J] = \begin{bmatrix} J_t \\ J_c \end{bmatrix}$$

Converting Voltage Sources to Current Sources

Current sources (change of independent voltage source emf's)

$$[E^{\text{new}}] = [E] - [Z][J]$$



Convert nJ voltage sources to current sources

[E, J] = ConvertE2J_AC(E, Z, nJ);

Cross Parameters

[E, J, Zt, Yt, A, B, XM] = ControlledSources_AC(E, J, I, U, Z, ...
nControl, nEI, nJU, nEU, nJI, nM, ...
Zta_max, Ztr_max, Yta_max, Ytr_max, …
Aa_max, Ar_max, Ba_max, Br_max, XM_max);

Controlled sources (change of independent voltage source emf's)

$$\left[E^{controlled}\right] = \left[Z_t\right]\left[I\right] \qquad \left[E^{new}\right] = \left[E\right] - \left[Z_t\right]\left[I\right]$$

$$\left[J^{controlled}\right] = \left[Y_t\right]\left[U\right] \qquad \left[E^{new}\right] = \left[E\right] - \left[Z\right]\left[Y_t\right]\left[U\right]$$

$$\left[E^{controlled}\right] = \left[A\right]\left[U\right] \qquad \left[E^{new}\right] = \left[E\right] - \left[A\right]\left[U\right]$$

$$\left[J^{controlled}\right] = \left[B\right]\left[I\right] \qquad \left[E^{new}\right] = \left[E\right] - \left[Z\right]\left[B\right]\left[I\right]$$

Mutual reactances

$$\left[E^{induced}\right] = \left[-\,j\,X_M\right]\left[I\right] \qquad \left[E^{new}\right] = \left[E\right] - \left[E^{induced}\right]$$

WEB ACCESSIBILITY

The system will be accessible on the INTERNET, to allow remote use, for both professors and students

Partial examination of problems will be done on the computer, In a face-to-face or remote setting.

The web accessibility is currently partially functional and partially under development

```
function retstr = webCAM(instruct, outfile)
%   webCAM returns circuit parameters into HTML table.
%   webCAM(INSTRUCT) returnes output in RETSTR.
%   webCAM(INSTRUCT, OUTFILE) returns output in RETSTR.
%   OUTFILE is a valid spec  for test output.
%
%   INSTRUCT is a structure created by the matweb program.
%   It contains fields corresponding to the HTML form fields
%   in the HTML form, webCAM1.html.  In webCAM1.html there
%   is a hidden field, called INSTRUCT.MLMFILE that references
%   this webCAM.m M-file.

function rs = webCircuitPlot(h)
% RS = webCircuitPlot(H) creates a plot of graph with handle h and
% returns HTML output in string RS.  Handle h is the
% structure created by matweb.  It contains variables
% from the HTML input form in Circuit_plot_generator.html
```


CONCLUSIONS

• A specialized e-learning system able to automatically generate large sets of circuit analysis problems, all with the same difficulty, but having different topological structures and parameters of the Circuits, has been designed, implemented and experimented.

• The problems are for use both during the tutorials and for examinations, thus -- despite the inherent risk for an engineer understanding of reality -- all parameters and variables describing the circuits should be integers to facilitate the computational task.

• Problems and solutions should be stored automatically on disk in distinct directories, with files referring to the same problem having related labels

• The system will be developed for making it accessible on the web