Lecture23

Sung-Min Hong (smhong@gist.ac.kr)

Semiconductor Device Simulation Lab.

School of Electrical Engineering and Computer Science

Gwangju Institute of Science and Technology

GIST Lecture on December 10, 2018

Today's goal

- Read a netlist file and solve the DC problem.

```
r1 in out 100
r2 out 0 200
vin in 0 1
```

Connectivity

- In principle, our problem is not very difficult. Application of KCL.
  - Key point: We should understand the connectivity of our elements.
  - Therefore, a table for the node names is required.
  - In the case of our example, we want to have `0 in out.`
  - Instead of its name, a number is assigned.
- Then, our netlist can be written as

```
r1 2 3 100
r2 3 1 200
vin 2 1 1
```

Read a netlist file

- Two elements
  - Resistor: Name starting with `r`
  - Voltage source: Name starting with `v`
  - Otherwise, an error occurs.
  - Again, a number is assigned to the element type (1 for voltage sources and 2 for resistors)
- Then, our netlist can be written as

```
2 2 3 100
2 3 1 200
1 2 1 1
```

Implementation (1)

- Open the input file.

```
inputFileName = 'test.sp';

fileID = fopen(inputFileName,'r');
if (fileID==-1)
    error(['I cannot open the file.']);
end

elementTable = [];
nodeTable = { '0' };
```

Implementation (2)

- Read each line. Update `nodeTable`.

```
while ~feof(fileID)
    line = fgetl(fileID);
    C = strsplit(line);
    elementName = C{1};
    node1 = C{2};
    node2 = C{3};
    value = str2num(C{4});

    ind1 = find(strcmp(nodeTable,node1));
    if (isempty(ind1))
        nodeTable{size(nodeTable,1)+1,1} = node1;
    end
    ind2 = find(strcmp(nodeTable,node2));
    if (isempty(ind2))
        nodeTable{size(nodeTable,1)+1,1} = node2;
    end
```

Implementation (3)

- Read each line. Update nodeTable.

```
ind1 = find(strcmp(nodeTable,node1));
ind2 = find(strcmp(nodeTable,node2));
switch elementName(1)
    case 'v'
        elementTable = cat(1,elementTable,[1 ind1 ind2 value]);
    case 'r'
        elementTable = cat(1,elementTable,[2 ind1 ind2 value]);
    otherwise
        error(['Only v and r are supported.']);
end
end
```

Our example

- When it is processed, we have the following results.

```
>> SPICE
>> nodeTable

nodeTable =

    '0'
    'in'
    'out'

>> elementTable

elementTable =

     2     2     3   100
     2     3     1   200
     1     2     1     1
```

Unknown variables

- Four unknown variables for each element
  - Current for the terminal 1
  - Current for the terminal 2
  - Voltage for the terminal 1
  - Voltage for the terminal 2
- How many unknown variables?
  - (# of elements) times (4)
  - Anything else?
  - Additionally, (# of nodes) for the node voltages.
  - In the case of our example, we have 15 unknonwn variables.

Resistor

- Four equations for each resistor
  - Ohm's law

$$i1 - \frac{v1 - v2}{R} = 0$$

  - (Current for the terminal 1) + (Current for the terminal 2) = 0

$$i1 + i2 = 0$$

  - (Voltage for the terminal 1) – (Connected node voltage) = 0
  - (Voltage for the terminal 2) – (Connected node voltage) = 0
- Also the terminal current is added to the node equation.

Voltage source

- Four equations for each resistor
  - Voltage across the voltage source

$$v1 - v2 - vsource = 0$$

  - (Current for the terminal 1) + (Current for the terminal 2) = 0

$$i1 + i2 = 0$$

  - (Voltage for the terminal 1) – (Connected node voltage) = 0
  - (Voltage for the terminal 2) – (Connected node voltage) = 0
- Also the terminal current is added to the node equation.

Implementation (4)

- Prepare some quantities.

```
nElement = size(elementTable,1);
nodeOffset = nElement * 4;
nNode = size(nodeTable,1);
nTotal = nodeOffset+nNode;

Jaco = sparse(nTotal,nTotal);
res = zeros(nTotal,1);
sol = zeros(nTotal,1);
```

Implementation (5)

- Consider each element.

```
for ii = 1:nElement
    elementOffset = (ii-1)*4;
    type = elementTable(ii,1);
    ind1 = elementTable(ii,2);
    ind2 = elementTable(ii,3);
    value = elementTable(ii,4);
```

Implementation (6)

- Ohm’s law / voltage difference

```
 % Equation 1
    switch type
        case 1
            res(elementOffset+1) = sol(elementOffset+3) - sol(elementOffset+4) - value;
            Jaco(elementOffset+1,elementOffset+1:elementOffset+4) = [0 0 1 -1];
        case 2
            res(elementOffset+1) = sol(elementOffset+1) - (sol(elementOffset+3)-sol(elementOffset+4))/value;
            Jaco(elementOffset+1,elementOffset+1:elementOffset+4) = [1 0 -1/value 1/value];
        otherwise
            error(['Only v and r are supported.']);
    end
```

Implementation (7)

- Other equations

```
% Equation 2
res(elementOffset+2) = sol(elementOffset+1) + sol(elementOffset+2);
Jaco(elementOffset+2,elementOffset+1:elementOffset+4) = [1 1 0 0];
% Equation 3
res(elementOffset+3) = sol(elementOffset+3) - sol(nodeOffset+ind1);
Jaco(elementOffset+3,elementOffset+1:elementOffset+4) = [0 0 1 0];
Jaco(elementOffset+3,nodeOffset+ind1) = -1;
% Equation 4
res(elementOffset+4) = sol(elementOffset+4) - sol(nodeOffset+ind2);
Jaco(elementOffset+4,elementOffset+1:elementOffset+4) = [0 0 0 1];
Jaco(elementOffset+4,nodeOffset+ind2) = -1;
```

Implementation (8)

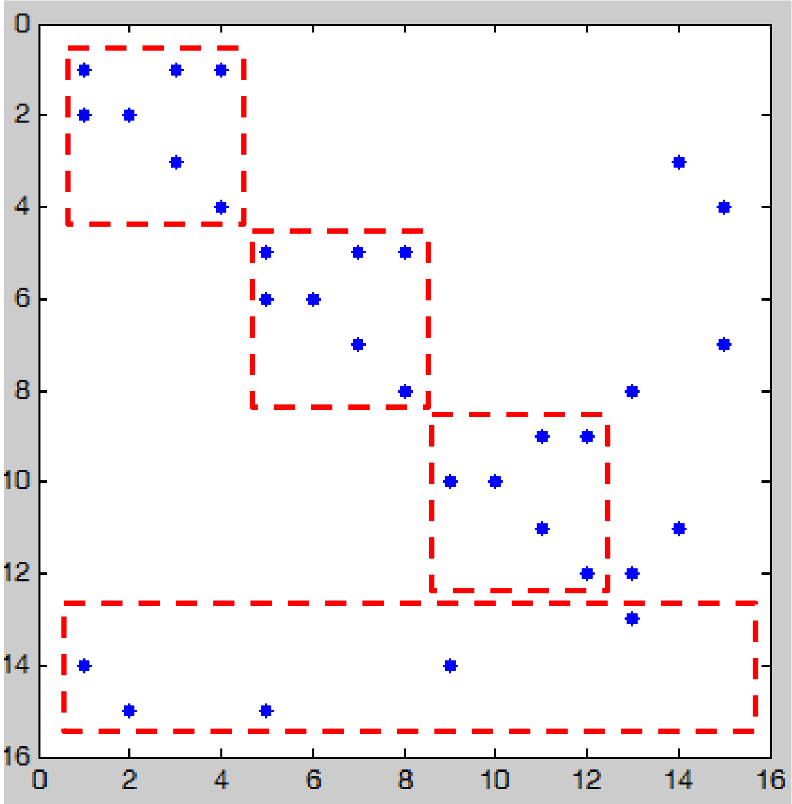
- KCL and all other stuffs

```
    % Node equation 1
    res(nodeOffset+ind1) = res(nodeOffset+ind1) + sol(elementOffset+1);
    Jaco(nodeOffset+ind1,elementOffset+1) = Jaco(nodeOffset+ind1,elementOffset+1) +
1;
    % Node equation 2
    res(nodeOffset+ind2) = res(nodeOffset+ind2) + sol(elementOffset+2);
    Jaco(nodeOffset+ind2,elementOffset+2) = Jaco(nodeOffset+ind2,elementOffset+2) +
1;
end
% Ground
res(nodeOffset+1) = sol(nodeOffset+1);
Jaco(nodeOffset+1,:) = 0;
Jaco(nodeOffset+1,nodeOffset+1) = 1;

sol = Jaco \ (-res);
```

Jacobian sparsity pattern

- We can read it.



Solution vector

- Check the solution.

```
>> sol

sol =

    0.0033
   -0.0033
    1.0000
    0.6667
    0.0033
   -0.0033
    0.6667
         0
   -0.0033
    0.0033
    1.0000
         0
         0
    1.0000
    0.6667
```