ECE 521 Lecture #2 Sept. 26, 2016

Note Title 9/26/2016

HW # 1 Due Oct. 12

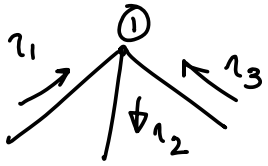
Formulation of circuit equations

- Physical Laws (KCL, KVL)
- Branch constitutive relationships (BCR) or equations (BCE)

$V = IR$ BCR for a resistor

KCL – The algebraic sum of currents leaving a node is zero

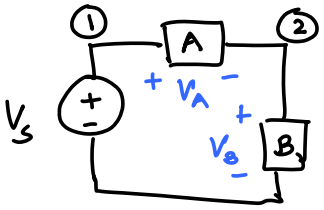
$-i_1 + i_2 - i_3 = 0$



KVL $v_A + v_B - v_S = 0$

relation between branch & node voltages

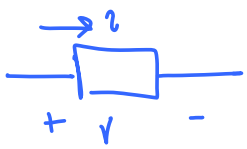
$v_A = v_1 - v_2$ $v_B = v_2$

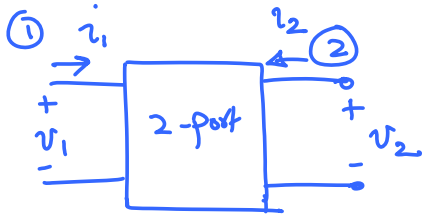


BCR or BCE

These are mathematical models of the circuit components. Relate current, voltage, charge or flux

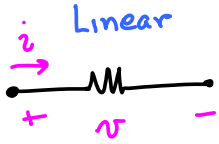
Reference direction

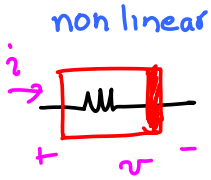




Ideal two-terminal elements

1) Resistor





current controlled $v = Ri$

voltage controlled $i = \frac{1}{R} v = Gv$

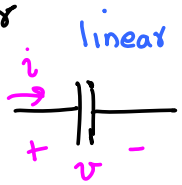
$v = v(i)$

$i = i(v)$

Diode: $i = I_s\left(e^{v/v_{th}} - 1\right)$

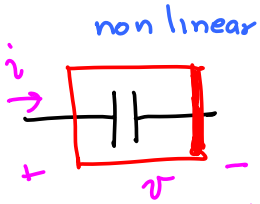
Can also have an implicit relation $f(i, v) = 0$

2) Capacitor

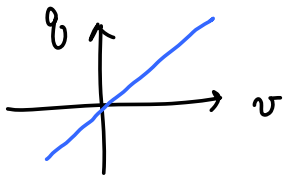


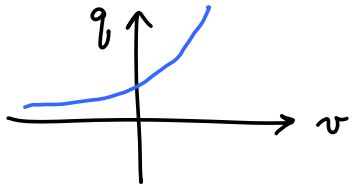
$q = Cv$

$i = \frac{dq}{dt} = C\frac{dv}{dt}$



$i = \frac{dq}{dt}$; $q = q(v)$





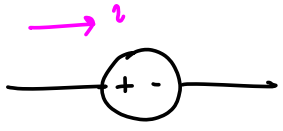
Implicit form: $f(q, v) = 0$

3) Inductors: Linear $v = L\frac{di}{dt}$ $\left(v = \frac{d\phi}{dt}\right)$

nonlinear: $v = \frac{d\phi}{dt}$ $\phi = \phi(i)$

4) Independent sources

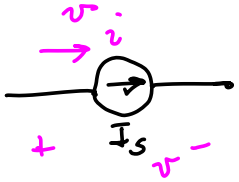
a) indep voltage source



$v = V_s$ (dc, ac, tran)

i = any value

b) indep current source

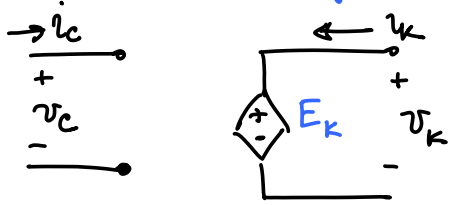


$i = I_s$ (dc, ac, tran)

5) Dependent (Controlled) sources

CCVS, CCCS, VCVS, VCCS

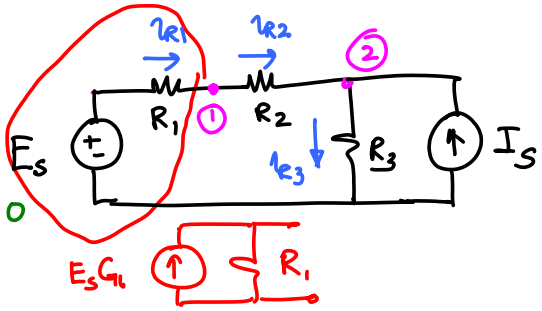
Voltage Controlled voltage source (VCVS)



$v_k = E_k v_c$; $i_c = 0$

i_k = any value

Nodal analysis



KCL@1: $-i_{R1} + i_{R2} = 0$

KCL@2: $-i_{R2} + i_{R3} - I_s = 0$

$i_{R1} = \frac{E_s - V_1}{R_1} = G_1 (E_s - V_1)$; $i_{R2} = \frac{V_1 - V_2}{R_2} = G_2 (V_1 - V_2)$; $i_{R3} = \frac{V_2}{R_3} = G_3 V_2$

Thus:

$$-G_1 (E_s - V_1) + G_2 (V_1 - V_2) = 0$$

$$-G_2 (V_1 - V_2) + G_3 V_2 - I_s = 0$$

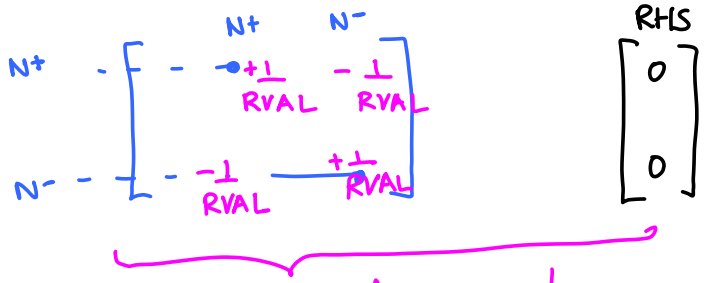
$$(G_1 + G_2) V_1 - G_2 V_2 = G_1 E_s$$

$$-G_2 V_1 + (G_2 + G_3) V_2 = I_s$$

G-matrix

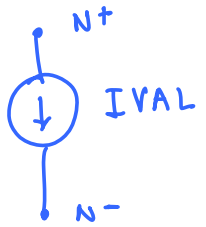
$$\begin{bmatrix} G_1 + G_2 & -G_2 \\ -G_2 & G_2 + G_3 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} G_1 E_s \\ I_s \end{bmatrix}$$

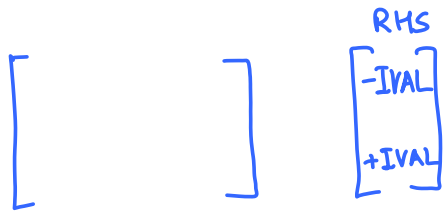
SPICE input: RK N+ N- RVAL



Stamp of a resistor

Indep Current Source : IK N+ N- IVAL (SPICE input)





What about an independent voltage source ?

Nodal analysis cannot incorporate a voltage source !

No floating or stacked voltage sources

Summary of nodal analysis

- Circuit equations can be assembled by inspection (stamp)
- In general G is sparse for most circuits (depends on the interconnection between nodes)
- G has non-zero diagonals an is diagonally dominant
- Cannot handle indep voltage sources
- Controlled sources $i = f(v)$

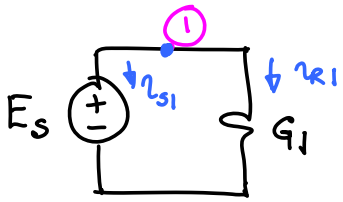
CCVS	x
VCVS	x
CCCS	x
VCCS	✓

Require voltage controlled elements!

BCR → $V_1 = E_S$

KCL → $i_{s1} + (i_{R1}) = 0$

$i_{R1} = G_1 V_1$

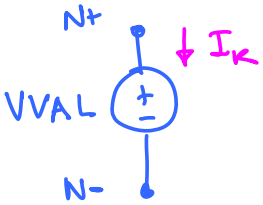


Modified Nodal Analysis (MNA)

$$\begin{bmatrix} 1 & 0 \\ G_1 & 1 \end{bmatrix} \begin{bmatrix} V_1 \\ i_{s1} \end{bmatrix} = \begin{bmatrix} E_S \\ 0 \end{bmatrix}$$

Floating voltage source

SPICE: VK N+ N- VVAL



	N+	N-	I_K	RHS
KCL@ N+			+1	
KCL@ N-			-1	
BCR	1	-1		VVAL

node N+ ------ $+1 \cdot I_K$

⋮

node N- ------ $-1 \cdot I_K$

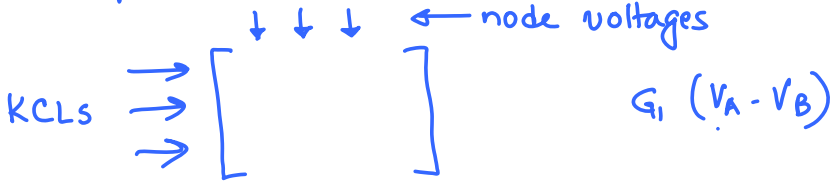
BCR (branch k) $V_{N+} - V_{N-} = VVAL$

- MNA matrix and RHS can be assembled directly from the input
- MNA can be applied to any circuit
- Some times there may be zeros on the main diagonal
- MNA matrix is "close" to the NA matrix
  It is NA + additional eqns & unknowns

Nodal analysis

Unknowns: node voltages

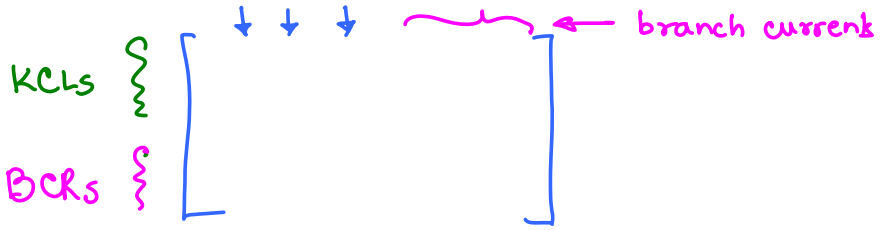
Equations: KCL at nodes

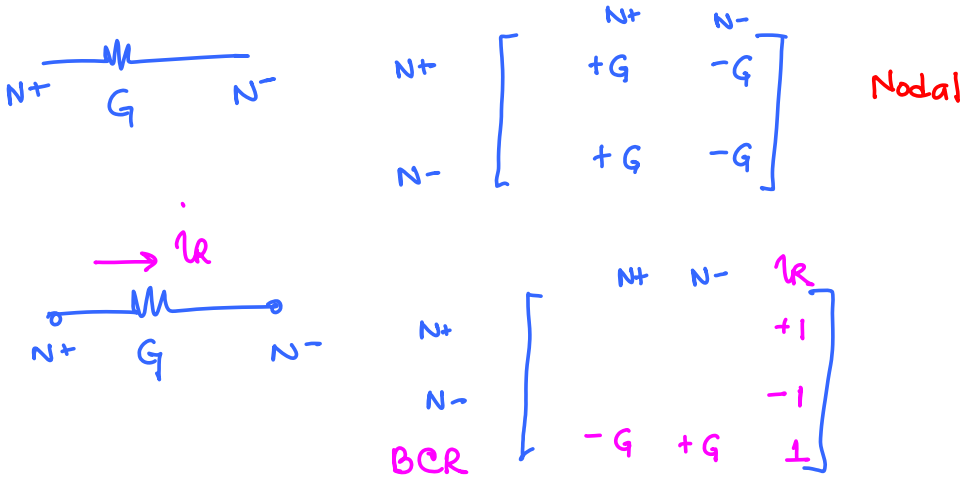


Modified nodal analysis (MNA)

Unknowns: node voltages + branch currents

Equations: KCLs + BCR of branches





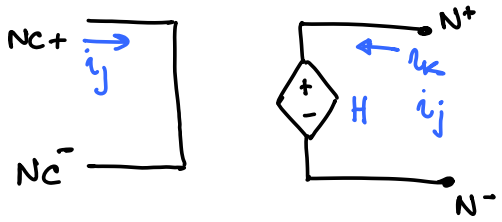
$$i_R - G(V_{N+} - V_{N-}) = 0$$

MNA formulation of resistor

MNA stamp for CCVS

$V_{N+} - V_{N-} = H\, i_j$

$V_{NC+} - V_{NC-} = 0$



	$+i_k$	$-i_k$	$+i_j$	$-i_j$	$-H i_j$
N+		+1			
N-		-1			
NC+			+1		
NC-			-1		
BCR(k)	1	-1			-H
BCR(j)			1	-1	

$V_{N+} - V_{N-}$

$V_{NC+} - V_{NC-}$

MYSPICE – res.h, vsrc.h (Components)

```
typedef struct resistor{
   char *name;  /* pointer to character string naming this instance */
   int pNode; /* number of positive node of resistor */
   int nNode; /* number of negative node of resistor */

   double value;  /* resistance */
   double conduct;  /* conductance */
} resistor;


typedef struct vsource{
   char *name;  /* pointer to character string naming this vsource */
   int pNode; /* number of positive node of vsource */
   int nNode; /* number of negative node of vsource */

   int branchNum ; /* number of branch */
   double voltage;  /* value of vsource */
} vsource ;
```

MYSPICE – main.c (Allocation)

```
main(ac, av)
char **av;
{
   char buf[MAXLINE];
   resistor   *Res[MAXELEM];
   isource   *Isrc[MAXELEM];
   vsource  *Vsrc[MAXELEM];
…
   int numRes = 0;
   int numIsrc = 0;
   int numVsrc = 0;
…
/* initialization */
   NodeArray = CALLOC(char *, MAXNODE);
   BranchArray = CALLOC(char *, MAXBRANCH);
   for(i = 0; i < MAXNODE; i++) {
            NodeArray[i] = CALLOC(char, MAXFIELD);
   }
   for(i = 0; i < MAXBRANCH; i++ ) {
            BranchArray[i] = CALLOC(char, MAXFIELD) ;
   }
   /* fill in ground node */
   strcpy(NodeArray[0], (char *)"0");
```

MYSPICE – main.c (Readin)

```
fpIn = fopen( inFile, "r" );
 while (fgets( buf, MAXLINE, fpIn ) != NULL) {
        if(tolower(buf[0]) == 'r')
        {
           /* resistor */
           numRes++;
           makeRes(Res, numRes, buf);
        }
        else if(tolower(buf[0]) == 'i')
        {
           /* isource */
           numIsrc++;
           makeIsrc(Isrc, numIsrc, buf);
        }
        else if(tolower(buf[0]) == 'v')
        {
           /* vsource */
           numVsrc++;
           makeVsrc(Vsrc, numVsrc, buf);
        }
        …
 }
 fclose( fpIn );
```

MYSPICE – main.c (Print, Setup, Stamp)

```
/* print circuit elements */
printRes(Res, numRes);
printIsrc(Isrc, numIsrc);
printVsrc(Vsrc, numVsrc);
…

/* setup circuit matrix */
numEqns = NumNodes+NumBranches+1;
cktMatrix = CALLOC(double *, numEqns);
for(i = 0; i <= NumNodes+NumBranches; i++) {
        cktMatrix[i] = CALLOC(double, numEqns);
}
/* setup Rhs vector */
Rhs = CALLOC(double, numEqns);

/* do any preprocessing */
setupRes(Res, numRes);
setupIsrc(Isrc, numIsrc);
setupVsrc(Vsrc, numVsrc);
…

/* stamp circuit matrix */
stampRes(Res, numRes, cktMatrix, Rhs);
stampIsrc(Isrc, numIsrc, cktMatrix, Rhs);
stampVsrc(Vsrc, numVsrc, cktMatrix, Rhs);
 …
```

MYSPICE – makeRes

```
void makeRes(Res, numRes, buf)
resistor *Res[];
int numRes;
char *buf;
{
    resistor *inst;
    int j, nodeA, nodeB, atoi();
    char name[MAXFIELD], node[MAXFIELD], num[MAXFIELD];
    double value, atof();

    j = 0;
    j = getNextField(buf, name, j);
    j = getNextField(buf, node, j);
    nodeA = getMappedNode(node);
    j = getNextField(buf, node, j);
    nodeB = getMappedNode(node);
    j = getNextField(buf, num, j);
    value = atof(num);

    inst = CALLOC(resistor, 1);
    inst->name = (char *)strdup(name);
    inst->pNode = nodeA;
    inst->nNode = nodeB;
    inst->value = value;
    Res[numRes] = inst;
}
```

MYSPICE – makeVsrc

```
void makeVsrc(Vsrc, numVsrc, buf)
vsource *Vsrc[];
int numVsrc;
char *buf;
{
    vsource *inst;
    int j, nodeA, nodeB, branchNum, atoi();
    char name[MAXFIELD], node[MAXFIELD], num[MAXFIELD];
    double value, atof();
   /* incrementing the number of branches */
    j = 0;
    j = getNextField(buf, name, j);
    branchNum = getMappedBranch(name);
    j = getNextField(buf, node, j);
    nodeA = getMappedNode(node);
    j = getNextField(buf, node, j);
    nodeB = getMappedNode(node);
    j = getNextField(buf, num, j);
    value = atof(num);

    inst = CALLOC(vsource, 1);
    inst->name = (char *)strdup(name);
    inst->pNode = nodeA;
    inst->nNode = nodeB;
    inst->branchNum = branchNum ;
    inst->voltage = value;
    Vsrc[numVsrc] = inst;
}
```

MYSPICE – setupRes

```
void setupRes(Res, numRes)
resistor *Res[];
int numRes;
{
    int i;
    resistor *inst;

    /* do any preprocessing steps here */
    for(i = 1; i <= numRes; i++) {
                inst = Res[i];
                inst->conduct = 1.0/inst->value;
    }
}
```

MYSPICE – setupVsrc

```
void setupVsrc(Vsrc, numVsrc)
vsource *Vsrc[];
int numVsrc;
{
    int i;
    vsource *inst;

    /* do any preprocessing steps here */
    for(i = 1; i <= numVsrc; i++) {
                inst = Vsrc[i];
                inst->branchNum += NumNodes;
    }
}
```

MYSPICE – stampRes

```
void stampRes(Res, numRes, cktMatrix, Rhs)
resistor *Res[];
int numRes;
double **cktMatrix;
double *Rhs;
{
    int i, pNode, nNode;
    double conduct;

    /* stamp resistor*/
    for(i = 1; i <= numRes; i++) {
                pNode = Res[i]->pNode;
                nNode = Res[i]->nNode;
                conduct = Res[i]->conduct;
                cktMatrix[pNode][pNode] += conduct;
                cktMatrix[nNode][nNode] += conduct;
                cktMatrix[pNode][nNode] -= conduct;
                cktMatrix[nNode][pNode] -= conduct;
    }
}
```

MYSPICE – stampVsrc

```
void stampVsrc(Vsrc, numVsrc, cktMatrix, Rhs)
vsource *Vsrc[];
int numVsrc;
double **cktMatrix;
double *Rhs;
{
    int i, pNode, nNode, branchNum;
    vsource *inst;
    double voltage;

    /* stamp voltage source*/
    for(i = 1; i <= numVsrc; i++) {
                inst = Vsrc[i];
                pNode = inst->pNode;
                nNode = inst->nNode;
                branchNum = inst->branchNum;
                voltage = inst->voltage;
                cktMatrix[pNode][branchNum] += 1;
                cktMatrix[nNode][branchNum] -= 1;
                cktMatrix[branchNum][pNode] += 1;
                cktMatrix[branchNum][nNode] -= 1;
                Rhs[branchNum] += voltage;
    }
}
```

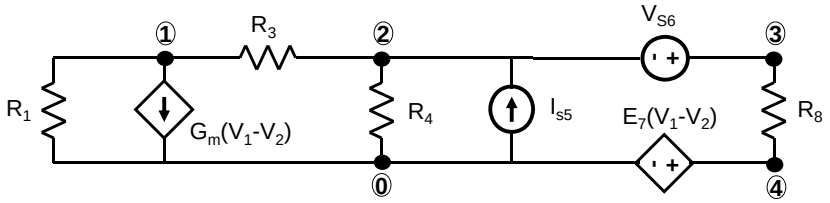
MYSPICE – node/branch Mapping

Not an efficient implementation

```
/* map a node name to an integer */
int getMappedNode(nodeName)
char *nodeName;
{
    int i;
    for(i = 0; i <= NumNodes; i++) {
                if(!strcmp(NodeArray[i], nodeName)) return( i );
    }
    /* node doesn't exist in NodeArray - so insert */
    NumNodes++;
    strcpy(NodeArray[NumNodes],  nodeName);
    return(NumNodes);
}

/* map a branch name to an integer */
int getMappedBranch(branchName)
char *branchName;
{
    int i;
    for(i = 0; i <= NumBranches; i++) {
                if(!strcmp(BranchArray[i], branchName)) return( i );
    }
    /* branch doesn't exist in BranchArray - so insert */
    NumBranches++;
    strcpy(BranchArray[NumBranches],  branchName);
    return(NumBranches);
}
```

MNA Example



$$\begin{bmatrix} \frac{1}{R_1}+G_m+\frac{1}{R_3} & -\left(G_m+\frac{1}{R_3}\right) & 0 & 0 & 0 & 0 \\ -\frac{1}{R_3} & \frac{1}{R_3}+\frac{1}{R_4} & 0 & 0 & -1 & 0 \\ 0 & 0 & \frac{1}{R_8} & -\frac{1}{R_8} & 1 & 0 \\ 0 & 0 & -\frac{1}{R_8} & \frac{1}{R_8} & 0 & 1 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ E_7 & -E_7 & 0 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ i_6 \\ i_7 \end{bmatrix} = \begin{bmatrix} 0 \\ I_{s5} \\ 0 \\ 0 \\ V_{S6} \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} Y_n & B \\ C & 0 \end{bmatrix} \begin{bmatrix} v \\ i \end{bmatrix} = S$$

From: A. Nardi