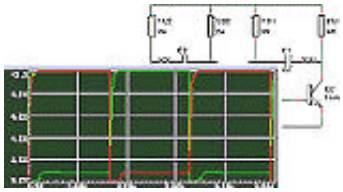
Le langage SPICE :

Sommaire



Avant-propos :
Ce dossier n´a certainement pas la prétention de se croire exhaustif, le sujet étant bien vaste... Il a uniquement pour but d´introduire les possibilités de simulation comportementale de SPICE afin de modéliser des fonctions qui peuvent être pluri-technologiques.
Pour plus d´informations sur SPICE, vous devriez vous rendre sur son site officiel...

Qu´est-ce que SPICE ?

SPICE est un noyau logiciel de **simulation électronique**.
La simulation rend en effet d´énormes services aux électroniciens pendant la phase de conception des circuits : elle permet de **limiter les essais réels**.
Ce procédé fait donc gagner du temps et de la souplesse puisque, dans l´idéal, on n´a plus besoin de câbler un circuit pour vérifier son fonctionnement...
Attention, la simulation n´évitera en **aucun cas** l´essai réel, seule étape permettant de valider rigou-reusement un circuit.

SPICE est architecturé autour d´un **algorithme complexe**, dont la version la plus aboutie a été développée par l´Université de Berkeley : **SPICE 3F5**.
SPICE ayant l´inconvénient de ne pas être très convivial (pas d´interface graphique, manipulation fastidieuse de fichiers,...), de nombreux logiciels de CAO intègrent un **noyau SPICE**, ce qui simplifie la simulation (*ex* : *PROTEL, PROTEUS, ICAP/4 de Intusoft, ORCAD, EED3 ou Edwin,...*).
L´appel au noyau SPICE est complètement transparent pour l´utilisateur : il suffit d´entrer le circuit sous forme graphique puis de lancer la simulation.

Pour effectuer les simulations, **SPICE** a besoin de connaître la manière dont se comportent les composants utilisés. Il utilisera donc des modèles logiciels des composants qu´il doit simuler. Ces modèles sont décrits sous formes de fichiers écrits en langage SPICE.

Les fichiers de modèles SPICE

Quel format ?

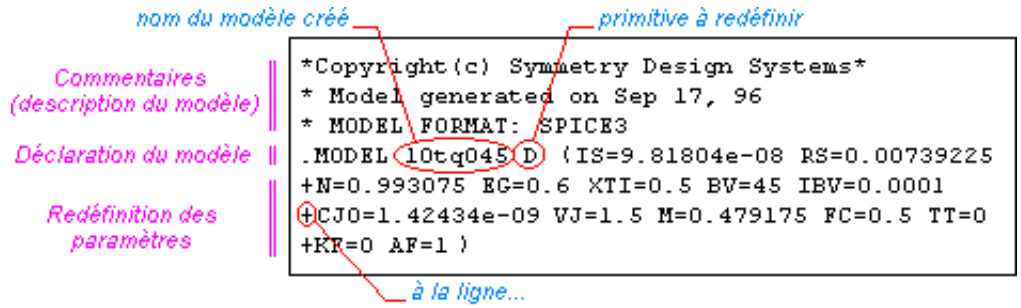
Concrètement, le modèle SPICE d´un composant est un fichier au **format texte**, qui est éditable avec n´importe quel éditeur de texte : Notepad, Wordpad, ou même Edit pour DOS,...
Les modèles SPICE sont souvent enregistrés avec l´extension **.CKT**, **.MOD**, ou encore **.MDL**.
On trouve de nombreux modèles à télécharger sur les sites des fabricants de composants électroniques (voir la rubrique Sites de simulation analogique).

Par exemple : un fichier UA741.CKT sera un fichier texte comportant le modèle SPICE de l´amplificateur opérationnel UA741.

Il est également possible de trouver des fichiers intégrant plusieurs modèles. Ils portent dans ce cas l´extension **.LIB**. Ces fichiers multi-modèles ont l´avantage de limiter le nombre de fichiers, mais l´inconvénient d´être plus longs à traiter lors de la simulation.

Le type . MODEL

Un fichier `.MODEL` permet de **personnaliser** les caractéristiques d´une primitive SPICE **paramètrable** (ex : diode, transistor...). Il est déclaré de la façon suivante :

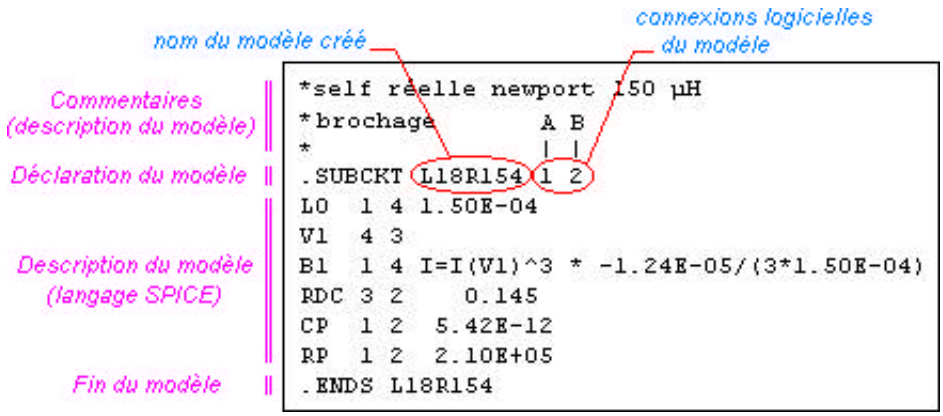


*Explications : ce fichier permet de définir un nouveau composant **10TQ045**, c´est une **diode** puisque la primitive de base utilisée est **D**.*

Précisions :
1- Il est possible de définir autant de modèles que l´on veut. Il suffira d´affecter à chaque modèle un nom différent (ex : 1N4001, 1N4002,...),
2- Un fichier `.MODEL` ne permet que de changer la valeur de paramètres prédéfinis (ex : tension de seuil d´une diode, coefficient de température d´une résistance,...) : il n´y a pas de création en soi ...,
3- On peut introduire des commandes `.MODEL` à l´intérieur même d´un macro-circuit `.SUBCKT`. On pourra ainsi créer des modèles **locaux** (s´ils sont définis avant l´instruction `.ENDS`), ou des modèles **globaux** si ils sont définis ailleurs dans le fichier.

Le type . SUBCKT

Un fichier `.SUBCKT` permet de créer un **macro-circuit**. Un macro-circuit est en fait un circuit résultant de l´association de plusieurs primitives SPICE. Pour toutes les versions de SPICE, un fichier `.SUBCKT` est constitué comme ceci :



Précisions :
1- Un commentaire s´écrit à la suite du caractère * (il est facultatif). Ce commentaire est souvent utilisé pour définir une **équivalence** entre la connexion logicielle du modèle (numéro des noeuds de connexion) et le brochage du composant réel (tel qu´il existe dans le logiciel de schéma).

*Par exemple : le commentaire du fichier-exemple explique le parallèle entre **A** et **B** (broches physiques de la self) et **noeud 1** et **noeud 2** (connexions logicielles du modèle).*

2- La déclaration `.SUBCKT` permet de signifier à SPICE que l´on souhaite définir un macro-composant (il ne faut pas oublier le "." devant SUBCKT).
3- Le composant est décrit sous la forme d´une **netlist** à l´aide du langage SPICE. Toutes les primitives utilisées sont connectées ensemble avec des noeuds (*NODES*). Chaque noeud étant affecté d´un numéro, l´ensemble structuré des connexions constituera la netlist.
4- La commande `.ENDS` permet d´informer SPICEque la description du modèle est terminée. A ne surtout pas oublier !

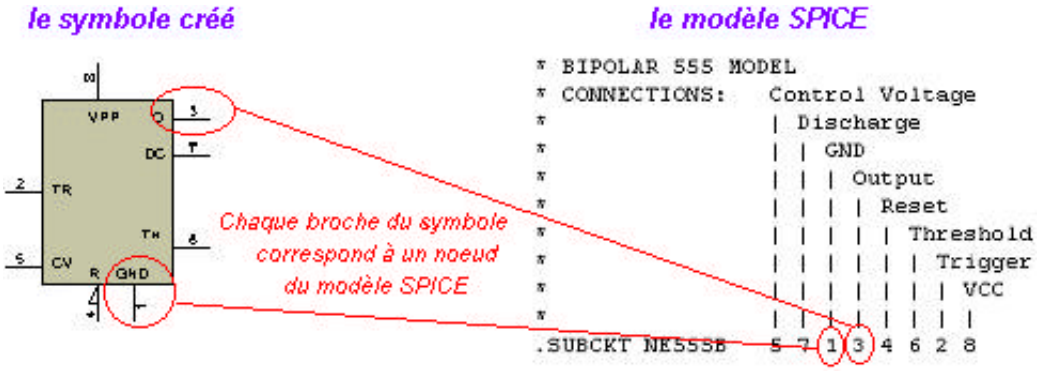
Utiliser les modèles SPICE

La finalité d´un modèle est évidemment d´être utilisé lors d´une simulation. Seulement voilà, il y a un petit problème : votre logiciel de CAO doit **faire le lien** entre le **modèle** et le **symbole** qu´il va utiliser dans sa bibliothèque...

Cela signifie qu´il ne suffit pas d´enregistrer le fichier-modèle sur le disque dur pour pouvoir l´utiliser : il faut encore **créer un composant correspondant** dans le logiciel de CAO.

Pour cela, il faut savoir que chaque logiciel utilise un principe différent (que ce soit PROTEL , PROTEUS,...). Cette manipulation est expliquée dans la **documentation** de votre logiciel. Cela dit, la procédure générale reste la même :

1- Placement des connexions électriques, puis dessin du symbole correspondant,
2- création du composant (device), grâce à la commande appropriée,
3- copie du fichier-modèle dans le répertoire des modèles propre au logiciel (s´il y en a un),
4- affectation d´un modéle à ce nouveau composant. En règle générale il faut éditer les propriétés du composant, spécifier le nom et le chemin d´accès du modèle, et enfin préciser le lien entre broches physiques (choisies lors de l´édition du symbole) et broches logicielles (définies dans la ligne `.SUBCKT`).



*Exemple avec PROTEUS : une fois créé le dessin, il faut **sélectionner** l´ensemble du symbole puis aller dans la barre de menu activer la commande **Make Device**. ISIS vous demande alors le **nom du composant**, et la **bibliothèque** dans laquelle il faut classer ce composant. Il faudra ensuite **lier le symbole** à son **modèle** grâce aux instructions suivantes :*

```
PRIMITIVE=ANALOGUE,SUBCKT
SPICEMODEL=NE555B,Ic555.LIB
SPICEPINS=CV,DC,GND,O,R,TH,TR,VPP
```

*Ces lignes doivent être écrites dans la rubrique **Other Properties** après avoir ouvert la boîte de dialogue propre au composant... L´**ordre des connexions** doit être le même pour le composant (pins CV DC GND O R TH TR VPP) et son modèle (5 7 1 3 4 6 2 8).*

Le langage SPICE - sa syntaxe

Généralités

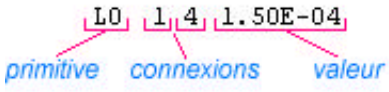
1- Le langage SPICE est une suite de lignes de code SPICE, chacune de ces lignes place ou définit un composant.
2- Pour fixer la valeur d´une primitive ou d´un paramètre, on peut utiliser indifféremment la notation réelle (*ex : -2.54, .5, 1743,...*), exponentielle (*ex : 25E-06, 17.3e+02,...*) ou enfin la notation symbolique : on écrit le coefficient multiplicateur à la suite du nombre (*ex : 25.3P, 72.4MEG, .54M*). Les coefficients disponibles dans SPICE sont :

F : femto, **P** : pico, **N** : nano, **U** : micro, **M** : milli, **K** : kilo, **MEG** : méga, **G** : giga, **T** : téra.

3- Le noeud **0** est réservé par SPICE : c´est la **masse**. Il peut cependant être utilisé pour relier des composants à la masse du circuit.

Placer des primitives simples - dites de bas niveau

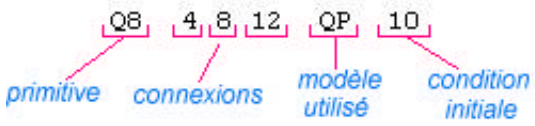
Constitution d´une ligne de **code SPICE** plaçant une primitive simple (non paramètrable) :



La ligne de cet exemple permet de brancher une **self** nommée **L0** entre les noeuds **1** et **4**, cette self a pour valeur **150 µH**.

Placer des primitives paramètrables - dites de haut niveau

Description d´une ligne de **code SPICE** plaçant une primitive paramètrable :



Le nom du modèle utilisé pour simuler la primitive doit être défini grâce à une commande `.MODEL` située dans le fichier.

Ici, on placera un **transistor** nommé **Q8** entre les noeuds **4**, **8** et **12**. Ce transistor sera un modèle **QP** (défini ailleurs grâce à une ligne .MODEL) qui aura pour valeur initiale **10**.

Cas particulier : les primitives R (résistance) et D (diode) peuvent être définies sans nom du modèle, dans ce cas SPICE utilisera le modèle par défaut (presque idéal).
ex : `D4 3 4` *ou* `R6 10 11 4,7K`.

Le langage SPICE - les primitives

Avant-propos

La **primitive** est l´élément de base qui sert à la construction des circuits et des modèles électriques. Chaque primitive correspond donc à un composant du circuit.

Comme on l´a vu auparavant, il existe deux types de primitives :
1- les primitives simples : ce sont les plus basiques, on peut simplement leur affecter une valeur. Il est impossible de redéfinir les paramètres internes. *Par exemple* : *self, condensateur, générateurs purs ou commandés,...*
2- les primitives paramètrables : on peut redéfinir leurs paramètres grâce à la commande `.MODEL`. *Par exemple* : *résistance, transistor, diode, MOS,...*

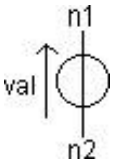
Attention : ce qui suit est une présentation **non exhaustive** des primitives du noyau SPICE 3F5. Certains attributs peu utilisés ont été volontairement ommis. Une documentation complète peut être obtenue sur le site officiel de SPICE 3F5.

Remarquons au passage qu´il existe des versions améliorées de SPICE, lesquelles proposent à l´utilisateur des compléments, voire de nouvelles primitives. C´est notamment le cas pour **PSPICE** ainsi que pour **XSPICE**, qui offrira entre autres toutes les primitives booléennes...

Primitive	Description	version	niveau
Vn	source de tension pure	*	bas
In	source de courant pur	*	bas
En	source de tension contrôlée par une tension	*	bas
Fn	source de courant contrôlée par un courant	*	bas
Gn	source de tension contrôlée par un courant	*	bas
Hn	source de courant contrôlée par une tension	*	bas
Bn	source universelle	3F5	bas
Rn	résistance	*	haut
Ln	inductance - self parfaite	*	bas
Cn	condensateur parfait	*	bas
Dn	diode	*	haut
Qn	transistor bipolaire (2 types : NPN et PNP)	*	haut
Jn	transistor à effet de champ (2 types : N et P)	*	haut
Mn	MOSFET (types N et P + 7 niveaux de modèles)	*	haut
Sn	interrupteur commandé par une tension	*	haut
Xn	utilisation d´un macro-circuit utilisateur .SUBCKT	*	haut
Kn	mutuelle inductance	*	bas
Tn	ligne de transmission parfaite	*	bas

Tableau 1 : résumé des primitives SPICE usuelles

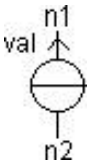
Source de tension V : `Vn n1 n2 type val`



n1 est le + de la tension de sortie.
n2 est le -.
type fixe la source en continu (type=DC) ou en sinusoidal (type=AC).
val représente la tension, exprimée en Volts.

Exemple : `V2 3 5 DC 10` *définit un générateur de tension* ***V2*** *entre les noeuds* ***3*** *et* ***5****. La tension est* ***continue*** *et vaut* ***10 volts****.*

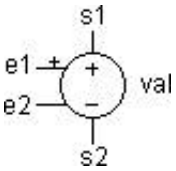
Source de courant I : `In n1 n2 type val`



n1 est le + du courant de sortie.
n2 est le -.
type fixe la source en continu *(type=DC)* ou en sinusoidal *(type=AC)*.
val représente le courant, exprimé Ampères.

Exemple : `I4 13 23 DC 24E-03` *positionne un générateur de courant* ***I4*** *entre les noeuds* ***13*** *et* ***23****. Le courant est* ***continu*** *et vaut* ***24 mA****.*

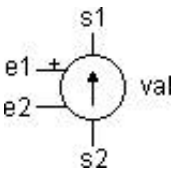
Source de tension commandée en tension : `En s1 s2 e1 e2 val`



s1 est le + de la tension de sortie.
s2 est le - de la sortie.
e1 est le + de la tension d´entrée.
e2 est le - de l´entrée.
val représente le coefficient de multiplication de la tension d´entrée (en V/V).

Exemple : `E1 10 20 1 2 .5` *positionne* ***E1****, un générateur de tension commandé en tension. La tension sort entre les noeuds* ***10*** *et* ***20****, la tension de référence du générateur arrive entre les noeuds* ***1*** *et* ***2****. La tension de sortie vaudra* ***0,5 fois*** *la tension d´entrée.*

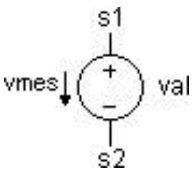
Source de courant commandée en tension : `Gn s1 s2 e1 e2 val`



s1 est le + du courant de sortie.
s2 est le - de la sortie.
e1 est le + de la tension d´entrée.
e2 est le - de l´entrée.
val représente le coefficient de multiplication de la tension d´entrée (en A/V).

Exemple : `G10 1 2 11 12 5E-03` *déclare* ***G10****, un générateur de courant commandé en tension. Le courant de ce générateur sort entre les noeuds* ***1*** *et* ***2****, la tension de référence entre sur les noeuds* ***10*** *et* ***20****. Le courant de sortie vaudra* ***0,005 fois*** *la tension d´entrée.*

Source de tension commandée en courant : `Fn s1 s2 vmes val`



s1 est le + de la tension de sortie.
s2 est le - de la sortie.
vmes est le nom de la sonde de courant.
val représente le coefficient de multiplication du courant d´entrée (en V/A).

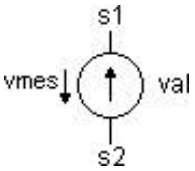
Attention : Cette primitive utilise une sonde de courant - **vmes** - qui est déclarée comme une source de tension de valeur nulle : `vmes s1 s2 DC 0`. Elle agira comme un **shunt** au niveau électrique. En général, on déclare la sonde juste après le générateur commandé.

Exemple :

```
F7 10 20 V2 500
V2 4 5 DC 0
```

Cela déclare ***F7****, un générateur de tension commandé en courant. La tension de ce générateur sort entre les noeuds* ***10*** *et* ***20****, le courant de référence est celui qui circule dans la sonde* ***V2****, placée entre les noeuds* ***4*** *et* ***5****. La tension de sortie vaudra* ***500 fois*** *le courant d´entrée.*

Source de courant commandée en courant : `Hn s1 s2 vmes val`



s1 est le + du courant de sortie.
s2 est le - de la sortie.
vmes est le nom de la sonde de courant.
val représente le coefficient de multiplication du courant d´entrée (en A/A).

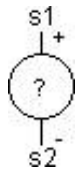
Attention : Cette primitive utilise une sonde de courant - **vmes** - qui est déclarée comme une source de tension de valeur nulle : `vmes s1 s2 DC 0`. Elle agira comme un **shunt** au niveau électrique. En général, on déclare la sonde juste après le générateur commandé.

Exemple :

```
H15 3 6 V12 4
V12 7 10 DC 0
```

*Cela déclare **H15**, un générateur de courant commandé en courant. Le courant issu de ce générateur sort entre les noeuds **3** et **6**, le courant de référence est celui qui circule dans la sonde **V12**, qui est placée entre les noeuds **7** et **10**. Le courant de sortie vaudra **4 fois** le courant d´entrée.*

Source universelle (SPICE 3F5 seulement) : `Bn s1 s2 type=valfunc`



s1 est le + de la grandeur de sortie.
s2 est le - de la sortie.
type déclare une source de tension *(type=V)* ou en source de courant *(type=I)* .
valfunc est une suite d´instructions qui fixe la valeur générée par la source.

Fonctionnement : La source Bn se programme grâce à une suite d´opérateurs et d´instructions placées après le signe = :

- *opérateurs :*

```
+ - * / ( ) ^
```

- *instructions :* x est un réel, r est en radians, n est un entier.

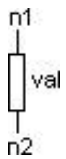
```
abs(x) sqrt(x) pwr(x,n) exp(x) ln(x) log(x),
sin(r) cos(r) tan(r) asin(r) acos(r) atan(r),
sinh(r) cosh(r) tanh(r) asinh(r) acosh(r) atanh(r),
```

- *sondes :* on peut utiliser tous les potentiels et courants du circuit, en utilisant des sondes :
  `V(n1)` : potentiel au noeud **n1**,
  `V(n1,n2)` : tension entre **n1** et **n2**,
  `I(Vn)` : courant circulant dans un **shunt** (source Vn de 0V à placer dans le circuit).

Exemples :

`B2 10 20 V=(22*V(1,5)-7)` *déclare une source de tension linéaire,*
`B20 1 3 I=40*LOG(V(3))` *déclare une source de courant logarithmique,*

Résistance : `Rn n1 n2 val (modele)`



n1 et *n2* sont les pattes de la résistance.
val est la valeur de la résistance (en Ohms).
modele donne le nom du modèle à utiliser pour la simulation, si on a redéfini une résistance avec .MODEL. A défaut la résistance est idéale.

Redéfinition : `.MODEL nom R(TC1=? TC2=? TEMP=? TNOM=?)`
nom est le nom du modèle créé,
TC1 et *TC2* sont les coefficients de température (voir formule ci-dessous),
TEMP est la température de simulation (27°C par défaut),
TNOM est la température de mesure de TC1 et TC2.

$$R(t) = R + TC1.(t-25) + TC2.t^2$$

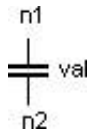
Exemple : `R12 3 2 4.7K` *place une résistance de **4,7 kohm** entre les noeuds **3** et **2**.*

Inductance parfaite : `Ln n1 n2 val (i0)`

n1 et *n2* sont les pattes de la self.
val est la valeur de la self (en Henry).
i0 est le courant initial dans la self en A, facultatif.

Exemple : `L10 4 7 2.2U` *place une inductance de* ***2.2 µH*** *entre les noeuds* ***4*** *et* ***7****.*

Condensateur parfait : `Cn n1 n2 val (v0)`



n1 et *n2* sont les pattes du condensateur (pas de polarisation).
val est la valeur de la capacité (en Farad).
v0 est la tension initiale aux bornes en V, facultatif.

Exemple : `C1 13 14 22N` *place un condensateur de* ***22 nF*** *entre les noeuds* ***13*** *et* ***14****.*

Diodes : `Dn n1 n2 (modele) (OFF)`

n1 est l´anode A,
n2 est la cathode K,
modele donne le nom du modèle à utiliser pour la simulation, si on a redéfini une diode avec .MODEL. Il existe un modèle prédéfini par défaut.
OFF est une condition initiale facultative : permet de bloquer au départ la diode.

Redéfinition : `.MODEL nom D(IC=? TEMP=? AREA=? IS=? RS=? N=? TT=? CJO=? VJ=? M=? EG=? XTI=? KF=? AF=? FC=? BV=? IBV=? TNOM=?)`

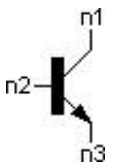
nom est le nom du modèle créé,
IS est le courant de saturation,
RS est la résistance passante,
CJO est la capacité de jonction,
VJ est le potentiel de jonction,
BV est la tension inverse,
IVB est le courant à la tension inverse,
TEMP est la température de simulation (27°C par défaut),
TNOM est la température de mesure des paramètres,
...les autres paramètres sont difficilement accessibles (constructeurs...).

Exemple :

```
D2 3 2 1N4003
.MODEL 1N4003 D (RS=.04 CJO=55E-12 IS=1.38E-09....)
```

Cela place une diode ***1N4003****, anode au noeud* ***3*** *et cathode au noeud* ***2****.*
Le ***.MODEL*** *décrit la diode 1N4003 en question.*

Transistors bipolaires : `Qn n1 n2 n3 modele (OFF)`



n1 est le collecteur C,
n2 est la base B,
n3 est l´émetteur E,
modele donne le nom du modèle de transistor à utiliser pour la simulation, défini avec .MODEL.
OFF est une condition initiale facultative : permet de bloquer au départ le transistor.

Redéfinition : `.MODEL nom <NPN ou PNP>(ICVBE=? ICVCE=? AREA=? TEMP=? IS=? BF=? BR=? IKF=? IKR=? NF=? NR=? ISE=? ISC=? NE=? NC=? RE=? RC=? RB=? RBM=? IRB=? VAF=? VAR=? VJE=? VJC=? VJS=? MJC=? MJE=? MJS=? CJC=? CJE=? CJS=? TF=? TR=? XTF=? VTF=? ITF=? PTF=? XCJC=? XTB=? EG=? XTI=? FC=? KF=? AF=? TNOM=?)`

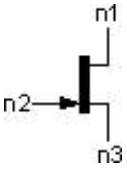
...On voit clairement que les paramètres ci-dessus sont difficilement accessibles. C´est pourquoi on utilise en général des modèles connus. En effet, les paramètres intrinsèques de transistors comme le 2N2222 ou le BC547 sont très faciles à trouver.

Exemple :

```
Q3 1 3 2 2N2222
.MODEL 2N2222 NPN(IS=3.108E-15 XTI=3 EG=1.11 ...)
```

*Cela place **Q3**, un transistor NPN **2N2222** défini par le .MODEL. La base est le noeud **3**, le collecteur est le noeud **1** et l´émetteur est le **2**.*

Transistors à effet de champ : `Jn n1 n2 n3 modele (OFF)`



n1 est le drain D,
n2 est la grille G,
n3 est la source S,
modele indique le nom du modèle à utiliser pour la simulation, redéfini avec .MODEL.
OFF est une condition initiale facultative : permet de bloquer au départ le transistor.

Redéfinition : `.MODEL nom <NJF ou PJF>(AREA=? TEMP=? VTO=? BETA=? LAMBDA=? IS=? RD=? RS=? CGS=? CGD=? PB=? FC=? B=? KF=? AF=? TNOM=?)`

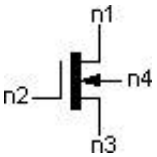
nom est le nom du modèle créé,
VTO est la tension de seuil,
BETA est le paramètre de transconductance,
LAMBDA est le paramètre de modulation de largeur du canal,
IS est le courant de saturation de grille,
RD est la résistance de drain,
RS est la résistance de source,
CGS est la capacité de G-S,
CGD est la capacité de G-D,
TEMP est la température de simulation (27°C par défaut),
TNOM est la température de mesure des paramètres,
...les autres paramètres sont difficilement accessibles (constructeurs...).

Exemple :

```
J2 1 3 2 J105
.MODEL J105 NJF(VTO=-7.25 BETA=5M LAMBDA=.035 RD=4.20E-01 ...)
```

*Cela positionne **J2**, un transistor JFET canal N type **J105** défini par le .MODEL.*
*La grille est au noeud **3**, le drain est le **1** et la source est le **2**.*

Transistors MOS -FET : `Mn n1 n2 n3 n4 modele (OFF)`



n1 est le drain D,
n2 est la grille G,
n3 est la source S,
n4 est le substrat (Bulk),
modele donne le nom du modèle à utiliser pour la simulation, défini avec .MODEL.
OFF est une condition initiale facultative : permet de bloquer au départ le transistor.

Redéfinition :
Les MOSFET sont un cas particulier de SPICE : il existe plusieurs types de modèles différents (dits L EVELs). SPICE 3F5 peut en effet simuler 7 niveaux de modèles (MOS1, MOS2, MOS3, BSIM1, BSIM2, MOS6 et BSIM3).
Pour information, les modèles MOSx comportent les paramètres suivants : `.MODEL nom <NMOS ou PMOS>(LEVEL=? L=? W=? ADE=? AS=? PD=? PS=? NRD=? NRS=? OFF=? ICVDS=? ICVGS=? ICVBS=? TEMP=? VTO=? KP=? GAMMA=? PHI=? LAMBDA=? IS=? RD=? RS=? CBD=? CBS=? PB=? CGSO=? CDSO=? CGBO=? KF=? AF=? RSH=? CJ=? MJ=? CJSW=? MJSW=? JS=? TOX=? LD=? UO=? UCRIT=? UEXP=? VMAX=? NEFF=? FC=? NSUB=? NSS=? NFS=? TPG=? XJ=? XD=? ALPHA=? ETA=? DELTA=? THETA=? KAPPA=? TNOM=?)`
C´est impressionnant non ?... ;-)

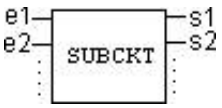
Exemple :

```
M4 2 5 6 6 NMOD1
.MODEL NMOD1 NMOS(LEVEL=1 VTO=2.474 RS=1.68 RD=0.0 ...)
```

*Cela déclare **M4**, un MOSFET canal N de type **NMOD1** défini par .MODEL. La gate est le noeud **5**, le*

*drain est le **2** et le substrat est connecté à la source au noeud **6**.*

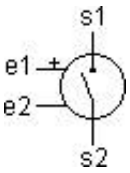
Macro-circuit SPICE : `Xn e1 e2 ... s1 s2 ... modele`



e1 est l´entrée 1 du macro-modèle.
s1 est la sortie 1 du macro-modèle.
... il y a autant de noeuds qu´il y a d´entrées/sorties au macro-modèle.
modele est le nom du macro-modèle à utiliser, défini par un .SUBCKT.

Exemple :
`X1 10 20 L18R154` *permet d´utiliser la **self de 150 µH** qui a été définie plus haut (cliquer pour voir). Cette self sera branchée entre les noeuds **10** et **20**.*

Interrupteur commandé en tension : `Sn s1 s2 e1 e2 modele`



s1 est la borne 1 de la sortie.
s2 est la borne 2 de la sortie.
e1 est le + de la tension de commande.
e2 est le - de la tension de commande.
modele est le nom du modèle à utiliser pour la simulation, définit par un .MODEL.

Redéfinition : `.MODEL nom SW(RON=? ROFF=? VT=? VH=?)`
nom est le nom du modèle créé,
RON est la valeur de résistance de l´interrupteur fermé (en ohms),
ROFF est la valeur de résistance de l´interrupteur ouvert (en ohms),
VT est la tension d´enclenchement de l´interrupteur (en V),
VH est la tension d´hystérésis à l´enclenchement (en V),

`si V(e1,e2) < VT - VH/2 alors l´interrupteur est` **`ouvert`**
`si V(e1,e2) > VT + VH/2 alors l´interrupteur est` **`fermé`**

Exemple :
```
S4 10 20 2 0 SWITCH
.MODEL SWITCH SW(RON=1U ROFF=10MEG VT=2.5 VH=0)
```
*Cela déclare **S4**, un interrupteur commandé par la tension entre le noeud **2** et la masse **0**. La tension de basculement est **2,5 V**. L´interrupteur (1 µohm en passant) est placé entre **10** et **20**.*

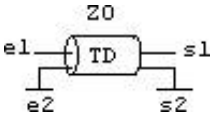
Induction mutuelle : `Kn L1 L2 val_k`

L1 est le nom de la self d´entrée,
L2 est le nom de la self de sortie,
val_k est le coefficient de couplage entre L1 et L2.

Calcul de val_k : on utilise l´induction mutuelle **M** : $val\_K = \frac{M}{\sqrt{L1 \cdot L2}}$

Exemple : `K1 L12 L20 0.6` *permet de coupler les inductances **L12** et **L20** avec un coefficient de couplage de **0.6** (pour en faire un transformateur par exemple). Les selfs **L12** et **L20** doivent être définies ailleurs, grâce à* Ln n1 n2 val.

Ligne de transmission parfaite : `Tn e1 e2 s1 s2 Z0=? TD=?`



e1 est l´entrée + de la ligne.
e2 est l´entrée - de la ligne.
s1 est la sortie + de la ligne.
s2 est la sortie - de la ligne.
Z0 est l´impédance caractéristique de la ligne (en ohms).
TD est la valeur du retard de ligne (en s).

Exemple : `T10 1 2 11 12 Z0=75 TD=50U` *place une ligne sans pertes **T10**, dont l´impédance est **75 ohms** et le retard vaut **50 µs**. On entre sur **1** et **2**, on sort entre **11** et **12**.*

Comprendre un modèle SPICE

Lecture d´un modèle SPICE

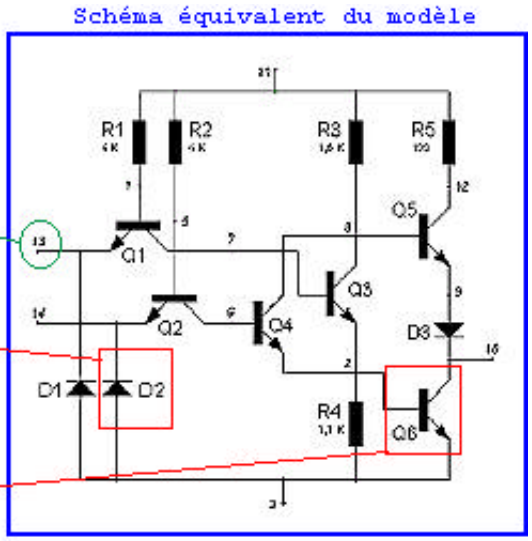
Pour interpréter un macro-modèle SPICE, il suffit en théorie de le mettre sous forme de **schéma équivalent**. Pour cela, on placera les composants les uns après les autres, au fil de la lecture des lignes de **code SPICE**, puisque chaque ligne de code SPICE définit un composant...

Remarque : le fonctionnement du modèle n´apparaissant en général pas clairement aux yeux, il s´agira ensuite de s´approprier le schéma et de l´étudier afin d´en déduire les lois de fonctionnement...

Exemple d´interprétation

Soient le modèle d´un circuit logique TTL 7402 et son circuit équivalent :

```
* Modèle NOR2 TTL 7402
* copyright Intusoft
* CONNECTIONS   eA   eB   s   vcc   gnd
.SUBCKT SN7402 13   14   10  11    3
Q2 6 1 14 QC
R1 1 11 4K
R2 11 5 4K
D1 3 13 DA
D2 3 14 DA
Q3 8 7 2 QC
Q4 8 6 2 QC
R3 11 8 1.6K
Q5 12 8 9 QC 5
D3 9 10 DA
R4 2 3 1.1K
Q6 10 2 3 QC 5
R5 11 12 133
Q1 7 5 13 QC
.MODEL QC NPN BF=20 BR=1 RB=70 RC=40 IS=1.0E-14 VA=50 CJE=10P
.MODEL DA D RS=40 IS=1.0E-14 CJO=1PF
.ENDS
```



Méthode de modélisation comportementale

1 - Elaboration du cahier des charges

Cette phase est la plus importante de la procédure, elle consiste à préparer la modélisation.
On écrit **noir sur blanc** ce que l´on attend de notre modèle. On établira également dans le cahier des charges l´affectation des entrées/sorties, et on définira les grandeurs d´entrées/sorties.

2 - Description du fonctionnement

Cette phase permet de **concrétiser** le modèle désiré.
Il s´agit de traduire le comportement désiré en un ensemble d´**équations de fonctionnement**, ou bien encore en blocs de fonctions (si le système est complexe).

Remarque : Plus on obtiendra des équations simples, plus facile sera la traduction en primitives SPICE. De plus, il vaut mieux utiliser de nombreuses équations simples plutôt qu´une équation complexe...

3 - Traduction des équations :

Une fois que l´on possède les équations de fonctionnement, il faut traduire ces équations dans le langage SPICE, en utilisant les primitives SPICE. Cela se fait généralement en 3 temps :

1- On traduit les équations en un **circuit équivalent** utilisant les symboles des primitives SPICE.
2- On place ensuite les numéros des différents **noeuds** sur le circuit.
3- Enfin, on peut écrire le **fichier .SUBCKT** du modèle SPICE, chaque ligne correspondant à une primitive du circuit.

Exemple n°1 : un filtre passe-haut du 2° ordre

Cahier des charges

- On souhaite modéliser un filtre analogique **passe-haut**, d´ordre **2**.
- La fréquence de coupure de ce filtre est de **f0 = 22050 Hz**, (donc w0 = 138544,24 rd/s).
- Le filtre est un Butterworth, l´amortissement est donc **z = 0,707**.

- L´impédance d´entrée est infinie, celle de sortie est nulle : le filtre ne doit pas perturber les autres montages.
- Le modèle s´appellera **HP22050** , l´entrée est le noeud **1**, la sortie est le **2** et le commun est le **20**.

Mise en équation

La fonction de transfert d´un filtre HP2 est :

$$H(jw) = \frac{\left(\frac{jw}{w0}\right)^2}{1 + \frac{2.z.jw}{w0} + \left(\frac{jw}{w0}\right)^2}$$

Une telle fonction peut être réalisée grâce à un réseau **R L C**, en sortant sur la self. La fonction de transfert d´un tel réseau est :

$$H(jw) = \frac{L.C.(jw)^2}{1 + R.C.jw + L.C.(jw)^2}$$

En identifiant les deux fonctions de transfert, on arrive aux formules suivantes :

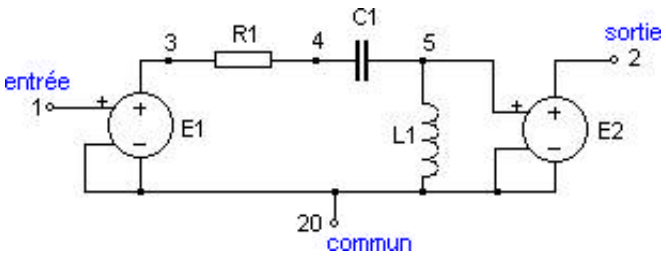
$$L.C.w0^2 = 1 \quad \text{et} \quad R.C = \frac{2.z}{w0}$$

Ces égalités nous permettent de calculer **R** et **L**, ceci en imposant la valeur de **C**.

En choisissant **C=100nF**, on peut calculer **R=102,06 ohms** et **L=521µH**

Traduction en circuit électrique

La synthèse de toutes les informations ci-dessus donne le circuit suivant :



*<u>Remarque</u> : on a rajouté sur ce circuit les 2 séparateurs, amplificateurs de gain **1**, afin de respecter le cahier des charges des impédances.*

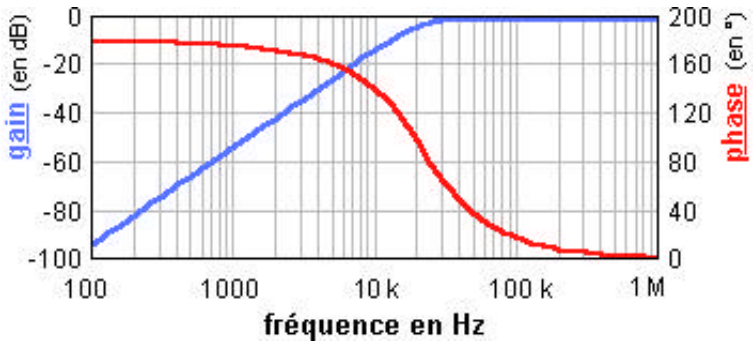
Le fichier modèle .SUBCKT

La traduction ligne par ligne du circuit ci-dessus donnera le modèle SPICE suivant :

```
* modèle du filtre HP2
* de fréquence 22050 Hz
*
.SUBCKT HP22050 1 2 20
E1 3 20 1 20 1
R1 3 4 102.06
C1 4 5 100N
L1 5 20 521U
E2 2 20 5 20 1
.ENDS HP22050
```

Vérification

La simulation du modèle ci-dessus à donné les résultats suivants :

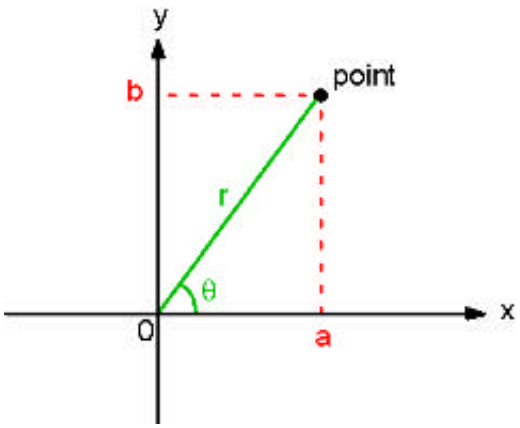


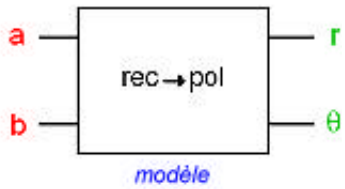
Conclusion : le cahier des charges est parfaitement respecté. On pourra changer la valeur des composants si l´on veut faire varier les paramètres du filtre.

Exemple n°2 : un convertisseur de coordonnées rectangulaires -> polaires

Cahier des charges

Ce second exemple propose de décrire un vrai modéle **comportemental**, indépendant de la technologie.
Le but est de synthétiser un modèle qui permette de calculer les coordonnées polaires d´un point à partir de ses coordonnées rectangulaires, comme le montre le diagramme ci-dessous :





Le principe de fonctionnement est donc simple : on entre les **coordonnés rectangulaires** , ici (a , b) sous la forme de tensions électriques; on récupèrera les tensions images des **coordonnés polaires**, ici (r ; θ) sur deux fils en sortie, θ étant compris entre **-pi** et **+pi**.

Remarques :
- Pour des raisons de simplification, il n´y a pas de potentiel de référence. Cela est géré en interne grâce au noeud n°0.
- Affectation des E/S logicielles : `noeud 1-> entrée a, 2->b, 3->r et 4->θ`
- On pourra appeler le modèle **REC2POL**.

Equations

Les formules générales de conversion rectangulaire-polaire sont :

$$r = \sqrt{a^2 + b^2} \quad \text{et} \quad \tan\theta = \frac{b}{a}$$

Ces formules seront parfaitement modélisables par SPICE en utilisant la primitive Bn, qui permet de calculer toutes sortes d´équations...
Attention cependant, il y aura un petit problème pour le calcul de l´arctangente en **radians** car la primitive Bn, comme toutes les calculatrices, retourne une valeur **décalée de pi** (donc erronée) **dans le cas où a est négatif**.
On va donc être obligé de prendre en compte 3 cas : **a** positif, **a** négatif et **b** positif, et enfin **a** négatif et **b** négatif.

L´aiguillage de ces 3 cas se fera avec une sorte de **structure en si-alors**, réalisée avec les primitives **interrupteur commandé** Sn utilisées en comparateur à **0**, afin de déterminer le signe de **a** et de **b**. L´agencement final sera prévu en utilisant les lois booléennes... :

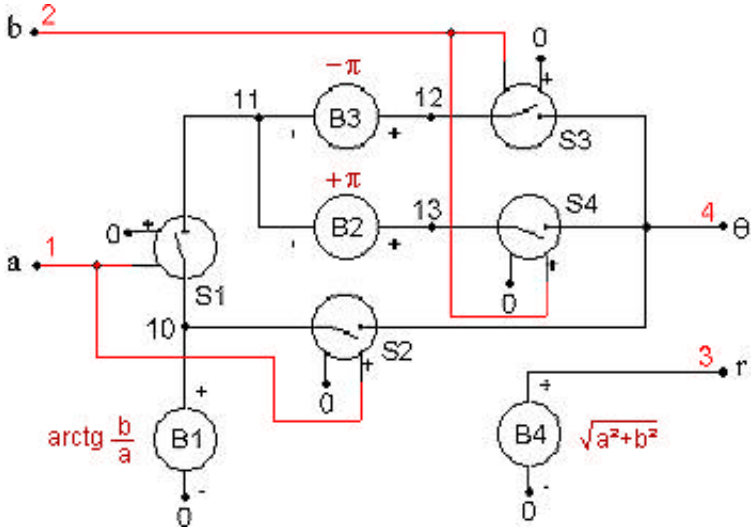
$$\theta = \text{arctg}\,\frac{b}{a} \quad \text{si } a > 0$$

$$\theta = \pi + \text{arctg}\,\frac{b}{a} \quad \text{si } a < 0 \text{ et } b > 0$$

$$\theta = -\pi + \text{arctg}\,\frac{b}{a} \quad \text{si } a < 0 \text{ et } b < 0$$

Traduction en circuit électrique

L´ensemble des spécificités vues ci-dessus peuvent être traduites par le circuit ci-dessous :



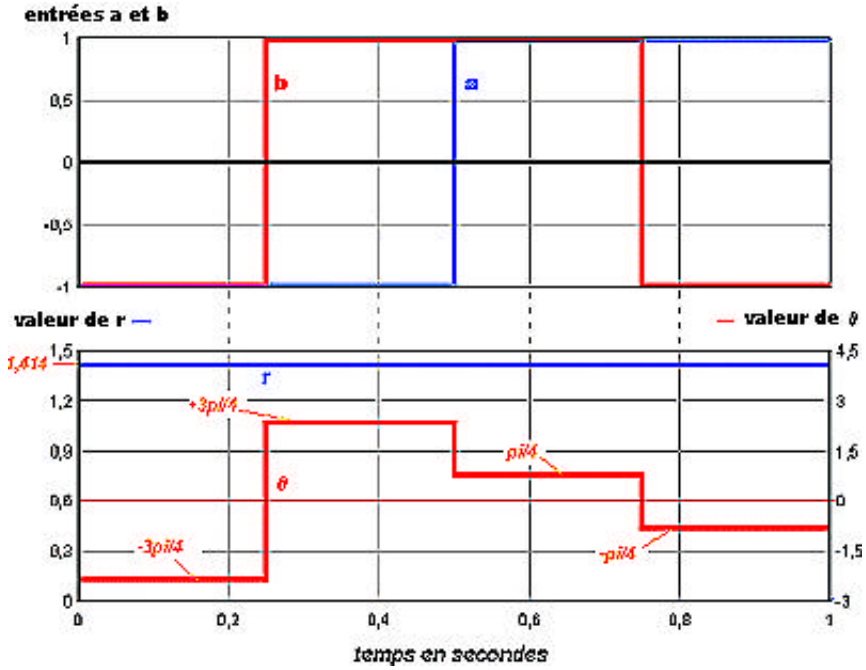
Le fichier modèle .S UBCKT (pour SPICE3F5)

La traduction ligne par ligne du circuit ci-dessus donnera le modèle SPICE suivant :

```
* modèle du convertisseur
* rectangulaire - polaire
* affectation : a b r O
*
.SUBCKT REC2POL 1 2 3 4
*
* fonctions booléennes :
*
S1 11 10 0 1 SWITCH
S2  4 10 1 0 SWITCH
S3  4 12 0 2 SWITCH
S4  4 13 2 0 SWITCH
*
* calculs mathématiques :
*
B1 10  0 V = ATAN ( V(2,0) / V(1,0) )
B2 13 11 V = 3.14159
B3 12 11 V = -3.14159
B4  3  0 V = SQRT ( V(1,0)^2 + V(2,0)^2 )
*
* redéfinition du modèle pour SWITCH :
*
.MODEL SWITCH SW (RON=1N ROFF=1T VT=1N VH=1P)
*
.ENDS
```

Vérification

La simulation du modèle ci-dessus avec des valeurs de **a** et **b** variant entre soit **-1** soit **+1** à donné les résultats attendus suivants :



*<u>Conclusion</u> : le graphe ci-dessus montre que le modèle fonctionne correctement dans **tous les cas** possibles. On remarque que les coordonnées sont calculées en temps réel par le simulateur. Ce genre de simulation peut dérouter au début car elle utilise des tensions ou des courants pour représenter toutes sortes de grandeurs (comme ici, des angles et des modules), il faut s´habituer !*

[Les autres Dossiers] [Retour au Sommaire]