REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTÈRE AUX UNIVERSITÉS

ECOLE NATIONALE POLYTECHNIQUE

Département **d'Electronique**

THESE

Présentée pour l'obtention du grade de

MAGISTER

en **ELECTRONIQUE** Option **Télécommunications**

par M^me^ **YAHIAOUI Chafia** née **MENZER**

Thème

ELABORATION D'UN PROGRAMME D'ANALYSE DE CIRCUITS INTÉGRÉS BIPOLAIRES

Soutenue le 3/07/1994 devant le jury composé de:

M^r^ H.FARAH	(Maître de conférence)	ENP	Président
M^me^ L.HAMAMI	(Chargée de cours)	ENP	Examinateur
M^r^ M.HADDADI	(Chargé de cours)	ENP	Rapporteur
M^r^ R.SAADOUN	(Maître Assistant)	ENP	Examinateur
M^r^ B.TRABELSI	(Chargé de cours)	ENP	Examinateur
M^r^ A .MERAGHNI	(Maître de conférence)	ENS	Invité

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTÈRE AUX UNIVERSITÉS

ECOLE NATIONALE POLYTECHNIQUE

Département **d'Electronique**

THESE

Présentée pour l'obtention du grade de

MAGISTER

en **ELECTRONIQUE** Option **Télécommunications**

par M^me **YAHIAOUI Chafia** née **MENZER**

Thème

ELABORATION D'UN PROGRAMME D'ANALYSE DE CIRCUITS INTÉGRÉS BIPOLAIRES

Soutenue le 3/07/1994 devant le jury composé de:

M^r H.FARAH	(Maître de conférence)	ENP	Président
M^me L.HAMAMI	(Chargée de cours)	ENP	Examinateur
M^r M.HADDADI	(Chargé de cours)	ENP	Rapporteur
M^r R.SAADOUN	(Maître Assistant)	ENP	Examinateur
M^r B.TRABELSI	(Chargé de cours)	ENP	Examinateur
M^r A .MERAGHNI	(Maître de conférence)	ENS	Invité

A Amar pour sa compréhension et son soutien permanent.
A mes parents, pour tout ce qu'ils m'ont permis de réaliser.
A Redouane , à Farouk.

J'exprime ma reconnaissance à Monsieur le Professeur E.Karakhanian de m'avoir confié cette étude .

Je tiens à exprimer toute ma gratitude à Monsieur M. Haddadi pour l'intérêt qu'il a manifesté pour ce travail et pour son soutien tant amical que scientifique.

Je remercie vivement Monsieur H.Farah pour l'honneur qu'il m'a fait en présidant ce jury de thèse et d'avoir accepté d'examiner ce mémoire.

Je suis heureuse de compter Madame L.Hamami et Monsieur B.Trabelsi parmi les membres du jury et qu'ils aient accepté d'examiner ce mémoire et d'en faire le rapport.

Mes remerciements s'adressent aussi à Monsieur R.Saadoun pour sa participation au Jury .

Je remercie Monsieur Meraghni , Maitre de conférence à L'E.N.S d'avoir accepté l'invitation de participer au Jury.

Je me garderai d'oublier Monsieur Hassen Fouchal pour son aide indispensable , particulièrement dans le domaine de l'informatique.

Un dernier remerciement à Monsieur B. Ait Yahia responsable du centre de calcul de Bab-Ezzouar pour les moyens qu'il a mis à ma disposition.

SOMMAIRE

INTRODUCTION

L'amélioration des performances et l'intégration accrue des circuits électroniques vont de pair avec une trés bonne connaissance de la physique liée à leurs composants et avec l'utilisation de moyens puissants et évolutifs de conception assistée par ordinateur (CAO)

L'utilisation de ces outils permet au concepteur de mettre au point des circuits de plus en plus complexes en un temps minimal. De plus ils lui permettent d'introduire dans la conception une démarche rigoureuse en abandonnant la plupart des hypothèses simplificatrices, restrictives et parfois incorrectes qu'il est amené à faire dans un processus de conception traditionnel.

En effet, s'il est possible de réaliser des études de circuits classiques à l'aide de maquettes, ceci est pratiquement impossible dans le cas des circuits intégrés pour lesquels les processus de fabrication sont ardus.

Ainsi sont apparus ces moyens de CAO qui vont de la simulation numérique des phénomènes physiques (diffusion, recombinaison ...) se produisant dans les composants à semi-conducteurs [1] à la simulation du fonctionnement électrique du dispositif [2,3] en passant par la simulation du processus technologique de fabrication [4].

Cet ensemble de logiciels, organisés autour d'une base de données commune, constitue alors une chaine de CAO qui couvre de manière complète tout le processus de conception et de réalisation d'un circuit intégré.

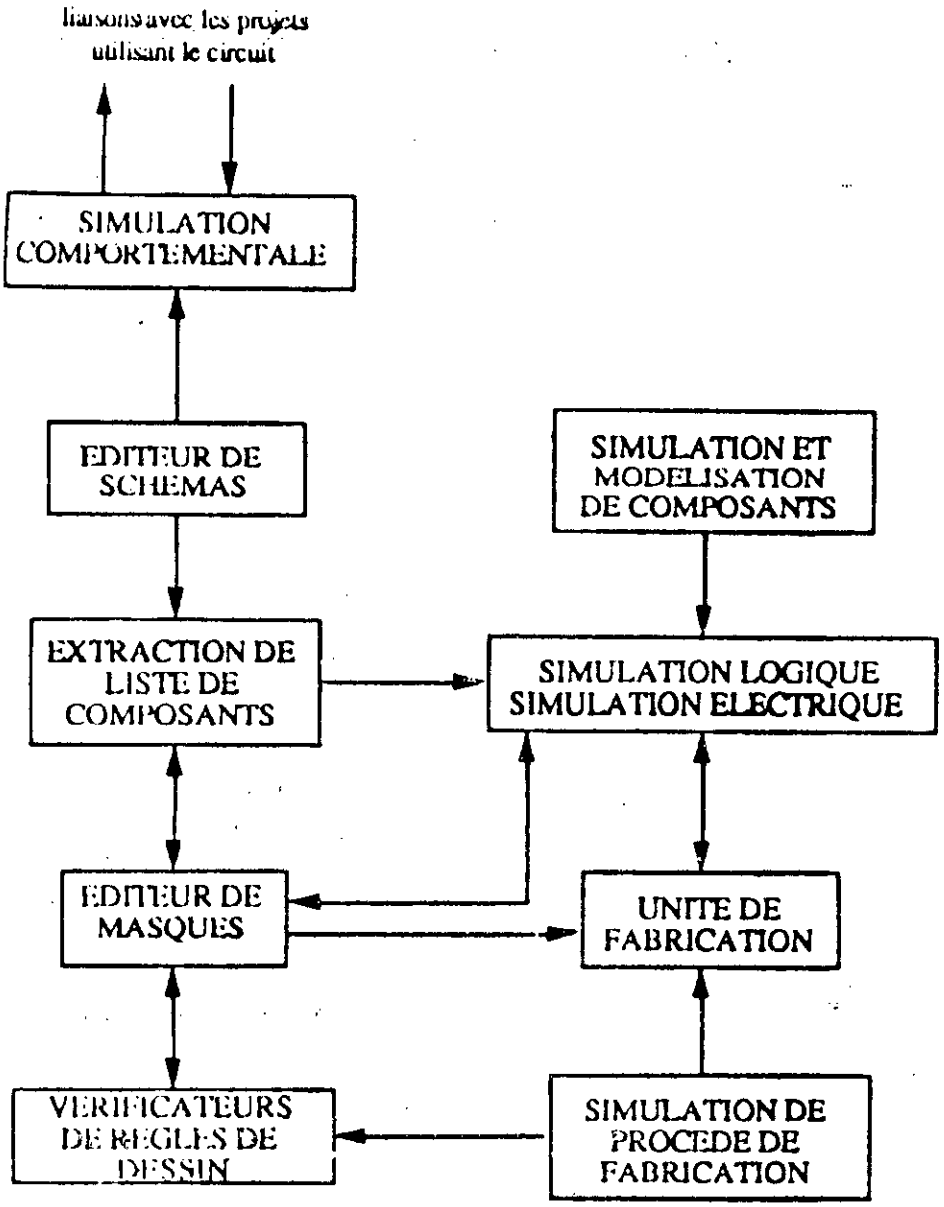
En effet nous allons faire un rapide panorama des étapes de conception des circuits intégrés vu au travers des logiciels mis en jeu.

1- Saisie de schéma et base de données

La saisie de schéma est la première phase du cycle de conception : un système hiérarchique permet de définir et de visualiser le schéma en cours de réalisation.

Ce logiciel communique à la base de donnée de conception toutes les informations de base qui seront exploitées aux différents stade de la conception.

D'autres applications périphériques à la saisie de schéma permettent de vérifier la cohérence des informations stockées dans la base de donnée ainsi que de générer des listes de connexions et d'informations qui seront ensuite communiqués à d'autres logiciels. Ce sont les techniques classiques de gestion de bases de données qui sont mises en oeuvre dans ces applications.



Chaîne de CAO d'un circuit intégré.

Quelques exemples sont:

- SDS (Silvar- Lisco)
- ESC (Dolphin Integration)
- NETED / SYMED (Mentor Graphics)

2- Outils d'édition de masques de circuits intégrés

Ces outils permettent de dessiner les différents niveaux de masques nécessaires lors de la gravure d'un circuit. La description peut être faite à plusieurs niveaux: soit du dessin au micron, pour lequel le concepteur dessine tous les éléments du circuit, soit une implantation sur une matrice précaractérisée ou prédiffusée où seules les tâches de placement et de routage des connexions sont à réaliser.

Un certain nombre d'outils périphériques aident au placement-routage, à la vérification des règles de dessin, et à l'extraction de listes de composants et de connexions. Des exemples de ces logiciels sont:

- MAGIC (Université de Berkeley)
- PRINCESS, CAL-MAP, CARDS (Silvar-Lisco)
- CHIPGRAPH (Mentor graphics)

3- Simulation de composants et de procédés technologiquess

Au plus haut niveau d'une chaine CAO se situent les simulateurs de composants électroniques élémentaires et les simulateurs de procédés technologiques de fabrication de circuits. Ces applications permettent pour les premières de créer des modèles analytiques et d'aider à la compréhension de la physique des composants, et pour les autres, de mettre au point de nouveaux procédés de fabrication de composants et de circuits.

Les techniques numériques généralement mises en oeuvre dans ces applications sont: les méthodes par éléments finis, et les méthodes particulaires Monté-Carlo. Des exemples de ces logiciels sont:

En simulation de technologies:

- SUPREM (Université de Stanford)
- FEEDS (IBM)
- SAMPLE (Université de Berkeley)

En simulation de composants:

- MINIMOS (Institut d'électronique, Autriche)
- FIELDAY (IBM)
- MONACO (Institut d'électronique, Orsay Université ParisXI)

4 - La simulation de circuits électriques

C'est en général le maillon essentiel d'une chaine de CAO en électronique. Les simulateurs de circuits électriques font partie des premières applications informatiques qui ont été mises en oeuvre dans le domaine de la CAO en électronique. Ils ont pour but de remplacer une expérimentation lente, coûteuse et parfois même impossible.

On distingue quatre niveaux de simulation et cela suivant le niveau hiérarchique auquel se fait la description du circuit dans un simulateur.

a. Simulateur comportemental ou fonctionnel

Ces simulateurs servent à valider les spécifications d'un projet d'architecture d'un système ou d'un circuit et peuvent en général travailler à n'importe quel niveau hiérarchique en utilisant des blocs fonctionnels qui réalisent les fonctions désirées. Ces blocs sont décrits par l'utilisateur dans un langage évolué.

Un simulateur fonctionnel serait plutôt à rapprocher d'un simulateur logique mais dans lequel les blocs pourraient réaliser une macrofonction quelconque définie par l'utilisateur en langage évolué, les signaux pouvant varier de façon continue. Ce type de simulateur est bien adapté à la simulation de circuits ayant des fonctionnalités à la fois logiques et analogiques. Des exemples de tels logiciels sont:

- SPLICE (Université de Berkeley)
- DIANA (Université de Louvain)

b. Simulateur logique

Dans un simulateur logique, les composants de base sont les différentes fonctions booléennes auxquelles est attribué un temps de traversée de l'information. En général, une importante bibliothèque de fonctions de haut niveau, et une hiérarchisation importante des blocs, permettent facilement la description de circuits logiques complexes.

Un simulateur logique produit des résultats sous forme de chronogrammes indiquant l'état logique des noeuds d'un circuit en fonction de vecteurs de test fournis par l'utilisateur, mais peut également fournir des informations du puissance consommée et temps de propagation. Des exemples de tels logiciels sont:

- HILO 3
- QUICKSIM (Mentor graphics)

c. Simulateur électrique

C'est un simulateur dans lequel les modèles de base sont les composants des circuits électriques: source de tension, source de courant, résistance, condensateur et inductance.

Les autres composants électroniques sont définis à partir d'un schéma équivalent faisant intervenir les composants de base. Les résultats d'une simulation électrique sont les variables d'un circuit: tensions, courants, puissance..., qui sont calculées à partir des grandeurs électriques appliquées aux entrées et des conditions de fonctionnement du circuit étudié.

Des simulations en présence de bruit (les composants ont leur propre modèles de bruit), et du calcul de sensibilité à des paramètres sont souvent possibles. Les logiciels de simulation électrique les plus connus sont:

- IMAG I et II (France)
- SPICE 2 (Université de Berkeley)
- ASTEC 3 (CISI)
- CIRCEC (Thomson)

d. Simulateur multi-niveaux et multi-modes

Les simulateurs multi-niveaux ont une structure hiérarchisée de description des éléments d'un circuit. Ceux-ci vont du composant élémentaire au macromodèle complexe. Ces derniers qu'ils soient logiques ou analogiques sont décrits soit à partir de leur schéma électrique, soit par une représentation fonctionnelle.

Les simulateurs multi-modes permettent un traitement simultané de signaux logiques et de signaux analogiques lors d'une simulation, et ceci de façon totalement transparente à l'utilisateur. L'exemple d'un tel logiciel est :

- ANDI (Silvar-Lisco)

Dans ce mémoire un simulateur électrique de circuits intégrés bipolaires a été élaboré, il est basé sur l'idée majeure d'une analyse à moindre coût en temps et en espace mémoire.

Comme l'efficacité de tout simulateur électrique repose essentiellement sur la qualité du modèle adopté pour ses composants ainsi que sur les techniques mathématiques utilisées nous y avons consacré les chapitres qui suivent.

5- Plan de l'étude

Le Chapitre I est consacré à l'étude des différents modèles de transistors bipolaires, modèles destinés tant à la constitution de leurs schémas équivalents qu'à leur étude physique .

Nous retiendrons de cette étude le modèle d'Ebers-Moll dont nous proposerons des méthodes d'extraction de ses paramètres.

En fin de chapitre, nous effectuons une validation du modèle par la simulation de l'inverseur.

Le Chapitre II est consacré à l'acquisition topologique pour la description du circuit à simuler ainsi qu'aux moyens mathématiques de formulation des équations qui le décrivent. Les travaux qui ont exercé l'influence la plus profonde dans ce domaine au cours des dernières années sont certainement ceux de Hatchel et al[5], où ils proposent d'écrire les équations de fonctionnement du dispositif en incluant toutes les variables électriques du circuit.

Cette approche permet l'obtention d'un système d'équations de dimensions beaucoup plus élevé, mais aussi beaucoup plus "creux". Pour cela, elle a été appelée "approche du tableau creux" (Sparse Tableau Approach) et sur laquelle notre travail s'est basé pour la formulation des équations qui décrivent le circuit à analyser.

Dans le chapitre III, un bref exposé des différentes techniques numériques d'intégration ainsi que les conditions de stabilité de chacune d'elles a été effectué. En effet le système algèbro-différentiel $F(X, \frac{dX}{dt}, t) = 0$ est obtenu quelque soit le moyen de formulation considéré et afin de résoudre pareil système, il a fallu discrétiser les dérivées en leur substituant une expression dont le calcul dépend de la méthode d'intégration choisie qui dans notre cas est la méthode d'Euler.

Le chapitre IV sera consacré, en premier lieu, aux algorithmes numériques nécessaires à la résolution du système $F(X, \frac{dX}{dt}, t) = 0$ qui sera linéarisé et ramené à un système matriciel de la forme : $A \ . \ X = B$.

Aprés un tour d'horizon sur les diverses méthodes de résolutions des systèmes matriciels, on concluera sur la technique de résolution des matrices éparses basée sur l'algorithme de Crout .

Le Chapitre V consiste en deux parties dont la première est consacrée à la description des différents modules du simulateur puisqu'il est structuré ainsi en vue d'une réduction considérable de sa complexité et d'une possibilité d'extension .

Dans la seconde partie une illustration sur les possibilités du programme est faite par le biais d'exemples de simulations de circuits logiques dans les différentes technologies TTL, ECL et I^2L.

Notation

β	gain en courant émetteur - commun en régime continu
Cje,Cjc	capacité de la jonction émetteur-base et collecteur-base (F)
$\varepsilon_c, \varepsilon_c, \varepsilon_b$	constante diélectrique de l'émetteur, du collecteur , de la base (F/cm)
ft	fréquence de coupure (Hz)
Ib,Ie,Ic	courant de base, d'émetteur, de collecteur (A)
Is	courant de saturation (A)
k	constante de Boltzmann (eV/K)
Qb (Qbo)	charge totale dans la base (à polarisation nulle)
q	charge de l'électron
ρ_c	résistivité du collecteur (Ωm)
T	température (K)
τ_b, τ_{br}	temps de transit direct, inverse (s)
τ_c	temps de transit du collecteur (s)
$\tau_{cb}(scr), \tau_{eb}(scr)$	temps de transit de la zone de charge d'espace collecteur-base, émetteur-base (s)
$\tau f, \tau r$	temps de transit direct, inverse (s)
τn	durée de vie des électrons dans la base (s)
U_T	tension thermique(=kT/q) (V)
Vbc,Vbe	tension de polarisation appliquée aux bornes des jonctions collecteur-base et émetteur-base (V)
Vce	tension appliquée entre le collecteur et l'émetteur (V)
Vjc,Vje	barrière de potentiel de la jonction collecteur-base, émetteur-base (V)
υ_s	vitesse de saturation des électrons (cm/s)
Wb,Wc,We	largeur de la zone neutre de la base , du collecteur,de l'émetteur (cm)
Wb_o	largeur de la base métallurgique.
Wcib	largeur de la base induite (cm)
Xdc	épaisseur de la zone de charge d'espace collecteur-base (cm)
Xne,Xpe	limites de la zone de charge d'espace de la jonction émetteur-base côté émetteur et côté base (cm)
Xnc,Xpc	limites de la zone de charge d'espace de la jonction base-collecteur côté collecteur et côté base (cm)

CHAPITRE I

MODELE DU TRANSISTOR BIPOLAIRE

I.1 ETUDE DES MODELES DU TRANSISTOR BIPOLAIRE

Les modèles du transistor bipolaire peuvent être distingués suivant la finalité désirée.Les deux principales distinctions sont basées sur la façon d'appréhender un modèle:

- Soit en représentant le dispositif sous la forme d'un circuit équivalent formé d'éléments actifs et passifs. Souvent il est considéré comme une "boite noire" et on ne s'intéresse qu'à ses caractéristiques électriques externes.
  Dans cette dernière représentation les valeurs des éléments du circuit sont exprimées mathématiquement grâce aux paramètres physiques et géomètriques du dispositif considéré d'où l'appellation de ces modèles : modèles mathématiques à circuit équivalent.
- Soit en partant du fonctionnement interne et de la technologie du dispositif, ce qui permet de mieux comprendre les phénomènes physiques qui gouvernent son fonctionnement. Ces modèles physiques servent surtout comme outils de développement du dispositif.

I.1.1 Modèles physiques

Ces modèles ne sont que l'aboutissement d'une étude physique fine des mécanismes internes qui régissent le fonctionnement global du dispositif. Les principaux effets physiques qui interviennent sont liés au phénomène de transport des porteurs électriques (électrons et trous) ainsi que la variation du champ électrique à l'intérieur des régions considérées.

Ces modèles sont alors obtenus à partir de la résolution des équations de transport ainsi que les équations de conservation de la charge et l'équation de Poisson.

Cependant ces dernières ne peuvent pas être résolues analytiquement à cause de la complexité dûe à la non-linéarité des systèmes d'équations aux dérivées partielles. L'appel au calcul assisté par ordinateur s'est alors avéré nécessaire, ce qui a généré plusieurs types de modèles physiques suivant les dimensions du dispositif ainsi que son rayon d'application.

On distingue:

- Les modèles de Monte-Carlo[1] qui sont des modèles statistiques qui fournissent la solution des équations de transport, modèlisant ainsi les dispositifs de dimensions inférieures à quelques dizaines de microns.

Une importante utilisation de ce modèle réside dans le calcul des paramètres physiques du matériau. Il permet aussi d'effectuer des simulations bidimensionnelles (2D) et même tridimensionnelles (3D).

- Les modèles classiques [6-8] fournissent les solutions des équations de Boltzmann pour les dispositifs dont les dimensions dépassent quelques dizaines de microns. Ces équations ont été formulées numériquement grâce aux méthodes des éléments finis[9-14].

  Afin de surmonter la grande complexité des calculs, des programmes trés élaborés ont été mis au point tels que : SUPREM [4] qui traite les modèles 1D, et ainsi il a pu donner la répartition des porteurs électriques et du champ qui influe sur leur déplacement connaissant le profil du dopage.

  Avec la venue de la VLSI, l'analyse 1D s'est avérée insuffisante vu la réduction des dimensions du dispositif, ce qui a amené J.W Slotboom à reconsidérer les phénomènes 2D ayant lieu [15]. Pour cela des programmes tels que le programme FIELDAY [16] ont été mis au point, ce dernier allant même à une analyse 3D.

Cette approche de modèlisation est trés utile dans son application à l'étude du dispositif seul ou à un réseau simple mais devient trop lourde à manipuler dés que le réseau se complique un peu.

I.1.2 Modèles mathématiques

Actuellement, les logiciels de circuits élaborés utilisent dans leur majorité des modèles mathématiques à circuit équivalent plus adaptés à une analyse fonctionnelle ou électrique.

Du modèle bien connu de Giacoletto[17], nous ne retiendrons que ses deux formes simplifiées valables en faibles signaux:

- La forme en T valable pour des fréquences inférieures à la fréquence de coupure du gain en courant dans la configuration considérée.
- La forme en π valable pour des fréquences de l'ordre de la fréquence de coupure du gain en courant en base-commune.

L'intérêt essentiel de ce schéma équivalent est qu'aucun de ses paramètres ne dépend de la fréquence.

Le modèle d'Ebers-Moll [18] qui, depuis sa formulation en 1954, a été le modèle large-signaux le plus utilisé, est basé directement sur la physique du transistor et couvre ainsi tous les régimes de fonctionnement : actif direct et inverse, saturé et bloqué.

En 1957 Beaufoy et Sparkes [19] ont analysé le transistor bipolaire du point de vue de la notion de contrôle de charge. Cette notion a été largement développée par la suite , pour l'élaboration du modèle complet de Gummel-Poon [20] où le courant de collecteur est exprimé en fonction des tensions au niveau des jonctions et de la charge globale dans la base.

Dans ce modèle plusieurs effets secondaires non considérés auparavant ont été incorporés, ce qui l'a rendu trés proche de la physique. Cependant le nombre de paramètres requis est assez élevé (de l'ordre de 40). En les réduisant on aboutit à la forme simplifiée équivalente au modèle d'Ebers-Moll.

Des modèles plus élaborés [21,22] sont utiles pour des études détaillées du transistor mais ne trouvent pas d'extension à leur utilisation à cause de leur complexité.

Avec le développement de la technologie, l'investigation dans le fonctionnement interne du transistor bipolaire est devenue possible, ce qui a permis l'identification de plusieurs effets secondaires tels que :

- L'effet d'Early [23], qui consiste en la modulation de la largeur de la base par la tension inverse de collecteur (en régime direct), l'effet de la génération et de la recombinaison des porteurs dans la zone de charge d'espace [24] qui a justifié la décroissance du gain à faible niveau d'injection et enfin l'effet de l'élargissement de la base aux forts niveaux du courant de collecteur dit aussi effet Kirk [25].

De même que les modèles physiques, les modèles mathématiques 1D ont trouvé leur limitation avec la venue de la VLSI , il a fallu tenir compte des effets bidimensionnels tels que l'effet de polarisation transverse dans la zone frontale de l'émetteur dû au courant de la base, ainsi que le phénomène de l'injection latérale des porteurs [26].

I.2 CHOIX DU MODELE

Nous venons de passer en revue plusieurs modèles relatifs au transistor bipolaire et les considérations sur lesquelles se basera notre choix du modèle dépendent de la facilité d'extraction de ses paramètres, de sa commodité ainsi que de la précision de son utilisation spécifique. De ce fait, notre choix s'est porté sur le modèle mathématique d'Ebers-Moll dont les qualités essentielles se caractérisent par :

- Des paramètres indépendants du point de polarisation.
- Le fait qu'il se présente sous la forme d'un réseau électrique qui couvre à la fois les opérations à faibles signaux linéaires et celles à forts signaux non-linéaires ainsi que les quatre régimes de fonctionnement entre l'actif normal, l'actif inverse, le saturé et le bloqué.

Il a été utilisé sous différentes formes dans la majorité des programmes d'analyse de circuits tels que: IMAG I (en France) [3], NET1 [27] et SPICE 2 [28] (aux USA).

I.2.1 Description du modèle mathématique d'Ebers-Moll

Dans l'élaboration de ce modèle, les auteurs[18] ont dû poser certaines hypothèses de base :

- Les résistivités des matériaux de l'émetteur, de la base et du collecteur sont faibles.
- Les densités de courant injectés sont faibles.
- L'élargissement de la région de charge d'espace est négligé.
- Les jonctions d'émetteur-base et de collecteur-base ont des caractéristiques de diodes.
- Les phénomènes de recombinaison en surface et dans la région de charge d'espace de l'émetteur sont ignorés.
- Le gain en courant est constant, il représente le facteur de transport dans la base ainsi que l'efficacité d'injection.

Plusieurs approches ont été adoptées afin de décrire un modèle en fonction de ses paramètres, la plus commune a été de spécifier certains d'entre eux comme fonction de la polarisation et de décrire cette dépendance sous forme tabulaire ou à travers des expressions paramètriques.

La représentation adoptée pour le modèle d'Ebers-Moll est donnée par des expressions analytiques entre les courants d'émetteur et de collecteur et les tensions des jonctions émetteur-base et collecteur-base. Ces expressions font aussi intervenir les gains en courant direct et inverse (en montage base-commune) ainsi que les courants inverses théoriques des jonctions émetteur-base et collecteur-base assimilées à des diodes sous la forme suivante:

$$I_e = I_{es} \left| \exp(qV_{be}/kT) - 1 \right| - \alpha_I I_{cs} \left| \exp(qV_{bc}/kT) - 1 \right| \qquad (I.1a)$$

$$I_c = \alpha_N I_{es} \left| \exp(qV_{be}/kT) - 1 \right| - I_{cs} \left| \exp(qV_{bc}/kT) - 1 \right| \qquad (I.1b)$$

Les termes I_{es}, I_{cs}, α_N et α_I constituent le groupe des quatre paramètres statiques du modèle dEbers-Moll:

I_{es} représente le courant de saturation inverse de la jonction émetteur-base (E-B), la jonction collecteur-base (C-B) étant en court-circuit.

I_{cs} représente le courant de saturation inverse de la jonction collecteur-base (C-B) jonction émetteur-base (E-B) étant en court-circuit.

α_N représente le gain en courant en mode de fonctionnement direct du montage base-commune.

α_I représente le gain en courant en mode de fonctionnement inverse du montage base-commune.

Ces deux derniers paramètres peuvent être exprimés respectivement en fonction des gains en courant du montage émetteur-commun en mode de fonctionnement direct et inverse comme suit:

$$\alpha_N = \frac{\beta_N}{\beta_N + 1} \qquad (I.2a) \qquad \text{et} \qquad \alpha_I = \frac{\beta_I}{\beta_I + 1} \qquad (I.2b)$$

On notera que les paramètres statiques ne sont pas indépendants, mais liés par l'expression de réciprocité suivante [18] :

$$\alpha_N I_{es} = \alpha_I I_{cs} = I_S \qquad (I.3)$$

Les équations (I.1a) et (I.1b) sont similaires à celles des circuits ordinaires couplés. Sous toutes les conditions de polarisation des jonctions, le courant qui traverse une jonction est dû à deux composantes : l'une propre à cette même jonction et l'autre due à la seconde jonction. Comme les jonctions E-B et C-B sont assimilées à des diodes qui obeissent à une loi en $exp(qV/kT)$ [29], on écrira alors les courants comme suit:

$$I_1 = I_{es} [\exp (qV_{be}/kT) - 1] \qquad (I.4a)$$

$$I_2 = I_{cs} [\exp (qV_{bc}/kT) - 1] \qquad (I.4b)$$

Afin de tenir compte à la fois du phénomène de diffusion des courants et de la formation des zones de charge d'espace, on ajoute aux équations (I.1a) et (I.1b) des relations qui définissent les effets dynamiques de stockage des charges [30]:

Au niveau des zones quasi-neutres, la charge des porteurs minoritaires en excès est définie par :

$$Q_D = \tau . I$$

Une jonction polarisée en direct est le siège d'une répartition de charge, ce phénomène est représenté par une capacité de diffusion E-B (C_{DE}) et C-B (C_{DC}), fonction du courant de la jonction.

$$C_D = \tau . G$$

La conductance dynamique de la jonction est donnée par:

$$G = dI / dV$$

V est la tension appliquée à la jonction

On obtient alors :

Pour la jonction E-B : $C_{DE} = \tau_e \ (q/kT) \ I_{es} \exp(qV_{be}/kT)$ (I.5a)

Pour la jonction C-B : $C_{DC} = \tau_c \ (q/kT) \ I_{cs} \exp(qV_{bc}/kT)$ (I.5b)

Quand une jonction est polarisée en inverse, il y a création d'une zone fortement dépeuplée de porteurs libres dite zone de transition de part et d'autre de la jonction. Ce phénomène est représenté par une capacité de transition qui dépend alors de la tension appliquée aux bornes de la jonction.

$$C_T = \frac{C_O}{\sqrt{1 - \frac{V}{\phi}}}$$

C_O représente la valeur de C_T à V= 0 Volt

V la tension appliquée aux bornes de la jonction

ϕ barrière de potentiel de la jonction

Nous admettons que pour des tensions V supérieures à la barrière de potentiel ϕ, la capacité de transition est constante.

Pour la jonction E-B :

$$C_{TE} = \frac{C_{OE}}{\sqrt{1 - \frac{Vbe}{\phi}}} \qquad (I.6a)$$

Pour la jonction C-B :

$$C_{TC} = \frac{C_{OC}}{\sqrt{1 - \frac{Vbc}{\phi}}} \qquad (I.6b)$$

A partir des expressions des courants définies précédemment ainsi que celles des capacités, nous pouvons représenter le schéma équivalent du transistor bipolaire constitué de deux générateurs de courant et des capacités de diffusion et de transition en parallèle sur deux diodes (voir figure I.1).

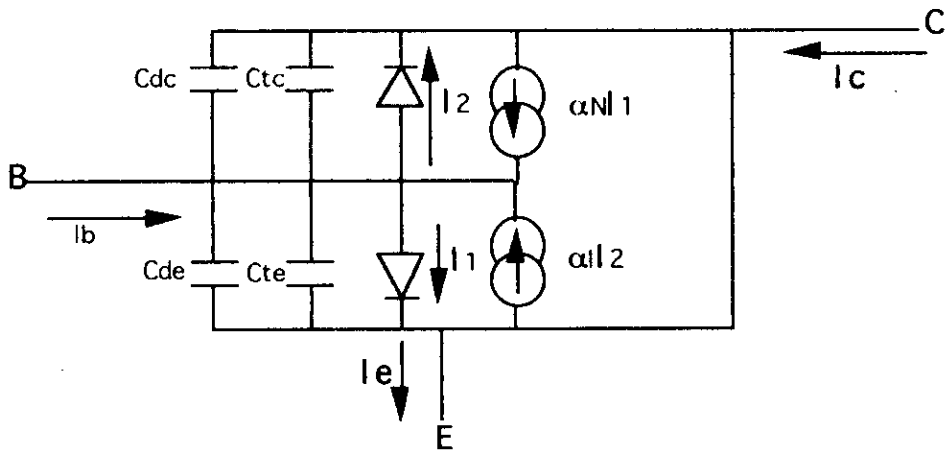


Figure I.1 Schéma équivalent complet du modèle d'Ebers-Moll

I.2.2 Méthode de détermination des paramètres du modèle d'Ebers-Moll

Parmi les problèmes posés par l'utilisation des programmes d'analyse de circuits, il en est un particulièrement délicat qui est la détermination des éléments relatifs au schéma équivalent. Les différents paramètres du modèle d'Ebers-Moll à extraire sont alors:

I_{CS}, I_{ES}, α_N, α_I, C_{DE}, C_{DC}, C_{TE} et C_{TC}

a) Mesure des paramètres statiques I_{CS}, I_{ES}, α_N, α_I

Le groupe des quatre paramètres statiques I_{CS}, I_{ES}, α_N et α_I permet de définir les caractéristiques du dispositif et sert de support à l'ensemble du modèle. Afin de les déterminer, un équipement minimal est nécessaire (un oscilloscope ou un transistormètre qui permet d'obtenir les caractéristiques statiques [31]).

Les paramètres α_N et α_I sont calculés à partir de la mesure des gains β_N et β_I puis extraits des relations (I.2a) et (I.2b).

Les mesures se font autour du point de fonctionnement choisi comme l'indique la figure I.2.

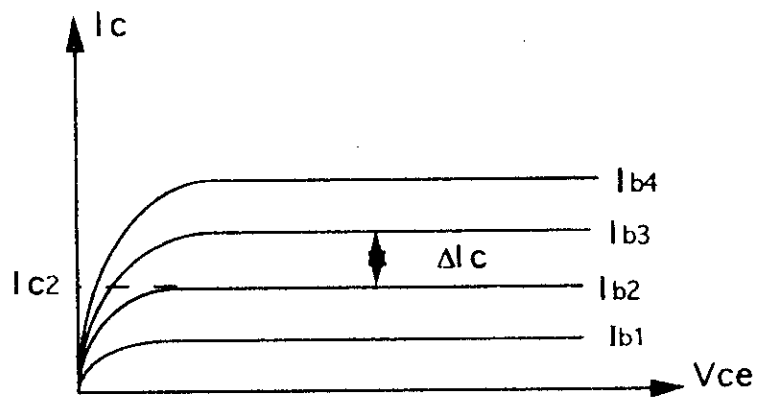


Figure I.2 Mesure de β_N et de β_I

La valeur appropriée de la constante β_N peut être déterminée à partir du traceur de courbes du courant collecteur en fonction de la tension collecteur-émetteur pour un courant de base fixé I_B fixé:

$$\beta_N = \frac{I_{c2}}{I_{b2}}$$

On notera que c'est la valeur continue qui est utilisée.

La même méthode de mesure est effectuée pour le gain inverse, en interchangeant le rôle de l'émetteur et du collecteur. Grâce à la relation de réciprocité (I.3) définie précédemment, les paramètres I_{es} et I_{cs} peuvent être aisément calculés une fois le courant de saturation I_s déterminé.

En ce qui concerne la mesure du courant de saturation I_s , on optera pour une méthode de mesure telle que la relation (I.3) soit vérifiée.

A partir des caractéristiques statiques I_c-V_{ce} à $V_{cb} = 0$ Volt, on relève la courbe $I_c = f(V_{be})$ avec $V_{be} = V_{ce}$ (la base et le collecteur étant reliés).

Pour $V_{be} = 0$ Volt, on relève la courbe $I_E = f(V_{bc})$, avec $V_{bc} = V_{ce}$ (la base et l'émetteur reliés). En prenant $V_{bc} = V_{be}$ on obtient pratiquement I_c il en découle:

$$\alpha_N I_{es} = \alpha_I I_{cs} = I_s$$

vu que :

$$I_C = - \alpha_N I_{es} [\exp (qV_{be}/kT)]$$

$$I_E = - \alpha_I I_{cs} [\exp (qV_{bc}/kT)]$$

D'aprés la courbe $I_C = f(V_{be})$, on peut tracer la courbe Logarithmique $Ln\ I_C = f(V_{be})$ afin de déduire la valeur de Is par une extrapolation graphique de la courbe à $Vbe=0$ (voir Fig.I.3).

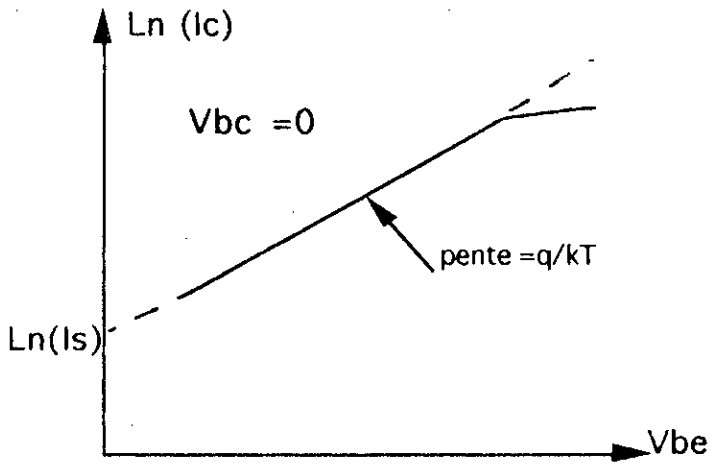


Figure I. 3. Determination de Is par extrapolation graphique (Ic = Is exp(Vbe/UT))

b) Mesure des paramètres dynamiques CDE, CDC, CTE et CTC.

Les paramètres dynamiques de transitions varient en fonction du potentiel appliqué à la jonction, selon la courbe donnée en figure I .4a. Leur mesure s'effectue sur une jonction polarisée en inverse mais peut se faire sur une jonction polarisée en direct jusqu'à des tensions égales à *0,4* Volts seulement car le courant débité est trés faible et la capacité de diffusion est négligeable.

La mesure nécessite un pont d'impédance [32] qui dispose de la possibilité d'appliquer une polarisation externe continue sur le composant à mesurer (voir fig.I.4b).

La courbe *C = f(V)* pourra être tracée et la valeur de la capacité qui correspond au point de fonctionnement considéré relevée .

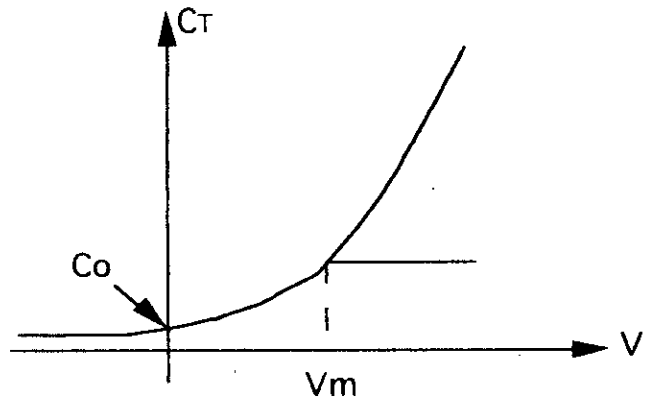


Figure I. 4a Variation de la capacité de transition (pour V>Vm,CT=constante)

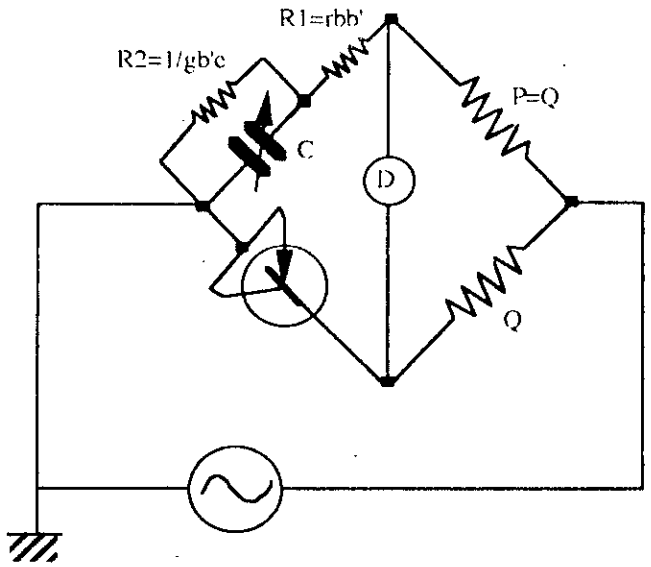


Figure I.4b Mesure de la capacité de transition au pont d'impédance

La mesure des capacités de diffusion n'étant pas possible directement, on passe par l'intermédiaire des constantes de temps de contrôle de charge τ_{cf} et τ_{cr} et de la relation qui lie $C_{D/c}$ et τ_{cf} d'une part et $C_{D/c}$ et τ_{cr} d'autre part.

Le montage utilisé [33] est représenté en figure I.5.

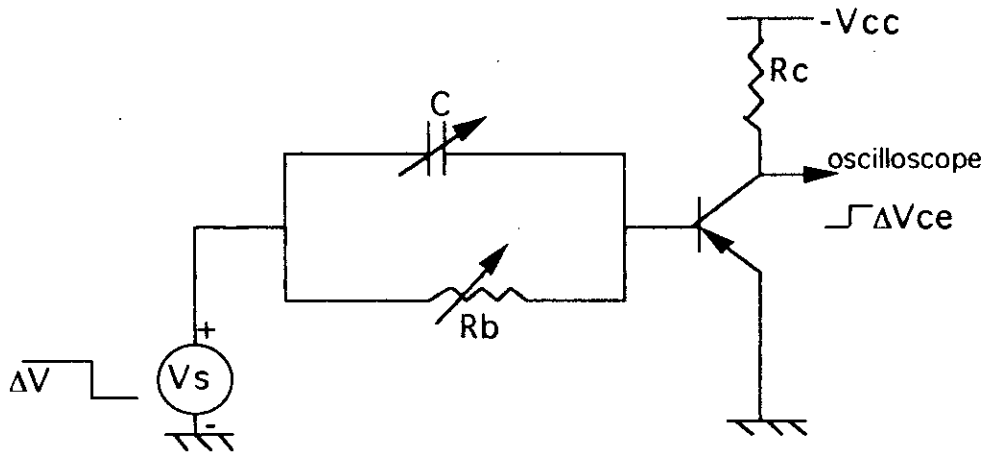


Figure I.5 Montage de mesure des constantes de temps de contrôle de charge τ_{cf} et τ_{cr}

La méthode consiste à injecter dans la base une charge de porteurs majoritaires Q_b au moyen d'un saut de tension ΔV, par la capacité C tout en imposant par R_b un saut de

courant qui tend à décroitre selon le phénomène de recombinaison avec une constante de temps τ_b.

Si l'on injecte une charge Q_b au travers du circuit RC à la vitesse même à laquelle elle décroit par recombinaison, on maintiendra le saut de courant imposé. On peut alors écrire [32]:

$$Ib = \frac{Qb}{\tau b} = \frac{C\ \Delta V}{\tau_b} = \frac{\Delta V}{R_b} \qquad (I.7a)$$

d'où

$$\tau_b = R_b.\ C \qquad (I.7b)$$

Le saut de courant Ic qui résulte est relié à la charge Qb par la relation suivante:

$$I_c = \frac{Q_b}{\tau_c} \qquad (I.7c)$$

Or nous avons:

$$Ic = \frac{\Delta V_{ce}}{R_c} \qquad (I.7d)$$

ce qui implique

$$\tau_c = \frac{C\ \Delta V}{\Delta V_{ce}} R_c \qquad (I.7e)$$

Puisque

$$\tau_b = R_b\ C$$

on peut écrire:

$$\tau_c = \frac{\Delta V\ R_c}{\Delta V_{ce}\ R_b} \tau_b \qquad (I.7f)$$

Ainsi , on peut déduire soit en sens direct $\tau_{cf} = \tau_f$, soit en sens inverse $\tau_{cr} = \tau_r$, les deux constantes de temps et remonter par la suite aux deux paramètres capacitifs C_{DE} et C_{DC} par les relations (I.5a) et (I.5b).

Le tableau 1 regroupe tous les paramètres du modèle d'Ebers-Moll ainsi que leurs valeurs respectives mesurées sur le transistor 2N2222 de Motorola.

Paramètres	Définition	Valeur 2N2222 de Motorola
α_N	Gain en courant direct du montage base-commune	0.99
α_I	Gain en courant inverse du montage base-commune	0.77
Ies (A)	Courant de saturation relatif à la diode D1	$0.37\ 10^{-13}$
Ics (A)	Courant de saturation relatif à la diode D2	$0.67\ 10^{-10}$
τf (μs)	Constante de temps relatif à la diode D1	$0.67\ 10^{-3}$
τr (μs)	Constante de temps relatif à la diode D2	0.05
Coe (pF)	Capacité de transition de la jonction E-B pour la tension V = 0 Volt	24
Coc (pF)	Capacité de transition de la jonction B-C pour la tension V = 0 Volt	13

Tableau 1. Paramètres mesurés du transistor 2N2222 de Motorola

Nous voyons donc qu'en général, le modèle d'Ebers-Moll ou ses dérivés [18-31] font appel à un nombre de paramètres assez important dont la détermination nécessite soit des mesures électriques précises soit de simples calculs.

Un programme du genre NETTRAN simplifie énormément le problème puisqu'il effectue directement une sorte de statistique ainsi que les calculs nécessaires. Il fait partie d'un ensemble de programmes destinés à fournir tous les paramètres utiles aux programmes d'analyse tels que le programme NET1.

1.3 VALIDATION DU MODELE D'EBERS-MOLL

Les tendances actuelles font que les essais soient de plus en plus abandonnés au profit du calcul par des méthodes de prévision grâce à des programmes conçus dans ce but.

Cependant afin d'assurer à l'utilisateur la validité de ceux-ci, on doit pouvoir en confronter le résultat avec l'expérience et pour cela on étudiera un montage inverseur simple, qui est à la base de toute porte logique.

Afin de vérifier l'exactitude du modèle choisi, on comparera la réponse calculée par programme en utilisant l'algorithme qui va suivre avec la réponse donnée par l'expérience.

I.3.1 Modèle d'Ebers-Moll numérisé

Afin de pouvoir simuler le comportement de l'inverseur, nous eu avons recours au modèle numérique associé au modèle d'Ebers-Moll. Pour cela nous avons considéré le schéma équivalent du modèle numérisé de la diode [3-4], ainsi que celui de la capacité (voir Annexe 2).

La linéarisation de la caractéristique de la diode donne le circuit équivalent suivant :

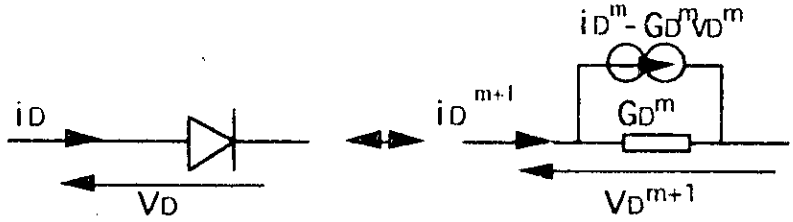


Figure I.7 Schéma équivalent du modèle numérisé de la diode

L'approximation de la dérivée donne le schéma équivalent de la capacité :

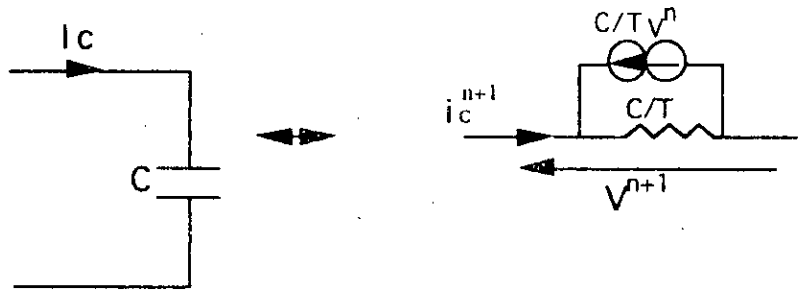


Figure I.8 Schéma équivalent du modèle numérisé de la capacité

Nous avons numérisé la diode ainsi que la capacité vu que le schéma équivalent du modèle d'Ebers-Moll est constitué de ces deux éléments.

Les courants seront alors discrétisés:

$$I_b^{n+1,m+1} = I_1^{n+1,m+1} + I_2^{n+1,m+1} + (CTE^n + CDE^n)\frac{Vbe^{n+1,m+1} - Vbe^n}{T}$$

$$I_c^{n+1,m+1}=\beta n\, I_1^{n+1,m+1}-(\beta i+1)\, I_2^{n+1,m+1}+(C_{TC}{}^{n}+C_{TE}{}^{n})\,\frac{Vbc^{n+1,m+1}-Vbc^{n}}{T}$$

Avec:

$$I_1^{n+1,m+1}=I_1^{n+1,m}-G_1{}^{n+1,m}\,Vbc^{n+1,m}+G_1{}^{n+1,m}\,Vbc^{n+1,m+1}$$
$$I_2^{n+1,m+1}=I_2^{n+1,m}-G_2{}^{n+1,m}\,Vbc^{n+1,m}+G_2{}^{n+1,m}\,Vbc^{n+1,m+1}$$

n : représente le pas de discrétisation

m : représente le nombre d'itérations

Les capacités de transition étant fonction des tensions appliquées et les capacités de diffusion fonction des courants de diodes, une valeur de capacité sera calculée à chaque pas (pour une tension et un courant donnés) et de ce fait, sera considérée constante sur ce pas.

$$C_T{}^{n}=\frac{C_0}{\sqrt{1-\frac{V^n}{\phi}}}\quad\text{et}\qquad C_D{}^{n}=\tau\; q/kT\; I_S\exp(qV^n/kT)$$

Une fois ce travail effectué, le modèle numérique d'Ebers-Moll sera prêt à l'application choisie.

Dans le paragraphe qui suivra, le modèle d'Ebers-Moll numérisé (voir fig.I.10a) sera appliqué à l'inverseur, les algorithmes traduisant son comportement en régime de commutation seront formulés et l'organigramme ainsi que le programme de calcul seront ensuite établis.

I.3.2 Simulation de l'inverseur

Le montage dont nous allons simuler le comportement en régime transitoire est celui d'un transistor bipolaire NPN, monté en émetteur-commun [30] (voir figure I.9).

Pour cela les expressions des courants s'écrivent conformément à la configuration émetteur-commun (Annexe 1):

$$I_b=I_{bs1}\left[\exp\left(\frac{Vbe}{U_T}\right)-1\right]+I_{bs2}\left[\exp\left(\frac{Vce}{U_T}\right)-1\right]\qquad(I.8a)$$

$$I_c=\beta n\, I_{bs1}\left[\exp\left(\frac{Vbc}{U_T}\right)-1\right]-(bi+1)I_{bs2}\left[\exp\left(\frac{Vce}{U_T}\right)-1\right]\qquad(I.8b)$$

Avec: $I_{bs1} = \frac{I_s}{\beta_N}$ $I_{bs2} = \frac{I_s}{\beta_I}$

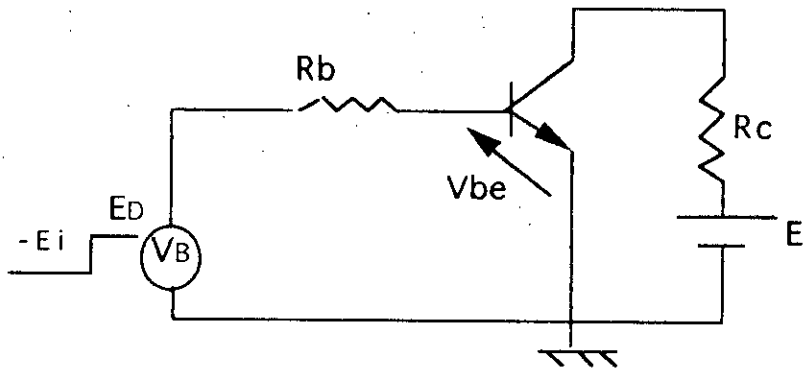


Figure I.9 Schéma du montage inverseur à simuler

Nous supposerons que le transistor se trouve initialement à l'état bloqué par la tension $-E_i$. On applique sur la base un échelon de tension E_D et nous allons suivre les changements qui vont s'opérer sur la tension V_{be} entre la base et l'émetteur et sur le courant de collecteur I_c.

Le régime transitoire peut être décomposé en trois phases décrites chacune par un système d'équations.

- ***1ère phase :***

La tension directe appliquée va dans un premier temps t_d, appelé temps de retard, amener la charge d'espace E-B à diminuer et la capacité de la jonction à se charger jusqu'à ce que la tension V_{be} aux bornes de la jonction passe de $-E_i$ à la tension équivalente de déblocage V_D.

Pendant cette phase, le courant Ic trés faible ne varie pratiquement pas. Les courants de saturation inverses I_{S1} et I_{S2} (de l'ordre de 10^{-13} A pour le transistor au Silicium) sont trés faibles devant les courants de charge des capacités. Ce qui réduit le schéma donné en figure I.10a au schéma plus simplifié de la figure I.10b.

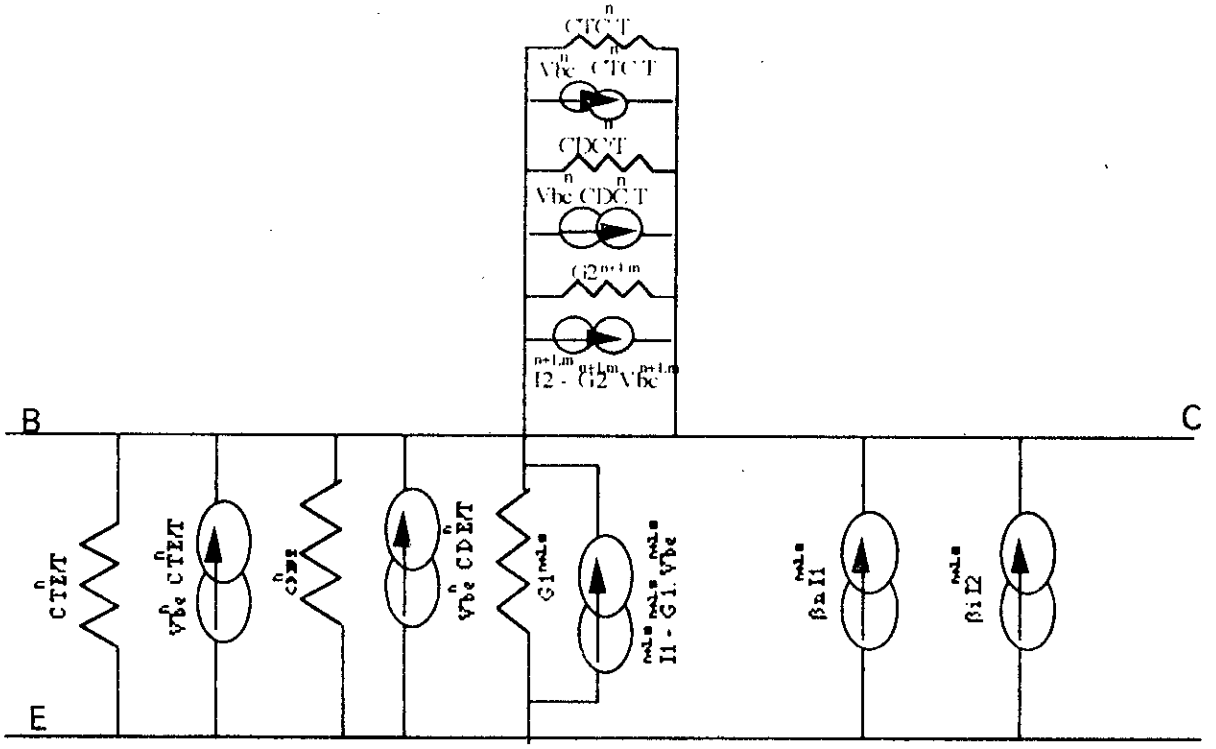


Figure I.10a Modèle complet d'Ebers-Moll numérisé

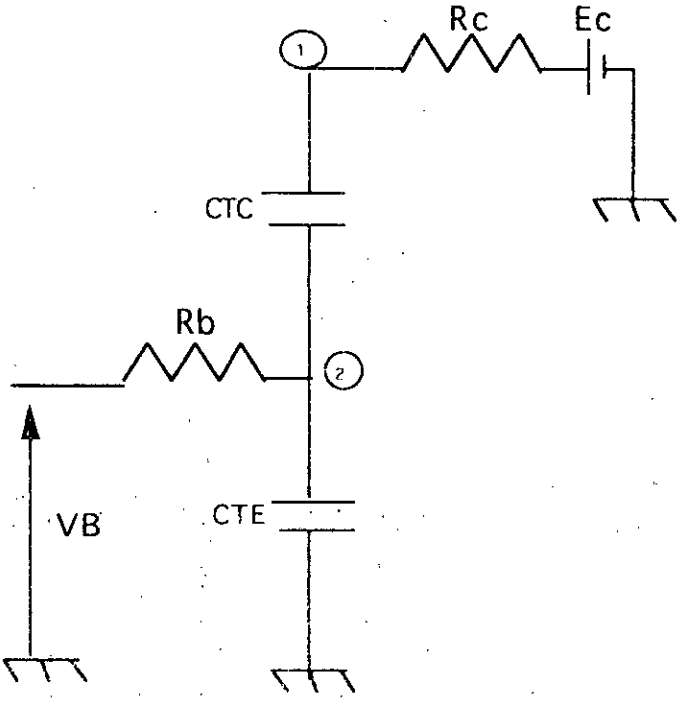


Figure I.10b Modèle d'Ebers-Moll numérisé correspondant à la phase 1 (simulation du temps de montée)

On définit deux noeuds 1et 2, V_1 sera la tension Vce et V_2 la tension Vbe. Ainsi Vbc sera égale à $V_2 - V_1$.

$$\text{Noeud 1} \rightarrow \frac{E-V_1}{Rc} = C_{TC}\frac{d(V_1-V_2)}{dt}$$

$$\text{Noeud 2} \rightarrow \frac{V_B-V_2}{R_B} = C_{TE}\frac{dV_2}{dt} - \frac{E-V_1}{Rc}$$

En discrétisant le système:

$$\frac{E-V_1^{n+1}}{Rc} = C_{TC}\frac{(V_1^{n+1} - V_2^{n+1}) - (V_1^{n} - V_2^{n})}{T}$$

$$\frac{V_B-V_2^{n+1}}{R_B} + \frac{E-V_1^{n+1}}{Rc} = C_{TE}\frac{V_2^{n+1} - V_2^{n}}{T}$$

Que l'on peut mettre sous la forme matricielle suivante:

$$\begin{bmatrix} \frac{1}{Rc}+\frac{CTC^n}{T} & \frac{CTC^n}{T} \\ \frac{1}{Rc} & \frac{1}{Rb}+\frac{CTE^n}{T} \end{bmatrix} \begin{bmatrix} V1^{n+1,m+1} \\ V2^{n+1,m+1} \end{bmatrix} = \begin{bmatrix} \frac{E}{Rc} + \frac{C_{TC}^n}{T} V_1^n - \frac{C_{TC}^n}{T} V_2^n \\ \frac{Vb}{Rb} + \frac{E}{Rc} - \frac{C_{TE}^n}{T} V_2^n \end{bmatrix}$$

Nous avons donc à résoudre un système d'équations linéaires $A.X = B$

Les capacités C_{DE} et C_{DC} sont négligeables devant les capacités C_{TC} et C_{TE}.

ALGORITHME:

$$\begin{bmatrix} \frac{1}{Rc}+\frac{CTC^n}{T} & \frac{CTC^n}{T} \\ \frac{1}{Rc} & \frac{1}{Rb}+\frac{CTE^n}{T} \end{bmatrix} \begin{bmatrix} V1^{n+1,m+1} \\ V2^{n+1,m+1} \end{bmatrix} = \begin{bmatrix} \frac{E}{Rc} + \frac{C_{TC}^n}{T} V_1^n - \frac{C_{TC}^n}{T} V_2^n \\ \frac{Vb}{Rb} + \frac{E}{Rc} - \frac{C_{TE}^n}{T} V_2^n \end{bmatrix}$$

$$C_{TC}^{\,n} = \frac{C_0}{\sqrt{1 - \frac{V_2^n - V_1^n}{\phi}}} \quad ; \quad C_{TE}^{\,n} = \frac{C_0}{\sqrt{1 - \frac{V_2^n}{\phi}}}$$

$n = 0, 1, 2, \ldots\ldots, n_{max}$

La valeur de n_{max} est obtenue quand $V_2^{\,nmax+1} \geq 0$.

Processus de résolution:

- Pour t = T c'est à dire n = 0 (premier pas), nous donnons les valeurs des tensions initiales du collecteur et de la base, les valeurs de $C_{TC}{}^{0}$ et $C_{TE}{}^{0}$ sont ensuite calculées. Par la résolution du système AX =B, on obtient les tensions $V_1{}^1$ et $V_2{}^1$
- Pour t = 2T , n = 1, on prend comme valeurs initiales les valeurs précédemment calculées $V_1{}^1$ et $V_2{}^1$ $C_{TC}{}^1$ et $C_{TE}{}^1$ sont calculées. Les solutions seront : $V_1{}^2$ et $V_2{}^2$
- .....
- .....
- .....
- A $t = (n_{max} + 1).T$, $V_2{}^{nmax+1} \geq 0$ le calcul s'arrête et la durée de cette phase sera donnée par td, tel que :

$$n_{max}.T \leq t_d \leq (n_{max}+1).T$$

- ***2ième phase :***

A partir de l'instant où $V_{eb} = V_D$, la jonction E-B va être polarisée en direct et injecter du courant dans la base. Il va y avoir effet transistor et le courant de collecteur I_c va croître.

On mesurera un second temps : t_r qui est le temps de montée du courant de collecteur avant qu'il n'atteigne sa valeur maximale.

- ***3ième phase :***

Suivant l'amplitude de E_D, on atteindra ou non les conditions de saturation. Dans le cas positif, le courant de collecteur I_c sera égal à E/R_C ; il aura sa valeur maximale I_{csat} dite de saturation.

A cet instant la relation $I_{csat} = \beta I_b$ resta applicable. Si I_b continue à croitre jusqu'à la valeur I_{BD}, le transistor entre en régime de saturation :

$$\beta I_{BD}/I_{csat} = E/R_c$$

Le transistor reste ensuite aussi longtemps dans l'état saturé que le courant I_{BD} de base soit maintenu. Afin de décrire ces deux phases (actif et saturé), le modèle complet d'Ebers-Moll sera considéré.

Les expressions des courants déduits à partir de la loi d'Ohm relative aux noeuds seront les suivantes:

$$I_b = I_{bs1}\left[\exp\left(\frac{Vbe}{U_T}\right)-1\right]+I_{bs2}\left[\exp\left(\frac{Vce}{U_T}\right)-1\right]+(C_{TE}+C_{DE})\frac{dVbe}{dt}$$

$$I_c = \beta n\, I_{bs1}\left[\exp\left(\frac{Vbe}{U_T}\right)-1\right]-(bi+1)I_{bs2}\left[\exp\left(\frac{Vce}{U_T}\right)-1\right]+(C_{TC}+C_{DC})\frac{dVbc}{dt}$$

On définit deux noeuds 1 et 2 :

V_1 sera la tension Vbe et V_2 la tension Vce ainsi Vbc = V_1 - V_2 .

V_B étant l'échelon de tension (-Ei,E_D) appliqué à l'entrée.

La loi de Kirchoff appliquée aux noeuds 1 et 2 nous donne le système suivant :

$$\frac{V_B - V_1}{Rb} = I_1 + I_2 + (C_{TE}+C_{DE})\frac{dV_1}{dt} + (C_{TC}+C_{DC})\frac{d}{dt}(V_1 - V_2) \qquad (I.9a)$$

$$\frac{E-V_2}{Rc} = \beta n\, I_1 - (\beta i+1)\, I_2 - (C_{TC}+C_{DC})\frac{d}{dt}(V_1 - V_2) \qquad (I.9b)$$

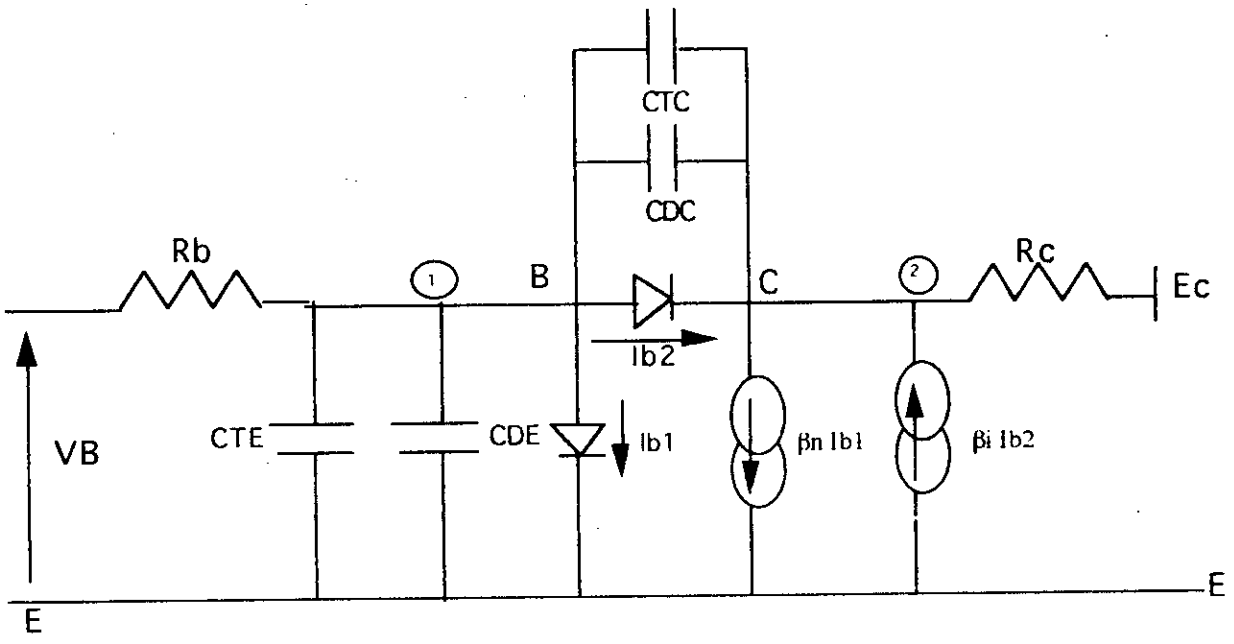


Figure I.10c Modèle d'Ebers-Moll numérisé correspondant à la phase 3 (simulation du temps de descente)

Aprés discrétisation et linéarisation, le système formé des équations (I.9a) et (I.9b) devient :

$$\frac{(V_B - V1^{n+1,m+1})}{Rb} = I_1^{n+1,m+1} + I_2^{n+1,m+1} + \frac{(C_{TE}^n + C_{DE}^n)}{T}(V_1^{n+1,m+1} - V_1^{n+1,m}) + \frac{(C_{TC}^n + C_{DC}^n)}{T}[(V_1^{n+1,m+1} - V_2^{n+1,m+1}) - (V_1^{n+1,m} - V_2^{n+1,m})]$$

$$\frac{E - V_2^{n+1,m+1}}{Rc} = \beta n\, I_1^{n+1,m+1} - (\beta i+1)\, I_2^{n+1,m+1} - \frac{(C_{TC}^n + C_{DC}^n)}{T}[(V_1^{n+1,m+1} - V_2^{n+1,m+1}) - (V_1^{n+1,m} - V_2^{n+1,m})]$$

En ramenant le système d'équations sous la forme AX =B avec :

- A une matrice carrée 2 x 2
- X le vecteur des tensions inconnues
- B le vecteur second membre

On arrive à l'écriture matricielle suivante :

$$\begin{bmatrix} \frac{1}{Rb} G_1^{n+1,m} + \frac{C_{TC}^n + C_{DC}^n}{T} - G_2^{n+1,m} & \frac{C_{TC}^n + C_{DC}^n}{T} + G_2^{n+1,m} \\ -\beta n + G_1^{n+1,m} + (\beta i+1) G_2^{n+1,m} + \frac{C_{TC}^n + C_{DC}^n}{T} & -\frac{1}{Rc}(\beta i+1) - \frac{C_{TC}^n + C_{DC}^n}{T} \end{bmatrix} \begin{bmatrix} V_1^{n+1,m+1} \\ V_2^{n+1,m+1} \end{bmatrix} =$$

$$\frac{V_B}{Rb} + I_1^{n+1,m} + I_2^{n+1,m} - V_1^{n+1,m}\left(G_1^{n+1,m} + \frac{C_{TE}^n + C_{DE}^n + C_{TC}^n + C_{DC}^n}{T}\right) + V_2^{n+1,m}\left(G_2^{n+1,m} + \frac{C_{TC}^n + C_{DC}^n}{T}\right)$$

$$-\frac{E}{Rc} + \beta n I_1^{n+1,m} - (\beta i+1) I_2^{n+1,m} - V_1^{n+1,m}\left(\beta n G_1^{n+1,m} - (\beta i+1) G_2^{n+1,m} + \frac{C_{TC}^n + C_{DC}^n}{T}\right) - V_2^{n+1,m}(\beta i+1) G_2^{n+1,m}$$

Processus de résolution:

Nous supposons que les tensions aux bornes des différentes capacités sont V_1 et V_2 donnés à t = 0, de même que les valeurs de ces capacités puisqu'elles sont fonctions de ces tensions.

- Premier pas n = 1

Nous posons pour m = 0 (pas d'itérations)

$V_2^0 = V_2^{1,0}$ et $V_1^0 = V_1^{1,0}$; en effet nous supposerons que lorsqu'aucune des itérations n'a encore été effectuée, les conditions initiales existent toujours. A partir de $V_1^{1,0}$ nous pouvons calculer :

En posant : $\lambda = 1/U_T$

$$I_1^{1,0} = Ibs1\,[exp(\lambda V_1^{1,0}) - 1]$$

et en déduire : $$G_1^{1,0} = \lambda\, Ibs1\, exp(\lambda V_1^{1,0})$$

de même qu'à partir de $V_2^{1,0}$ on a :

$$I_2^{1,0} = Ibs2\,[exp(\lambda (V_1^{1,0} - V_2^{1,0}) - 1]$$

et en déduire: $$G_2^{1,0} = \lambda\, Ibs1\, exp(\lambda (V_1^{1,0} - V_2^{1,0}))$$

Ceci nous donne toutes les informations nécessaires pour pouvoir calculer les tensions aux noeuds $V_2^{1,1}$ et $V_1^{1,1}$.

Parallèlement, les valeurs des capacités de diffusion et de transition sont aussi calculées :

$$C_{DE} = \tau_b \cdot G_2^{1,0} \quad ; \quad C_{DC} = \tau_b \cdot G_1^{1,0}$$

$$C_{TE} = \frac{C_0}{\sqrt{1 - \frac{V_2^{1,0}}{\phi}}} \quad ; \quad C_{TC} = \frac{C_0}{\sqrt{1 - \frac{V_2^{1,0} - V_1^{1,0}}{\phi}}}$$

En résolvant le premier système, nous obtenons les solutions $V_2^{1,1}$ et $V_1^{1,1}$.

- Pour m = 1 :

$I_2^{1,1}$ et $I_1^{1,1}$ sont calculés à partir de $V_2^{1,1}$ et $V_1^{1,1}$ ainsi que $G_2^{1,1}$ et $G_1^{1,1}$ et ainsi de suite l'itération est effectuée de cette manière jusqu'à ce que l'écart entre deux tensions $V_1^{1,mf}$ et $V_1^{1,mf-1}$ soit assez faible pour que l'on puise conclure que la solution a convergé.

On établit donc à chaque fois qu'une nouvelle tension V_1 est calculée le test suivant :

$$V_1^{1,mf} - V_1^{1,mf-1} \leq \varepsilon$$

Le nombre ε fixe la précision que l'on désire obtenir et mf désigne le nombre d'itérations final.

- Second pas n = 2

On pose $V_2^{1,mf} = V_2^{2,0}$ et $V_1^{1,mf} = V_1^{2,0}$; m = 0, 1, 2, ..., mf

ensuite

$V_1^{2,1} = f\ (\ V_1^{2,0},\ V_2^{2,0},\ I_1^{2,0},\ I_2^{2,0},\ G_1^{2,0},\ G_2^{2,0},\ C_{TC}{}^{2,0},\ C_{TE}{}^{2,0},\ C_{DC}{}^{2,0},\ C_{DE}{}^{2,0})$

$V_1^{2,2} = f\ (\ V_1^{2,1},\ V_2^{2,1},\ I_1^{2,1},\ I_2^{2,1},\ G_1^{2,1},\ G_2^{2,1},\ C_{TC}{}^{2,1},\ C_{TE}{}^{2,1},\ C_{DC}{}^{2,1},\ C_{DE}{}^{2,1})$

. . . . . .

$V_1^{2,mf} = f(V_1^{2,mf-1}, V_2^{2,mf-1}, I_1^{2,mf-1}, I_2^{2,mf-1}, G_1^{2,mf-1}, G_2^{2,mf-1}, C_{TC}{}^{2,mf-1}, C_{TE}{}^{2,mf-1}$

$, C_{DC}{}^{2,mf-1}, C_{DE}{}^{2,mf-1})$

Quand mf est atteint : $V_1^{2,mf} - V_1^{2,mf-1} \leq \varepsilon$

- Le même processus est répété pour $n = 3,4,5....,nmax.$

Le régime normal n'étant qu'un état intermédiaire entre le régime bloqué et le régime saturé, il possède une durée déterminée qui est définie quand la tension de la base devient polarisée en direct.On devra donc introduire une condition pour déterminer la fin du régime normal et connaitre le temps de montée du courant I_C :

$$V_2^{nmax+1,mf} - V_1^{nmax+1,mf} \geq 0 \qquad \text{à} \quad t = (nmax+1).T$$

Si aprés un temps suffisamment long l'échelon appliqué à l'entrée passe de E_D à Ei le processus inverse va s'amorcer:

Dans un premier temps appelé temps de stockage t_s , les charges stockées vont disparaitre par diffusion et par recombinaison jusqu'à ce que l'on atteigne le niveau correspondant à la limite $I_{csat} = \beta\ I_b$; ensuite I_c diminue et V_{be} reste sensiblement égal à V_D, on observe alors un temps de descente t_f .

La tension V_{be} va diminuer et changer de signe, les conditions initiales qui correspondent au blocage de la diode E-B. Le courant I_c ,trés faible, sera celui des conditions initiales.

L'observation du passage de l'état saturé à l'état bloqué se fera par l'application du même système d'équations en phase 2 et 3, le seul changement qu'il y a est au niveau de la tension de commande V qui devient égale à $-Ei$.

Les algorithmes traduisant le comportement transitoire de l'inverseur viennent d'être établis durant les trois phases de son fonctionnement. L'organigramme ainsi que le programme sont donnés en annexe 3.

Aprés avoir défini les paramètres à l'entrée, nous appliquons une impulsion fictive sur la base du transistor monté en émetteur - commun; les réponses du programme sont alors observées sur le terminal graphique TEKTRONIX 4105 .

I.4 Résultats de la simulation

Afin de relever le temps de descente et le temps de montée de l'inverseur numérisé, nous affectons à la tension de commande V_B différentes valeurs (voir le tableau 2) et ceci pour les valeurs de résistances $R_b = 5K\Omega$ et $Rc = 1K\Omega$ et une tension d'alimentation $E_c = 4V$. Les courbes de simulation sont données dans la figure I.11a et la figure I.11b.

V_B (V)	temps de descente (µS)	V_B (V)	temps de montée (µS)
2	25	-1	21
3	15	-2	12
3,5	11	-3	8

Tableau 2 Relevé des temps de descente et de montée de l'inverseur simulé

On effectue par la suite une analyse à travers les résistances de la base*Rb* etdu collecteur*Rc* afin de voir leur influence sur le temps de descente et le temps de montée. On relève alors le tableau 3 suivant :

R b (KΩ) pour Rc = 1KΩ	temps de descente (µS)	Rb = 5 KΩ Rc (KΩ)	temps de montée (µS)
5	11	1	8
10	20	5	26

Tableau 3 Relevé des temps de descente et de montée de l'inverseur simulé pour différentes valeurs des résistances *Rb* de base et *Rc* de collecteur

D'après les résultats obtenus, on peut faire les commentaires suivants :

- Quand V_B diminue le temps de montée augmente ce qui traduit bien le fait que les capacités de transition soient élevées .
- Le temps de descente est d'autant plus court que la tension directe de commande V_B augmente, ce qui est dû à l'influence du courant de la base.
- L'influence des résistances sur les deux temps est bien simulée ce qui est prouvé sur les courbes de la figure I.11c et de la figure I.11d.

Dans ce qui suit nous procédons à la comparaison des temps de montée et de descente simulés avec ceux mesurés sur un transistor bipolaire 2N2222 de Motorola. Le montage considéré est celui donné en figure I.7 avec les valeurs suivantes:

E = 4 V, Rb = 5KΩ et Rc = 1KΩ . On obtient les résultats suivants :

V_B (V)	temps de montée (µs)	V_B (V)	temps de descente (µs)
-1	26	2	21,2
-2	14,5	3	16
-3	8,8	3,5	8,7

Tableau 4 Relevé des temps de descente et de montée de l'inverseur mesurés dans le montage donné en figure I.7

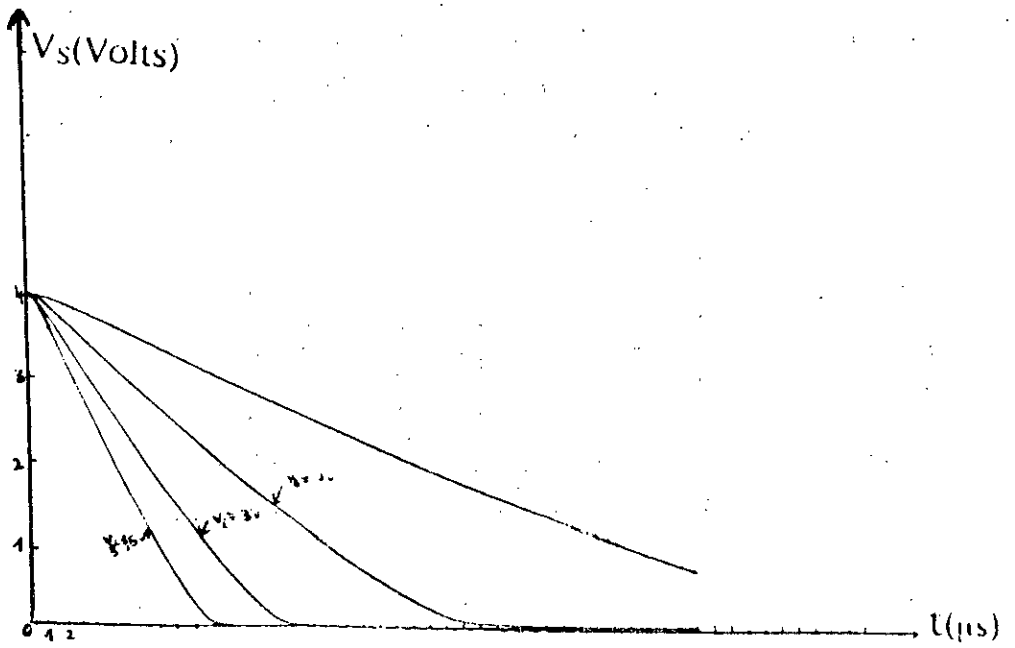


Figure I.11a Courbes de simulation du temps de descente

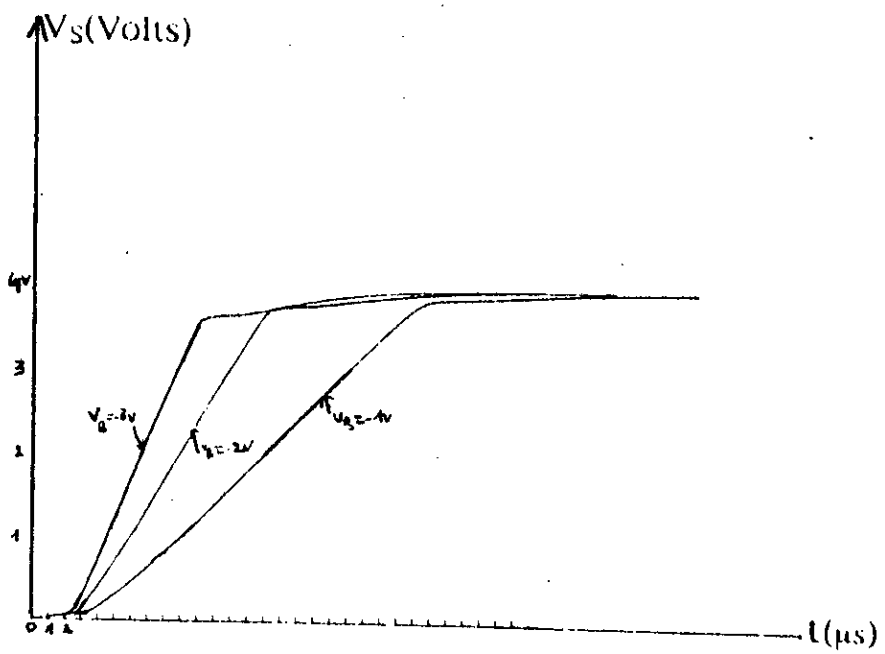


Figure I.11b Courbes de simulation du temps de montée

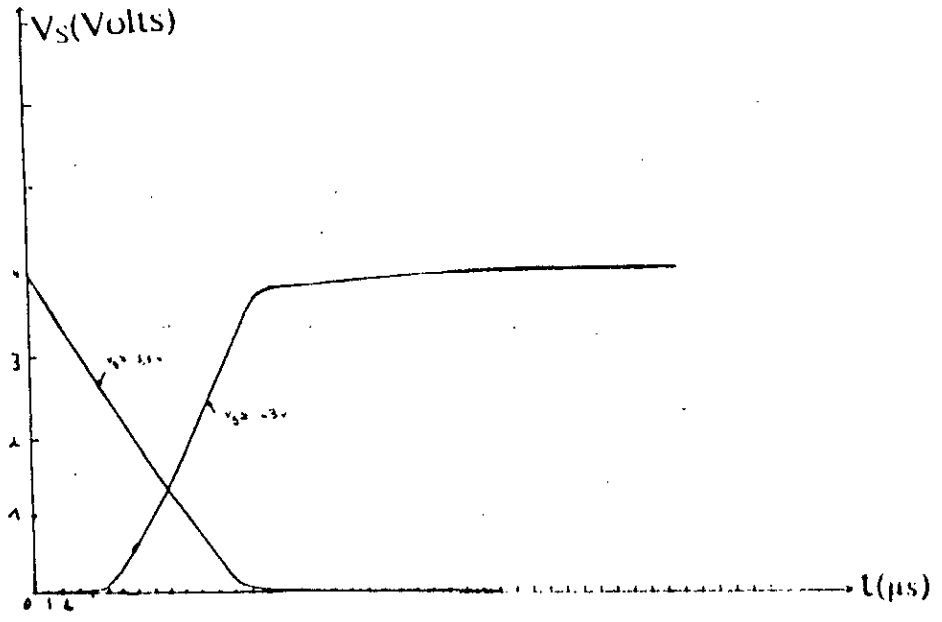


Figure I.11c Simulation de l'influence des résistances de base Rb(=5KΩ) et de collecteur Rc(=1KΩ) sur le temps de descente et le temps de montée

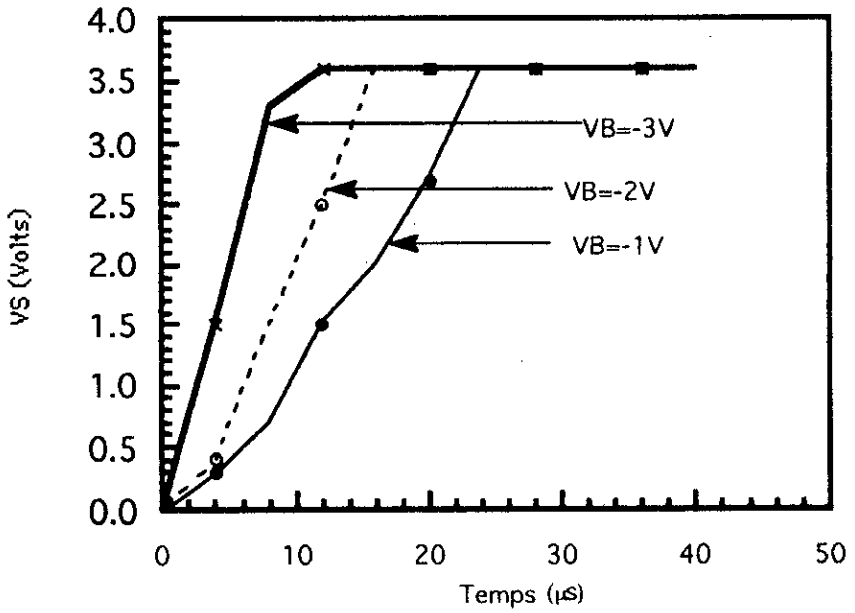


Figure I.12b Mesure des temps de montée d'un transistor bipolaire pour une résistance de base Rb = 5KΩ et une résistance de collecteur Rc=1KΩ

A la lumière des comparaisons entre les temps obtenus expérimentalement (voir figures I.12a et I.12b) et ceux par simulation, on conclut que le modèle numérique associé au modèle d'Ebers-Moll décrit d'une manière assez correcte le sens de variation de la tension de sortie V_{ce} lors de l'application d'une impulsion à l'entrée.

Les différences observées entre les deux résultats expérimentaux et théoriques, sont dues à certains effets physiques qui interviennent dans le fonctionnement du transistor en commutation et dont on n'a pas tenu compte dans notre modèle.

Un apercu des différents modèles physiques et des modèles mathématiques a été fait. Ces modèles mis en oeuvre concernent des objectifs différents.

Les modèles physiques décrivent de façon précise le comportement du transistor bipolaire en tenant compte des effets physiques. Seulement ils s'adaptent mal au calcul vu que la détermination d'une relation entre les paramètres électriques s'avère difficile à obtenir dans la plupart des cas.

Par contre, les modèles mathématiques ne tiennent pas compte de certains mécanismes secondaires complexes à décrire sous forme mathématique, donc manquent de précision mais s'apprêtent bien au calcul surtout.

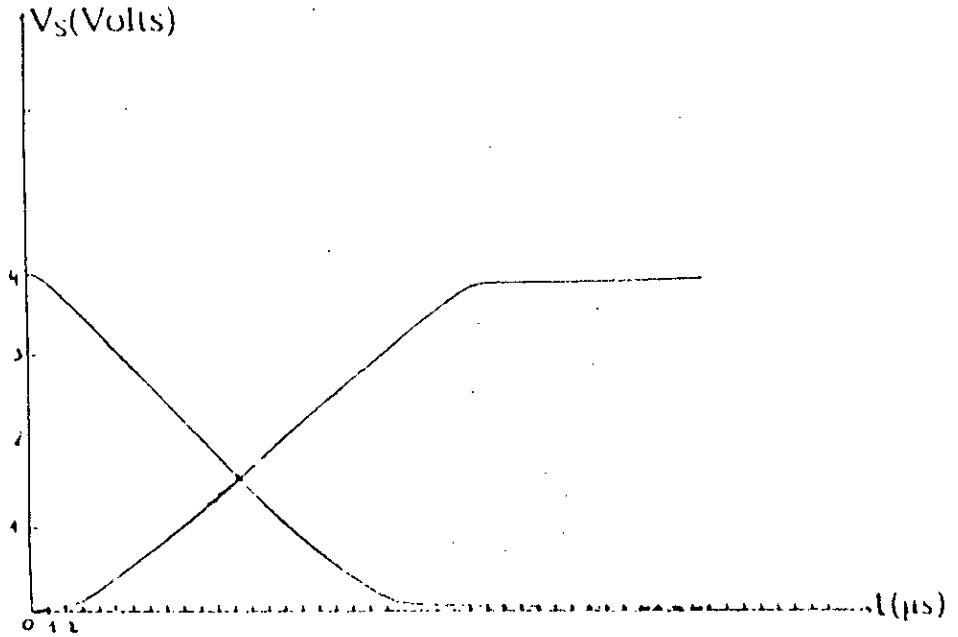


Figure I.11d Simulation de l'influence des résistances de base Rb et de collecteur Rc sur le temps de descente(Rb=5KΩ, Rc=1KΩ) et le temps de montée(Rb=5KΩ, Rc=5KΩ)

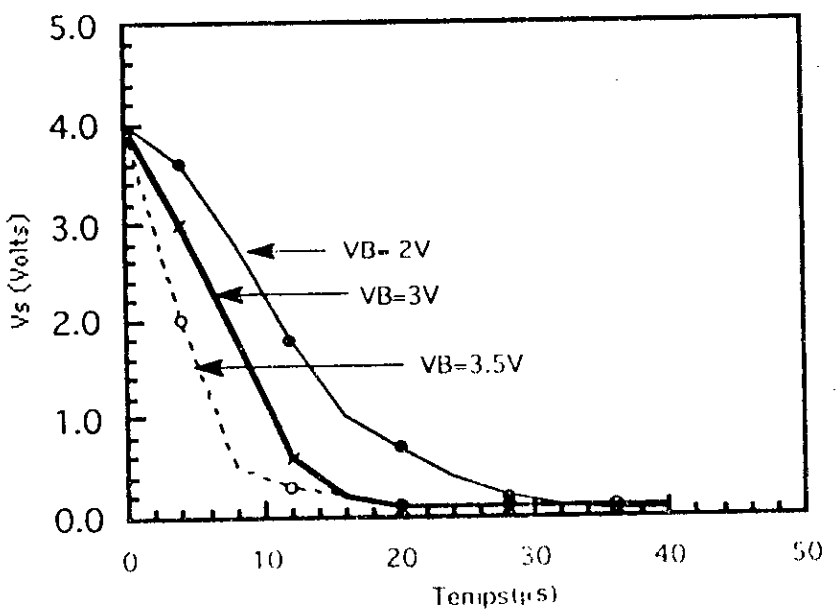


Figure I.12a Mesure des temps descente d'un transistor bipolaire pour une résistance de base Rb = 5KΩ et une résistance de collecteur Rc=1KΩ

A la lumière de cette étude, notre choix s'est porté sur le modèle mathématique d'Ebers-Moll qui est un schéma équivalent "naturel" qui calque la réalité géométrique du dispositif ainsi que son fonctionnement. Plusieurs versions de ce modèle ont été proposées, certaines pouvaient être complexes dans un but de précision, comme dans le cas du modèle adopté dans le programme NET 1 avec 36 paramètres. Par contre, si on se limite à une étude approchée, on peut se contenter d'un nombre de paramètres plus réduit, comme cela a été le cas du modèle adopté dans le programme ECAP [35] avec 10 paramètres.

La version retenue dans notre cas fait intervenir huit paramètres dont quatre statiques et quatre dynamiques pouvant être facilement déterminés.

Afin de valider le modèle, on a considéré le montage inverseur dont on a simulé le comportement en mode de commutation aprés avoir numérisé le modèle d'Ebers-Moll et comparé les temps obtenus à ceux réalisés par l'expérience.

La mise au point du programme d'analyse de circuits nécessite un moyen de formulation des équations pouvant le décrire .

L'exposé des différentes techniques mathématiques de descriptions de circuits ainsi que leur domaine d'application constituera le contenu du prochain chapitre.

CHAPITRE II

METHODES D'ANALYSE DES CIRCUITS ELECTRIQUES

CHAPITRE II

METHODES D'ANALYSE DES CIRCUITS ELECTRIQUES

La simulation des circuits électroniques repose essentiellement sur les techniques de représentation mathématique des circuits à analyser ainsi que sur les moyens de calcul nécessaires à la résolution des systèmes d'équations obtenus.

De ce fait, le présent chapitre sera entièrement consacré à la méthode générale d'acquisition automatique de la configuration qui concerne la topologie du circuit et la formulation des équations qui le décrivent.

Nous aborderons de façon brève l'une des premières méthodes fondamentales d'analyse de circuits qui est l'analyse nodale, vu que tout circuit non-linéaire peut être ramené à un circuit équivalent résistif.

Nous ne tarderons pas dans le développement de la méthode d'analyse par la variable d'état, du fait de la difficulté de l'obtention de la forme normale des équations d'état, ainsi que du nombre minimal de variables impliquées.

L'apparition de la méthode du tableau offre la possibilité de regrouper toutes les variables du circuit en un tableau large et creux, ce qui permettra par la suite l'exploitation de ces deux propriétés dans le processus de résolution.

En fin de chapitre, nous exposerons largement les méthodes classiques d'intégration numériques utilisées par d'autres programmes ainsi que les nouvelles méthodes qui sont venues pallier aux carences des premières.

II.1 ACQUISITION TOPOLOGIQUE

Il s'agit de formuler les équations du circuit en vue d'une exploitation sur machine. Pour cela un moyen simple de décrire un circuit électrique est la notion de graphe et d'arcs orientés [36].

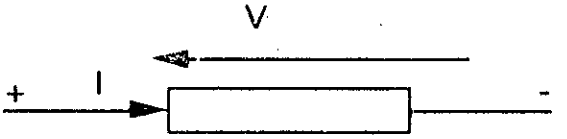
Ce procédé consiste à dessiner un graphe orienté *Gd* associé à un réseau *N* donné et ceci en remplaçant chaque dipôle par un segment de droite appelé branche et qui sera orienté par une flèche.

Ainsi une description complète d'un circuit doit contenir trois informations, à savoir:

- La façon dont sont connectées les branches.
- Les directions de référence des courants.

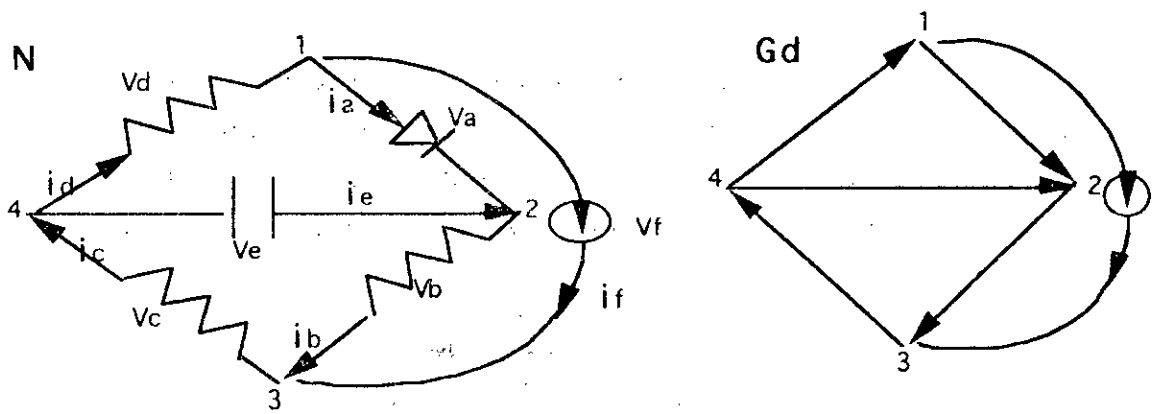
- Les caractéristiques des branches.

Les deux lois de Kirchoff l'une en tension (KVL) et l'autre en courant (KCL) représentent le mode d'interconnexion des branches du circuit dont les éléments sont caractérisées grâce aux relations constitutives. Quant au courant, il est représenté par la flèche orientée dans la même direction que le courant supposé positif à travers la branche selon le schéma ci-dessous :



On définira aussi le concept de noeud et de boucle:

- Un noeud : est donc un point où sont connectées deux ou plusieurs branches.
- Une boucle : c'est un chemin fermé constitué de branches du graphe. L'exemple ci-dessous représente un circuit *N* avec son graphe *Gd* associé.



II.1.1 Matrice d'incidence

Quoique le graphe orienté *Gj* décrit complètement les interconnexions et les directions de référence des branches du circuit , il ne constitue pas une forme convenable pour le stockage en machine. Nous avons recours à un autre moyen où l'information contenue dans le graphe est complètement stockée dans une matrice appelée matrice d'incidence.

Pour un graphe G_d qui possède n noeuds et b branches, on définit la matrice d'incidence $A = [a_{ij}]$ de dimension $n \times b$ où :

- $a_{ij} = 1$ si la branche j est incidente au noeud i et en se déplaçant la flèche est pointée en s'éloignant du noeud i.
- $a_{ij} = -1$ si la branche j est incidente au noeud i et en se déplaçant la flèche est pointée en s'approchant du noeud i.
- $a_{ij} = 0$ si la branche j n'incide pas au noeud i.

Il s'ensuit que chaque colonne contiendra deux composantes a_{ik} et a_{jk} telle que : $a_{ik} = a_{jk} = \pm 1$ les autres éléments étant nuls.

Le courant de branche du circuit N représenté par le vecteur colonne i d'ordre $b \times 1$, la loi de Kirchoff relative aux courants (KCL) appliquée à tous les noeuds peut être exprimée de façon compacte grâce à l'équation matricielle :

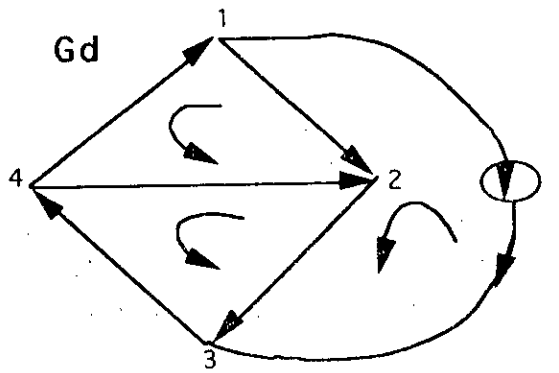
$$A \cdot i = 0 \quad \text{(II.1)}$$

Les équations données par (II.1) ne sont pas linéairement indépendantes, alors que dans l'analyse de circuits il est utile d'avoir des équations indépendantes et toute équation du système matriciel est déterminée à partir des (n-1) équations restantes.

II.1.2 Matrice de circuit

Les équations des mailles ou loi de Kirchoff relative aux tensions (KVL), sont écrites en une seule équation matricielle et introduites par la matrice de circuit B, associée au graphe orienté G_d.

A chaque boucle on assigne une des deux orientations possibles indiquées aussi par une flèche selon le schéma ci-dessous :



Pour un graphe orienté G_d qui possède b branches et n_l boucles orientées, on définit la matrice de circuit $B = (b_{ij})$ d'ordre $(n_l \times b)$ où :

- $b_{ij} = 1$ si la branche j est dans la boucle i et a la même direction que celle-ci.
- $b_{ij} = -1$ si la branche j est dans la boucle i et n'a pas la même direction que celle-ci.
- $b_{ij} = 0$ si la branche j ne se trouve pas dans boucle i.

En se rappelant que les KVL établissent que la somme algébrique des tensions dans une boucle est nulle à tout instant et que les tensions de branches sont représentées par le vecteur colonne *(b x 1)*, on obtient pour toutes les mailles du circuit l'équation matricielle:

$$B.v = 0 \qquad \text{(II.2)}$$

On démontre [36] que la relation suivante:

$$A.B^T = 0 \qquad \text{(II.3)}$$

constitue un résultat général indépendant du choix des boucles qui permet de définir un couplage entre les variables courants et les variables tensions.

II.1.3 Génération automatique des matrices topologiques A et B

La génération de la matrice *A* est assez simple, il suffira d'assigner des nombres entiers consécutifs aux branches du graphe du réseau et les noeuds seront numérotées de la même façon.

Si la branche *k* est connectée entre les noeuds *i* et *j* avec la flèche pointée vers le noeud *j*, on décrit l'information par le triplet *(k,i,j)* alors deux éléments de la matrice *A* sont générés $a_{ik} = 1$ et $a_{jk} = -1$.

La génération de la matrice *B* nécessite la définition du concept d'arbre et de coarbre ou lien dans la théorie des graphes:

- Un arbre : c'est un sous-graphe qui contient tous les noeuds du graphe et tel que toute branche ajoutée aux branches de l'arbre formerait une boucle.
- Un lien : c'est le sous-graphe formé des branches du graphe n'appartenant pas à l'arbre.

Le choix des branches arbres se fait souvent relativement au type d'éléments inclus dans le graphe, ce qui requiert un ordre préférentiel par ces derniers qui sera en priorité

pour les sources de tensions indépendantes suivies des sources de tension contrôlées, les capacités, les résistances et en dernier les self-inductances.

Il y aura donc deux problèmes à considérer à savoir :

- Trouver l'arbre T avec une préférence aux types d'éléments du circuit inclus dans l'arbre.
- Trouver la matrice B relative à l'arbre T choisi.

Les points suscités peuvent être réalisés par la machine qui effectuera des opérations élémentaires sur les lignes de la matrice A.. Cette dernière sera écrite sous la forme partitionnée suivante :

$$A = [A_T \ ; \ A_L]$$

Les colonnes sont arrangées de gauche à droite dans l'ordre correspondant aux éléments préferentiels de la structure d'arbres.

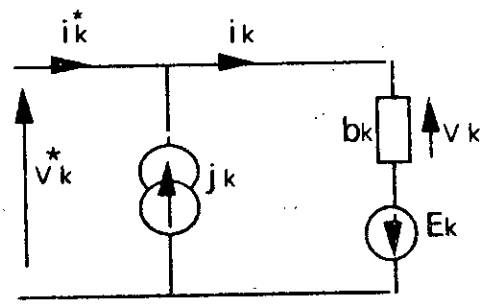
La renaissance des colonnes indépendantes est obtenue en réduisant la matrice A sous forme échelonnée par une série d'opérations élémentaires sur les lignes qui donne:

$$B^t = [-A_T^{-1} \ ; \ A_L] \quad avec \quad A_T^{-1} = I \quad (\text{I : matrice identitité})$$

II.2 ANALYSE NODALE

L'analyse nodale [34] constitue l'une des premières méthodes fondamentales d'analyse de circuits. Plusieurs programmes en langage évolué ont été basés sur cette méthode, tels que le programme ECAP[35] qui permet l'analyse de circuits en régime continu et alternatif.

Une branche k du graphe se présente sous la forme générale d'une branche du circuit de la figure suivante :



Le dipôle bk est soit une résistance linéaire, soit un courant contrôlé par une tension qui dépend de façon linéaire de la tension aux bornes d'une autre résistance.

En notant qu'une source contrôlée en courant dans la branche k s'écrit :

$$i_k = b_{kj} \cdot ij \qquad (II.4)$$

peut être remplacée par une source de courant contrôlée en tension :

$$i_k = g_{kj} \cdot vj \qquad (II.5)$$

où

$$g_{kj} = \frac{b_{kj}}{R_j} \quad et \quad R_j = \frac{vj}{ij} \qquad (II.6)$$

Le graphe du circuit étant formé de plusieurs branches composites, on écrit les vecteurs tensions v aux bornes de l'élément b_k du circuit et les vecteurs courants i qui le traversent comme suit :

$$v = [\, v_1, v_2, v_3, \ldots\ldots, v_n]^t$$
$$i = [\, i_1, i_2, i_3, \ldots\ldots, i_n]^t$$

On pose aussi pour les vecteurs sources de tension et pour les vecteurs sources de courant connus :

$$E = [\, E_1, E_2, E_3, \ldots\ldots, E_n]^t$$
$$J = [\, J_1, J_2, J_3, \ldots\ldots, J_n]^t$$

On écrira alors :

$$v^* = v - E \qquad (II.7a)$$
$$i^* = i - J \qquad (II.7b)$$

Soit A la matrice d'incidence, à partir de l'équation (II.1)$A\ i^* = 0$, on obtient la relation suivante:

$$A.i = A.J$$

Soit A_b la matrice admittance relative aux éléments b_k du circuit, on a alors :

$$i = A_b \cdot v \qquad (II.8a)$$
$$A \cdot A_b \cdot v = A \cdot J \qquad (II.8b)$$
$$A \cdot A_b (v^* + E) = A.J \qquad (II.8c)$$

Si on appelle X le vecteur des potentiels aux noeuds du circuit (noté vn , n étant le nombre total de noeuds du circuit), on aboutira à la relation suivante:

$$v^* = A^t \cdot X \qquad (II.9)$$

d'où

$A.Ab.A^t.X = A(J - Ab.E)$ (A^t étant la transposée de la matrice A)

On déinit la matrice admittance de l'analyse nodale de la façon suivante:

$$A = A.Ab.A^t \qquad \text{(II.10a)}$$

On déinit le vecteur second membre par:

$$Y = A(J - Ab.E) \qquad \text{(II.10b)}$$

on aboutit au système d'équation final de l'analyse nodale :

$$A.v_n = Y \qquad \text{(II.11)}$$

Cette dernière équation détermine complètement la solution du réseau, elle est applelée équation nodale et le processus de résolution du vecteur colonne v_n des potentiels aux noeuds est appelé analyse nodale.

Une fois la mise en équation effectuée c'est à dire la matrice d'incidence connue, il est alors facile de déterminer le vecteur v_n , soit par l'inversion de la matrice A (nxn) ce qui requiert plus de temps de calcul, soit par la résolution directe plus efficace.

Pour les circuits linéaires, la méthode d'analyse nodale est peut être plus populaire à cause de sa simplicité et sa facilité de programmation, mais elle souffre essentiellement de l'inconvénient qu'un seul type de source contrôlée est permis qui est celui des sources de courant à tension contrôlée et tous les autres types doivent être ramenés à ce dernier.

II.3 ANALYSE PAR LA METHODE DES VARIABLES D'ETAT

Depuis 1950, l'approche par la méthode de la variable d'état a été trés utilisée pour l'analyse des circuits larges et ceci à l'aide de l'ordinateur [34].

Dans cette méthode, un réseau linéaire est caractérisé par deux équations de la forme suivante:

$$\dot{x} = T x + H u \qquad \text{(II.12a)}$$

$$y = C x + D u + (D' u + ...) \qquad \text{(II.12b)}$$

où :

- u = (mx1) vecteur à m entrées (sources indépendantes)

Ces équations constituent un système de $(2b + n - 1)$ équations linéaires à $(2b + n - 1)$ inconnues (b courants de branches, b tensions de branches et $n-1$ tensions aux noeuds).

Ainsi par cette approche, nous disposons de beaucoup plus d'équations qu'avec les deux premières méthodes.

Comme ces équations sont extêmement éparses, il serait plus commode de les écrire sous la forme d'un tableau matriciel :

$$\begin{bmatrix} (n-1)\ \text{KCL} & A & 0 & 0 \\ b\ \text{KVL} & 0 & I & -A^t \\ b\ \text{relations} & k_i & k_v & 0 \end{bmatrix} \begin{bmatrix} i \\ v \\ v_n \end{bmatrix} = \begin{bmatrix} A.J \\ E \\ S \end{bmatrix} \qquad (II.13)$$

Différemment des deux autres approches , la solution de l'équation matricielle (II.13) implique toutes les variables du circuit et l'efficacité du calcul dépend de l'éparsité du tableau matriciel T [36].

$$T = \begin{bmatrix} A & 0 & 0 \\ 0 & I & -A^t \\ k_i & k_v & 0 \end{bmatrix} \qquad (II.14)$$

A titre d'exemple, un circuit RC sera formulé par la méthode du tableau dans l'annexe 4.

Nous venons de voir les différentes techniques mathématiques mises en oeuvre afin de formuler les équations du circuit tout en essayant à chaque fois de retenir la solution la plus adaptée à un traitement par un ordinateur en particulier.

Le concept trés puissant de la théorie des graphes qui consiste à numéroter les noeuds et les branches consécutivement .

L'orientation de ces dernières permet de déterminer les grandeurs électriques du circuit : tensions et courants des branches ainsi que les potentiels aux noeuds.

Par la suite, les équations d'interconnexions sont formulées grâce aux matrices d'incidence A (relative aux noeuds) et de circuit B (relative aux branches).

Ce qui est intéressant à noter du point de vue informatique est que la matrice B n'est autre que la matrice transposée de A .

Les relations descriptives des branches ont été déduites de l'application de la loi d'Ohm pour compléter les informations nécessaire à l'analyse des circuits.

Le système algèbro-différentiel : $F(X, \frac{dX}{dt}, t) = 0$ a été obtenu dans les trois différentes approches d'analyse et cela suivant le sous-ensemble x de X considéré.

1/ Dans la méthode nodale $x = v$ ensemble des tensions aux noeuds.

2/ Dans la méthode des variables d'état $x = q$ ensemble des énergies réactives (capacités, self)

3/ Dans la méthode du tableau $x = X$ ensemble de toutes les variables.

Nous avons retenu la dernière méthode afin de mieux exploiter les algorithmes d'inversion des matrices creuses.

Avant de résoudre pareil système, il a fallu discrétiser les dérivées en leur substituant une expression dont le calcul dépend de la méthode d'intégration choisie.

A cet effet nous consacrerons le chapitre III aux techniques d'intégration numériques .

CHAPITRE III

METHODES D'INTEGRATION NUMERIQUES

Le principe général de ces méthodes est de remplacer l'expression exacte [37,38]:

$$x_n = x_{n-k} + \int_{t_{n-k}}^{t_n} \frac{dx}{dt}\, dt$$

par une expression approchée du type :

$$x_n = \sum_{j=1}^{k} \alpha_j x_{n-j} + h \sum_{j=0}^{k} \beta_j \dot{x}_{n-j} \qquad \text{(III.1)}$$

où :

h est le pas d'intégration, α_j et β_j des coefficients d'intégration .

Dans le cas où $\beta_0 = 0$, la valeur de x_n cherchée ne dépend que des valeurs caculées aux instants antérieurs; ce qui consiste en la méthode dite explicite.

Si $\beta_0 \neq 0$, on ne peut plus calculer x_n explicitement à partir des valeurs précédentes car on a besoin de $\dot{x}_n$.

En pratique, ce système doit être résolu par une méthode itérative : cette méthode est dite implicite.

III.1 Méthodes explicites

Elles sont dites explicites car l'expression d'intégration suivante:

$$x_{n+1} = x_n + \phi(t_n, x_n, h) \qquad \text{(III.2)}$$

donne explicitement la valeur inconnue en fonction des variables calculées précédemment , on distingue les méthodes suivantes :

III.1.1 Méthode d'Euler

Elle est aussi appelée Backward Euler dans laquelle $h = 1$, $\alpha_1 = 1$ et $\beta_0 = 0$ dans l'expression (III.1) ce qui donne :

$$x_n = x_{n-1} + h \ \dot{x}_{n-1} \qquad \text{(III.3)}$$

Sous une forme plus améliorée, cette méthode devient avec $\dot{x} = f(x,t)$

$$x_{n+\frac{1}{2}} = x_n + \frac{h}{2} \dot{x}_n$$
$$x_{n+1} = x_n + h \ \dot{x}_{n+\frac{1}{2}}$$

III.1.2 Méthode de Runge-Kutta d'ordre 4 "RK 4"

Le principe de cette méthode consiste à évaluer quatre valeurs intermédiaires $K1$, $K2$, $K3$ et $K4$ et à faire la moyenne pondérée:

$$K1 = h\, f(x_n, t_n),$$
$$K2 = h\, f(x_n + K1/2, t_n + h/2),$$
$$K3 = h\, f(x_n + K2/2, t_n + h/2),$$
$$K4 = h\, f(x_n + K3, t_n + h),$$

L'expression finale de x_{n+1} devient la suivante:

$$x_{n+1} = x_n + [K1 + 2\,K2 + 3\,K3 + K4]/6 \qquad \text{(III.4)}$$

C'est une méthode qui gagne plus en précision que la première méthode car elle fait appel à des calculs intermédiaires.

III.2 Méthodes implicites

Le procédé de calcul est implicite vu que pour obtenir x_{n+1}, il faut au préalable résoudre l'équation :

$$x_{n+1} = \phi(x_{n+1})$$

La méthode la plus simple pour calculer les premiers points est de prédire une première valeur de x_{n+1} et cela grâce à une méthode explicite appelée le prédicteur. L'étape suivante consiste à procéder par des approximations successives ou des itérations sur la formule implicite appelée correcteur et cela jusqu'à ce que l'erreur entre deux calculs successifs soit acceptable. Parmi les méthodes les plus utilisées, on distingue :

III.2.1 Méthode d'Euler ou "Forward Euler"

Dans l'expression (III.1) on a $h = 1$, $\alpha_1 = 1$, $\beta_0 = 1$ et $\beta_1 = 0$ (III.1) ce qui donne :

$$x_n = x_{n-1} + h\ \dot{x}_n \qquad \text{(III.5)}$$

La détermination de x_n nécessite une résolution par itérations, largement facilitée par le fait qu'on dispose d'une valeur approchée x_{n-1} .

III.2.2 Méthode "des trapèzes"

Dans l'expression (III.1) on a $h = 1$, $\alpha_1 = 1$, $\beta_0 = 1/2$ et $\beta_1 = 1/2$, ce qui donne:

$$x_n = x_{n-1} + h/2\ (\dot{x}_n + \dot{x}_{n-1}) \qquad \text{(III.6)}$$

Si la méthode d'Euler explicite est utilisée comme prédicteur de la valeur de x_n, on aura:

$$x_{pn} = x_{cn-1} + h\ \dot{x}_{cn-1} \qquad \text{(III.7a)}$$

$$x_{cn} = x_{cn-1} + h/2\ (\ \dot{x}_{cn-1} + \dot{x}_{pn}\ ) \qquad \text{(III.7b)}$$

Cette paire d'équations constitue le prédicteur-correcteur de la méthode trapèzoïdale. Ainsi l'équation (III.7a) est le prédicteur qui fournit x_{pn} comme une fonction explicite de x_{cn-1}. L'équation (III.7b) est le correcteur qui fournit x_{cn} comme fonction explicite de x_{cn-1} et x_{pn}.

Une plus grande précision résulte de ces méthodes par rapport à celles dites explicites grâce au concept de prédicteur-correcteur.

D'autres méthodes plus sophistiquées [39] ont suivi telles que celles de Brayton [40] et dont l'intérêt particulier est qu'elle autorise un pas d'intégration supérieur à la plus faible des constantes de temps du circuit tout en restant stable et s'applique surtout aux circuits non-linéaires.

Un autre procédé plus performant utilisant la méthode de Gear [41] et qui consiste à ajuster le pas d'intégration à la vitesse d'évolution des phénomènes mérite un bref exposé.

III.2.3 Méthode de Gear

C'est une méthode à ordre et à pas variables et qui utilise comme formule implicite (ou correcteur) l'expression suivante :

$$x_n = \sum_{i=1}^{k} \alpha_i \; x_{n-i} + h \, \beta_0 \, \dot{x}_n \qquad \text{(III.8a)}$$

quant au prédicteur, il s'écrit :

$$x_n^* = \sum_{i=1}^{k} \alpha_i^* \; x_{n-1} + h \, \beta_1^* \, x_{n-1} \qquad \text{(III.8b)}$$

x_n^* est la valeur prédite au moyen des coefficients $\beta_i^* \neq \beta o$ et $\alpha_i^* \neq \alpha i$.

Le pas et l'ordre varient vu qu'en chaque point une valeur de h et de k seront utilisées tel que :

$$\varepsilon = Ck \; .h^{k+1}$$

soit la plus faible possible (le terme Ck dépend de la dérivée d'ordre $k+1$).

ε est appelée erreur de troncature et c'est une évaluation de l'erreur commise en tronquant le développement de Taylor à l'ordre p quand on écrit la formule approchée du correcteur.

Le contrôle de cette erreur de troncature permet donc d'ajuster le pas d'intégration à la vitesse d'évolution des phénomènes, ce qui fait l'originalité et la puissance d'une telle méthode.

Bien que cette méthode offre plus de précision à cause du contrôle de l'erreur, elle est plus adaptée aux circuits non-linéaires gérés par des équations différentielles et algébriques couplées.

III.3 ETUDE DE LA STABILITE DES METHODES D'INTEGRATION

Lors de cette étude, nous nous intéresserons surtout à l'influence du choix de la méthode d'intégration sur la précision ainsi que sur la condition de stabilité.

Le principe général de ces méthodes étant de remplacer l'expression exacte [34]:

$$x_n = x_{n-k} + \int_{t_{n-k}}^{t_n} \frac{dx}{dt}\, dt \qquad \text{(III.9a)}$$

par une expression approchée du type:

$$x_n = \sum_{j=1}^{k} \alpha_j x_{n-j} + h \sum_{j=0}^{k} \beta_j \dot{x}_{n-j} \qquad \text{(III.9b)}$$

où h représente le pas d'intégration et α , β des coefficients bien déterminés.

Pour $x(t)$ un polynôme de degré p du temps, cette relation approchée est équivalente sur l'intervalle (t_{n-k}, t_n) à une série de Taylor tronquée à l'ordre p.

$\varepsilon_T(n)$ est alors le reste de Lagrange et a pour expression [31]:

$$\varepsilon_T(n) = \frac{h_n^{p+1}}{(p+1)} \dot{x}^{p+1}(t_n) + \theta(h^{p+1}) \qquad \text{(III.10)}$$

Où $\dot{x}^{p+1}(t_n)$ est la dérivée $(p+1)^{ième}$ en t_n et h_n^{p+1} l'intervalle de temps à l'instant n élevé à la puissance $p+1$.

On voit bien que l'erreur, c'est à dire l'écart entre la vraie valeur $x(tn)$ et sa valeur approchée est lié à l'intervalle de temps ou "pas" hn.

Il est fondamental que la simulation donne des résultats satisfaisants donc entachés d'une faible erreur de calcul.

Il faut pour cela choisir la méthode de calcul en se donnant comme but à la fois d'optimiser la solution sur le plan informatique et de satisfaire au mieux la rigueur mathématique , ce qui revient à répondre aux impératifs de précision et de stabilité.

III.3.1 Notion d'erreur locale et d'erreur totale

Comme la solution obtenue par les méthodes d'intégration numériques n'est jamais exacte, il est important de connaître l'erreur commise à chaque pas aussi bien que l'erreur cumulée aprés un intervalle de temps défini. Pour cela nous considèrerons l'équation test:

$$\dot{x} = f(x) = -Ax$$

dont la solution exacte est donnée par l'expression suivante:

$$x(t) = \sum_i \mu_i e^{-\lambda_i t}$$

Les λi représentent les valeurs propres de la matrice A.

Le choix de cette équation test n'est pas seulement dû au fait qu'elle possède une solution exacte (à laquelle les autres solutions numériques peuvent être comparées), mais aussi car la majorité des solutions des autres équations différentielles peuvent être approximées par une portion de l'exponentielle.

Les mesures d'erreur de l'équation test sont données par les deux erreurs suivantes:

L'erreur locale est définie par:

$$\varepsilon = (x_n e^{-\lambda h}) - x_{n+1} \qquad \text{à } t = t_{n+1}$$

L'erreur totale est définie par:

$$\varepsilon_T = (x_0 e^{-\lambda t_{n+1}}) - x_{n+1} \qquad \text{à } t = t_{n+1}$$

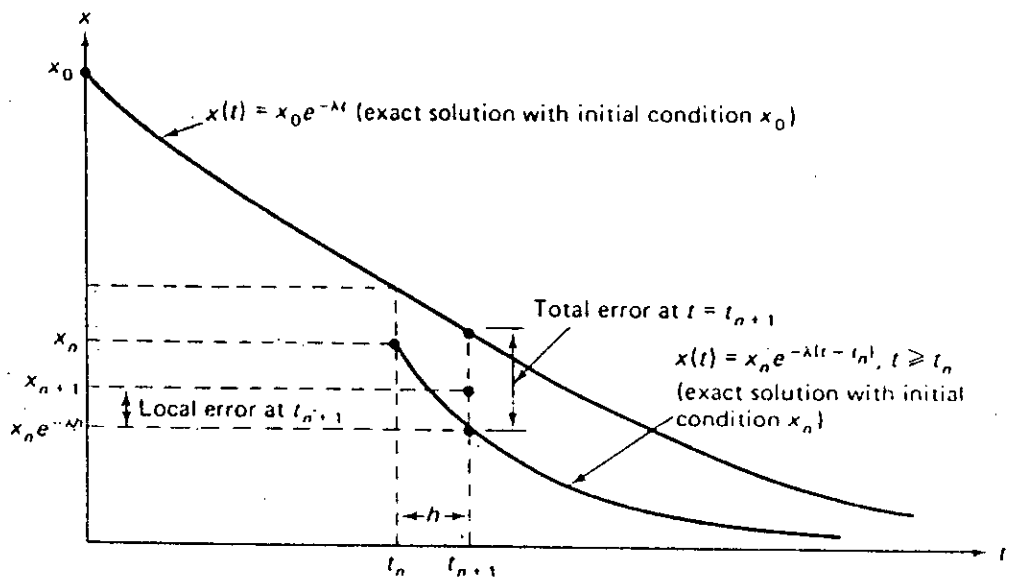


Figure III.1 Différence entre l'erreur locale et l'erreur totale

L'erreur locale peut être interprétée comme l'erreur qui se produit à $t = t_{n+1}$ avec x_n comme condition initiale. Au contraire l'erreur totale peut être interprétée comme l'erreur actuelle accumulée de l'instant $t = 0$ à $t = t_{n+1}$ avec x_0 comme condition initiale.

La différence entre ces deux erreurs est illustrée sur la figure III.1.

L'erreur locale à $t = t_{n+1}$ peut être soit positive soit négative et l'erreur totale qui tient compte de l'accumulation des erreurs locales peut ou ne peut pas augmenter avec le temps.

En termes d'approximation, l'algorithme d'intégration numérique dont l'erreur totale n'est pas amplifiée mais décroît avec le temps est dit numériquement stable.

Les algorithmes qui ne possèdent pas cette propriété sont dits numériquement instables. Si l'erreur locale est faible, l'erreur totale d'un algorithme instable peut éventuellement être amplifiée suffisamment pour rendre la solution inutile.

Nous nous intéresserons en premier lieu à l'algorithme d'Euler que nous avons retenu comme moyen d'intégration.

III.3.2 Stabilité de la formule d'Euler explicite

L'étude consiste à comparer la relation exacte :

$$x(t_n) = x(t_{n-1}) + h\,\dot{x}(t_{n-1}) + \varepsilon_T \qquad \text{(III.10a)}$$

à la relation approchée suivante:

$$x_n = x_{n-1} + h\,\dot{x}_{n-1} \qquad \text{(III.10b)}$$

Dans le cas linéaire où $\dot{x} = f(x) = -Ax$ on aura:

$$x(t_n) = x(t_{n-1}) + h\,A\,x(t_{n-1}) + \varepsilon_T \qquad \text{(III.11a)}$$

$$x(t_n) = (I + h\,A)\,x(t_{n-1}) + \varepsilon_T \qquad \text{(III.11b)}$$

qui s'écrira alors sous la forme approchée suivante:

$$x_n = (I + h\,A)\,x_{n-1} \qquad \text{(III.12a)}$$

et l'erreur deviendra alors:

$$\varepsilon_n = x(t_n) - x_n = (I + h\,A)\,\varepsilon_{n-1} + \varepsilon_T \qquad \text{(III.12b)}$$

Afin d'éviter l'amplification des erreurs, il faut que les valeurs propres de la matrice (I+hA) soient de module inférieur à 1.

$$1+h\lambda_i < 1$$

Les points qui représentent cette inéquation sont à l'intérieur d'un cercle de centre -1 et de rayon 1; ce qui définit le domaine de stabilité de cette méthode (voir fig.III.2).

Dans le cas simple où les valeurs propres sont réelles, le rayon de convergence est réduit au segment de droite (-2,0).

Les λ_i sont en fait les inverses des constantes de temps du circuit:

$$\lambda_i = -1/\tau_i$$

La condition de stabilité est alors la suivante:

$-2 < -h/\tau_i < 0$ soit $h < 2\tau_i$ et ceci sera vérifié pour tout $h < 2\tau_{min}$

III.3.3 Stabilité de la formule d'Euler implicite

La démarche est identique, on trouve que la condition de stabilité est la suivante:

$$1/(1-h\lambda_i) < 1$$

Le domaine de convergence est l'extérieur d'un cercle de rayon 1 et centré en 1 (Fig.III.3).

On remarque que si les valeurs propres sont réelles, elles sont obligatoirement négatives donc $(1-h\lambda_i) > 1$ est toujours respecté donc la formule d'Euler implicite est stable sans restriction sur h (>0).

Ce cas pour lequel le demi-plan des réels négatifs est inclus dans le domaine de stabilité est appelé A-stable.

III.3.4 Stabilité de la formule des trapèzes

La relation $x_n = x_{n-1} + h/2\ (\dot{x}_n + \dot{x}_{n-1})$ montre directement que c'est une méthode implicite d'ordre 2 donc elle est A-stable.

A la lumière de cet exposé sur les différentes méthodes d'intégration numériques consacré à la première partie de ce chapitre, nous avons retenu la méthode d'Euler explicite qui est la plus adaptée à notre cas. En effet, les circuits considérés sont linéaires d'où une marge de précision suffisante et du point de vue temps, le gain est considérable d'où le compromis précision-rapidité.

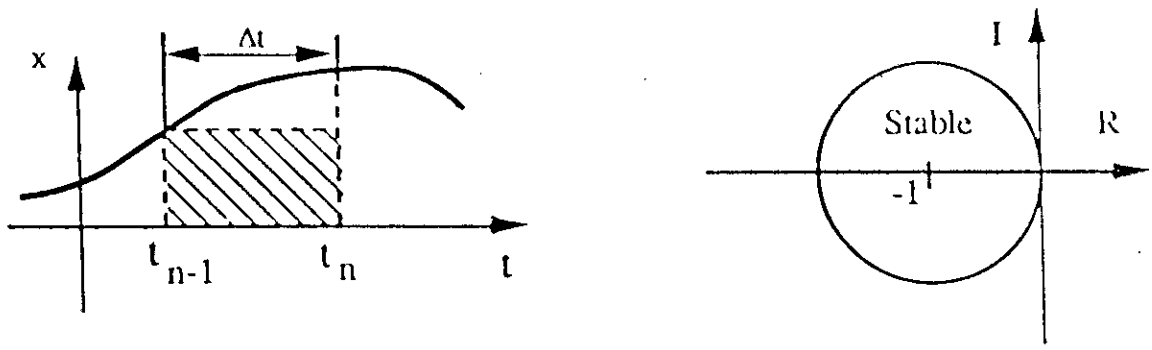


Figure III.2 Principe et domaine de stabilité de la méthode d'Euler explicite

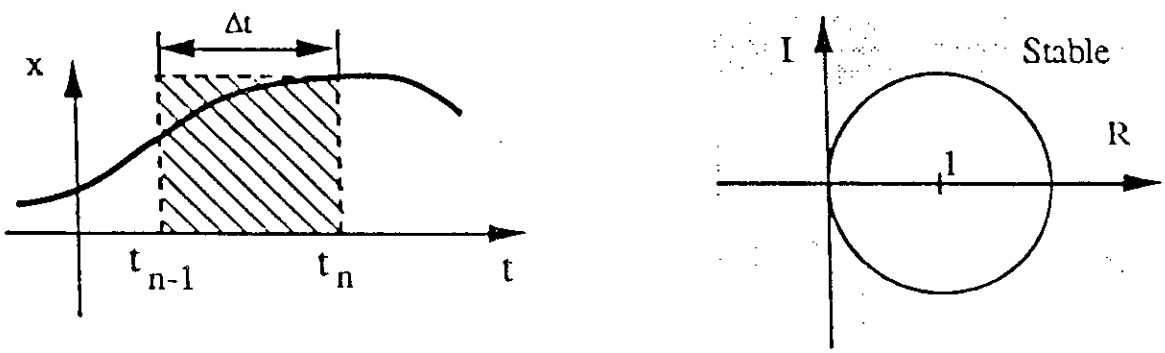


Figure III.3 Principe et domaine de stabilité de la méthode d'Euler implicite

L'emploi de la méthode d'Euler explicite implique de choisir un pas d'intégration directement lié aux constantes de temps les plus brèves du circuit. Le choix de cet algorithme comme moyen d'intégration a été dicté par les raisons suivantes:

La rapidité d'éxécution ainsi que la réduction de l'encombrement mémoire vu que le calcul des itérations de détermination des dérivées a été évité, chose impossible dans les méthodes implicites.

En effet, ces dernières permettent une amélioration de la précision par un moyen de correction où deux niveaux d'intervention globale et locale sont proposés.

Afin de garantir une réponse exacte, il est nécessaire de disposer d'une méthode qui permet de corriger cette erreur et qui consiste en un renouvellement du calcul sans incrémenter l'instant t_n.

Cette idée consiste en un algorithme de prédiction- correction, où le premier passage est le calcul à un nouvel instant t_n (prédiction) suivi de plusieurs itérations toujours à t_n et ceci afin de trouver la valeur finale (correction).

En règle générale, le prédicteur et le correcteur ont deux lois différentes et le nombre d'itérations est déduit aprés un test de convergence.

Pour une méthode de prédiction du premier ordre, l'utilisation de la même loi lors de la correction fait converger la réponse plus rapidement (2 ou 3 itérations).

On conclut donc que l'apport de la correction pénalise le temps CPU d'un coefficient 2(m+1), m étant le nombre d'itération de correction.

Le prochain chapitre sera consacré en premier lieu aux algorithmes numériques nécessaires à la résolution du système $F(X, \frac{dX}{dt}, t) = 0$ qui sera linéarisé et ramené à un système matriciel de la forme : $\boldsymbol{A} \cdot X = B$.

En second lieu, une étude sera largement élaborée concernant la méthode de résolution retenue.

CHAPITRE IV

METHODES DE RESOLUTION DU SYSTEME MATRICIEL

A.X = B

Comme il a été vu précédemment, avec la méthode du tableau considérée pour la formulation des équations du circuit à analyser, on est amené à la résolution d'un système matriciel d'équations linéaires de la forme :

$$A \cdot X = B \qquad (IV.1)$$

Pour cela, il existe deux classes de méthodes: les méthodes directes et les méthodes indirectes ou itératives [42].

IV.1 METHODES DIRECTES

Avec ces méthodes, la résolution se fait en un nombre fini d'opérations.

IV.1.1 Méthode de Cramer

C'est certainement la plus connue et la moins recommandée surtout pour des matrices dont l'ordre dépasse 10. En effet, du système $A \cdot X = B$ on déduit la solution X par le calcul de la matrice inverse (A^{-1}) .

Le nombre d'opérations impliquées dans ce calcul est alors:

- Pour un petit système (n = 10) de l'ordre de 3.10^9.
- Pour un système moyen (n = 50) de l'ordre de 7.10^{67}.

Si on suppose qu'un ordinateur exécute une opération par nanoseconde, alors le temps de calcul du déterminant d'une matrice de taille moyenne est pratiquement égal à 10^{47}années.

Il est clair que cette méthode est impraticable au sens du temps de calcul, de plus l'erreur d'arrondi cumulée va en croissant avec le nombre d'opérations, ce qui rend la précision douteuse pour les moyens et grands systèmes.

IV.1.2 Méthode d'élimination de Gauss

C'est un algorithme récursif qui, à chaque étape transforme le système original en un autre plus petit. Ce procédé appélé "triangularisation" de la matrice, ne met en oeuvre que la résolution d'une équation à une inconnue. Il est suivi d'un autre procédé appelé "substitution arrière" au cours duquel les équations sont résolues successivement dans l'ordre inverse.

Pour les circuits de grandes taille ($n > 50$), le nombre total d'opérations effectuées est égal à $\frac{n^3}{3} + \frac{n^2}{2}$ dont $\frac{n^3}{3}$ opérations pour la triangularisation et $\frac{n^2}{2}$ pour la substitution arrière.

IV.1.3 Décomposition L U

Cette méthode résulte d'une modification de la méthode d'élimination de Gauss. Le principe de base consiste à mettre la matrice A du système (IV.1) sous la forme d'un produit de deux matrices, l'une triangulaire inférieure L et l'autre triangulaire supérieure U:

$$A = L\,U$$

Cette expression subsituée dans le système (IV.1) donne:

$$L\,U\,.X = B$$

on pose :

$$U\,X = Y \quad (IV.2a) \qquad \text{et} \qquad L\,Y = B \quad (IV.2b)$$

La résolution de (IV.2b) est appelée "substitution avant", tandis que la résolution (IV.2a) est appelée "substitution arrière". L'avantage de cette méthode est qu'une fois la factorisation de la matrice A effectuée, le système d'équations peut être résolu pour plusieurs vecteurs second membre.

Le nombre total d'opérations nécessaires à la triangularisation est de l'ordre de n^2 seulement.

Les variantes de la décomposition LU correspondent à l' algorithme de Crout dans lequel les éléments diagonaux de la matrice U sont égaux à 1, alors que pour l'algorithme de Dolittle ce sont les éléments diagonaux de la matrice L qui sont égaux à 1.

On peut noter que cette méthode nécessite moins d'espace mémoire que la méthode de Gauss vu que les élément des matrices L et U viennent "écraser" ceux de la matrice A au fur et à mesure qu'ils sont calculés.

Le déroulement de ces méthodes de résolution des systèmes matriciels repose sur l'utilisation de pivots comme " charnières" de ces procédures.

Afin d'éviter un blocage de la décomposition, on doit au préalable s'assurer que tous les éléments diagonaux sont non nuls. L'opération de "préarrangement" est réalisée grâce à la permutation d'une ligne.

D'autre part, en cours de décomposition on doit avoir le souci de minimiser les erreurs d'arrondi afin de se prémunir des débordements de calcul, chose possible si on prend comme pivot l'élément de plus forte valeur absolue c'est l'opération de pivotage.

Il existe deux types de pivotage:

- Le pivotage complet dans lequel on intervertit deux lignes et deux colonnes au niveau de la matrice réduite. L'élément ayant la plus grande

valeur absolue est alors pris comme pivot, c'est le cas de la méthode de Gauss.

• Le pivotage partiel dans lequel l'élément ayant la plus grande valeur absolue dans la première ligne ou la première colonne de la matrice réduite est alors pris comme pivot.

En ce qui concerne la décomposition LU, le choix du pivot se restreint aux éléments de la ligne ou de la colonne en cours d'élimination, aprés leur mise à jour. Quant aux éléments calculés à chaque étape de la décomposition, sont obtenus par des décomposition linéaires de lignes ou de colonnes.

Ainsi des emplacements qui contenaient initialement des zéros peuvent être "remplis" par d'autres non nuls, c'est le phénomène de "remplissage".

Nous nous sommes attardés sur le problème de pivotage ainsi que son influence sur le remplissage vu que dans ce qui suivra, on verra comment tout cele sera optimisé dans le cas des matrices creuses grâce à l'algorithme de Crout adapté.

IV.2 METHODES ITERATIVES

Ces méthodes font passer les calculs d'un estimé x(k) de la solution à un autre estimé x(k+1), s'il y a convergence la solution ne pourrait être atteinte qu'aprés un nombre fini d'étapes.

La matrice n'étant pas transformée au cours des calculs, le problème de l'accumulation des erreurs devient moins crucial.

Le principe de ces méthodes consiste à écrire la matrice A du système (IV.1) sous la forme suivante:

$$A = M - N \qquad (IV.3)$$

Suivant la nature des matrices M et N, on distingue trois méthodes.

IV.2.1 Méthode de Jacobi

Dans ce cas: $M = D$

$N = L+U$

D matrice diagonale

L matrice triangulaire inférieure

U matrice triangulaire supérieure

Le système d'équations (IV.1) devient alors:

$$[D - (L + U)]\, X = B \qquad (IV.4)$$

en admettant que D^{-1} existe, on aboutit à la solution suivante:

$$X = D^{-1}(L+U)\, X + D^{-1} B \qquad (IV.5)$$

X(0) étant estimé

$X(k+1) = D^{-1}(L+U)\, Xk + D^{-1} B$ et ε est tel que : $| X_i^{k+1} - X_i^k | < \varepsilon$; $i = 1, n$

Le terme X(k) correspond à la valeur de X à la $k^{ième}$ itération.

IV.2.2 Algorithme de Gauss-Seidel

Dans ce cas, $M = D - L$

$N = U$

Le système d'équations (IV.1) devient :

$$[D - (L + U)]\, X = B \quad (IV.6)$$

En admettant que D^{-1} existe, on aboutit à la solution suivante:

$$X = D^{-1}(L+U)\, X + D^{-1} B \qquad (IV.7)$$

X(0) étant estimé

$X(k+1) = D^{-1}(L+U)\, X(k) + D^{-1} B$ et ε est tel que : $| X_i^{k+1} - X_i^k | < \varepsilon$; $i = 1, n$

Le chemin de la résolution pour ces deux algorithmes est itératif avec une garantie de convergence assurée, seulement en ce qui concerne la rapidité, la méthode de Gauss-Seidel converge plus vite que celle de Jacobi.

Un autre algorithme dit de relaxation, permet d'optimiser la convergence de ces méthodes itératives.

IV.2.3 Algorithme de relaxation

Dans cette méthode, un paramètre de pondération w est introduit de façon à obtenir un convergence optimale, on a :

$$M = \frac{1}{w}(D - L)$$

$$N = \frac{1-w}{w}(D + U)$$

D'où :

$$X(k+1) = (D - L)^{-1}[(1-w)\, D + w\, U]\, X(k) + w\, (D - L)^{-1} B \qquad (IV.8)$$

Le résidu à la $k^{ième}$ itération est donné par:

$$r^k = B - A X^k \qquad (IV.9)$$

ou encore pour la $i^{ième}$ composante:

$$r_i^{k+1} = a_{ii} (X_i^{k+1} - X_j^k) \; ; \quad a_{ii} \neq 0$$

Avec la relaxation, on obtient:

$$X_i^{k+1} = w \frac{r_i^{k+1}}{a_{ii}} + X_i^k \qquad (IV.10)$$

L'idée de cette méthode, est que si la correction apportée à la iième composante améliore la convergence, il est préférable de l'accentuer ($w > 1$).

Il a été démontré [42] que la convergence de la méthode de relaxation est assurée pour

$$1 < w < 2$$

Il est évident que les méthodes itératives ne conviennent pas aux matrices creuses et de taille élevée ($n > 100$) vu que la résolution du circuit se fait dans son ensemble à chaque itération conduisant à des durées de calcul excessives.

IV.3 METHODE DE RESOLUTION DES MATRICES EPARSES

La méthode du tableau que nous avons retenue pour la formulation des équations du circuit, nous montre que nous avons affaire à une matrice creuse dite aussi "éparse" (moins de 30% des éléments sont non nuls). De ce fait nous exploiterons les algorithmes de résolution proposés dans ce but par Hatchel et al [5].

IV.3.1 Principe de la méthode

L'algorithme de résolution proposé par les auteurs est basé sur celui de Crout dans lequel le temps de calcul est fortement réduit car les opérations triviales qui font intervenir les éléments nuls sont totalement omises dans les calculs.

Par conséquent, l'adressage mémoire de ces éléments est aussi évité, d'où les deux traits essentiels de la technique des matrices éparses:

- Seuls les éléments non nuls avec nécessairement une information indexée sont stockés.
- Les opérations qui font intervenir des zéros ne sont pas exécutées.

IV.3.2 Stockage des éléments non nuls

Comme il a été vu précédemment, on ne garde que les éléments non nuls et on n'opère que sur eux.

Quand la matrice A d'ordre n est éparse, c'est inefficace de la stocker dans un tableau à deux dimensions ce qui requiert n^2 places mémoires.

Si lors de la factorisation de la matrice A en une nouvelle matrice Q (Q = L+U -I) dans l'algorithme de Crout, de nouveaux éléments non nuls peuvent être crées (remplissage), ils doivent aussi être stockés.

En d'autres termes, cela revient à stocker les éléments non nuls de la matrice Q en plus de ceux de A. Pour cela , il suffirait de reécrire la matrice Q sur la matrice A.

Au début de la factorisation LU, la la matrice considérée est la matrice A, à la fin elle devient la matrice Q.

Les éléments de cette dernière sont stockés en trois tableaux unidimensionnels:

- Le tableau DIAG : Il stocke les éléments diagonaux non nuls de la matrice Q dans l'ordre suivant : q_{11}, q_{22}, q_{33},..., q_{nn}.
- Le tableau UPACK : Il stocke les éléments diagonaux non nuls de la matrice U dans l'ordre suivant : ligne 1, ligne 2, ligne 3, ..., ligne n.
- Le tableau LPACK : Il stocke les éléments diagonaux non nuls de la matrice L dans l'ordre suivant : colonne 1,colonne 2,colonne 3, ..,colonne n.

Les éléments non nuls ayant été stockés, une façon simple de les adresser doit être alors élaborée.

IV.3.3 Mode d'adressage des éléments non nuls

Afin de localiser l'élément non nul et non diagonal a_{ij} (ou q_{ij}) on définit deux termes:

Le premier UCOL qui fournit l'information sur la position de la colonne des éléments dans le tableau UPACK.

En général, UCOL (k) = m signifie que le kième élément dans UPACK est dans la $m^{ième}$ colonne de la matrice A (ou Q) et comme la matrice A est symétrique structuralement, cette équation signifie que le $k^{ième}$ élément dans LPACK est dans la mième ligne de A .

Le second terme UROWST, renseigne sur quel élément de UPACK chaque ligne de la matrice U commence. En général, UROWST(I) = m et UROWST(I+1) = m1 signifie qu'il y a (m1 - m) éléments non nuls, non diagonaux de la ième ligne de U qui sont stockés dans UPACK , qui commence avec le $m^{ième}$ élément de UPACK.

Par symétrie, cela signifie aussi que la $1^{ère}$ colonne de la matrice L commence avec le $m^{ième}$ élément dans LPACK.

Si on considère un programme d'analyse nodale de 50 noeuds, de 200 branches alors les tableaux DIAG et UROWST sont dimensionnés à 50.

En moyenne, 200 branches produisent 400 éléments diagonaux non nuls dans la matrice d'admittance Yn.

Ainsi UPACK, LPACK et UCOL sont chacuns dimensionnés à 200. Un total de 700 cellules sont nécessaires pour le stockage et l'indexage comparé avec les 600 cellules requises pour stocker le triplet (i, j, a_{ij}) et les 2500 cellules requises pour stocker Yn comme un tableau à deux dimensions.

Le schéma précédent requiert donc plus de place mémoire que le procédé des triplets [42]; cet inconvénient est compensé par la facilité de factorisation LU qui utilise le procédé décrit dans ce qui suit.

IV.3.4 Factorisation L U

Une fois la structure de la matrice Q obtenue et les régions DIAG, UPACK, UCOL et UROWST mises au point, on peut procéder à la factorisation LU en opérant seulement sur les éléments non nuls.

Afin de mieux comprendre l'algorithme de Crout adapté aux matrices éparses, on rappellera brièvement celui relatif aux matrices quelconques. Il s'agit de transformer le système A . X = B sous forme de deux systèmes d'équations:

L .Y = B où L est une matrice triangulaire inférieure et U.X = Y où U est une matrice triangulaire supérieure avec $U_{ii} = 1$.

Pour cela, une matrice auxilliaire Q = L+U-I est définie (I matrice identité).

L'algorithme de factorisation est alors le suivant:

1/ Copier la colonne 1.

2/ Dans la ligne 1, diviser tous les éléments non diagonaux par l'élément diagonal.

3/ Soustraction: pour chaque élément (i,j); i >1 et j >1

soustraire du produit (i,1).(1, j) l'élément aij on obtient alors:

$$l_{ik} = a_{ik} - \sum_{j=1}^{k-1} l_{ij}\, u_{jk} \quad k = \overline{1,n} \; ; \; i = \overline{k,n} \qquad \text{(IV.11a)}$$

$$u_{kk} = 1$$

$$u_{kj} = \frac{\left[a_{kj} - \sum_{i=1}^{k-1} l_{ki}\, u_{ij} \right]}{l_{kk}} \quad ; \quad j = \overline{k+1,n} \qquad \text{(IV.11b)}$$

4/ Rechercher si l'ordre de la sous-matrice sujette aux soustractions dans l'étape 3/ est de deux ou plus. Se referer à la sous-matrice et répéter l'algorithme, sinon la matrice obtenue est la matrice Q.

Pour l'algorithme de résolution, on procède comme suit:

$Y = L^{-1}.B$ donc :

$$y_i = \frac{\left[b_i - \sum_{k=1}^{i} l_{ik} \cdot y_k \right]}{l_{ii}} \quad ; \quad i = \overline{1,n} \qquad \text{(IV.12a)}$$

$X = U^{-1}.Y$ donc :

$$x_j = \left[y_j - \sum_{k=j+1}^{n} u_{jk} . x_k \right] ; \quad j=\overline{1,n} \qquad \text{(IV.12b)}$$

La procédure de résolution adaptée aux matrices éparses est la même sauf qu'un effort est nécessaire afin d'accéder à tout élément a_{ij}.

L'algorithme de résolution est alors le suivant:

- L'étape 1 consiste à copier la colonne 1 , les éléments sont stockés dans le tableau LPACK.
- L'étape 2: consiste à efectuer une division connaissant UROWST, UPACK et DIAG.
- L'étape 3 consiste à effectuer le produit des éléments des tableaux LPACK et UPACK. Le terme UCOL permettant de déterminer l'emplacement des éléments du tableau UPACK , on peut alors en soustraire le produit précédent.
- L'étape 4 La sous-matrice à considérer par la suite est complètement décrite par les dernières parties des différentes régions comme suit:

DIAG et UROWST commençant à partir du 2ième élément.UPACK, LPACK et UCOL commençant à partir de l'élément UROWST(2) = m , ce qui complète la première itération.

La seconde itération décrite encore avec l'étape 1, mais la matrice correspond à la sous -matrice définie à l'étape 4. Le processus est alors répété jusqu'à la (n-1) ^{ième} itération .

La dernière matrice mise à jour est la matrice Q = L+U-I.

Afin d'éviter les opérations sur les éléments nuls, on utilise les cinq tableaux définis auparavant et on organise la substitution avant et arrière de telle façon que les coefficients de la matrice U soient utilisés ligne par ligne, car ils sont stockés dans le tableau UPACK. De manière similaire, les coefficients de la matrice L sont utilisés colonne par colonne.

Suivant la décomposition faite précédemment, nous avons obtenu les deux équations suivantes:

$$L.y = B \qquad (IV.13a) \qquad \text{et} \qquad U.x = y \qquad (IV.13b)$$

L et U étant triangulaires, les solutions sont obtenues par une substitution avant de ((IV.13b)) et arrière de (IV.13a) .

$$y(o) = B \qquad (IV.14a)$$

$$y_i^{(i)} = \frac{y_i^{(i-1)}}{l_{ii}} \qquad (IV.14b)$$

$$y_k^{(i)} = y_k^{(i-1)} - l_{ki}.y_i^{(i)} \quad ; \; i = 1, n \quad \text{et } k = i+1, n \qquad (IV.14c)$$

Les éléments de la matrice L sont dans le tableau LPACK et seuls les éléments dans la même colonne de L sont utilisés dans chaque réduction de colonne et on obtient :

$$x_j = y_j - \sum_{k=j+1}^{n} u_{jk}\, x_k \qquad j = n-1,1 \qquad (IV.15a)$$

$$xn = yn \qquad (IV.15b)$$

De la même façon, les éléments sont pris dans UPACK et seuls les éléments d'une ligne sont considérés dans l'algorithme de la façon dont UPACK est structurée.

Nous venons d'aborder les différentes techniques mathématiques de résolution des systèmes matriciels, en essayant de relever leurs avantages ainsi que leurs inconvénients.

La méthode de résolution retenue et qui s'adapte bien à la situation particulière des matrices éparses est celle qui correspond à l'algorithme de Crout [36].

Afin d'exploiter au mieux cette technique, un mode simple et économique de stockage et d'adressage des éléments de la matrice à réduire a été élaboré et qui consiste en deux traits :

1/ Seuls les éléments non nuls avec nécessairement une information indexée sont stockés dans trois régions distinctes .

2/ Les opérations qui font intervenir des zéros ne sont pas exécutées.

Dans le prochain chapitre, on exposera en premier lieu l'élaboration du programme proprement dite en identifiant les différents modules qui le constituent . En second lieu, des exemples de simulations de circuits seront donnés afin de montrer les possibilités du programme.

CHAPITRE V

ELABORATION DU PROGRAMME
APPLICATION DU PROGRAMME

V.I STRUCTURE DU SIMULATEUR ELECTRIQUE DE CIRCUITS

Un simulateur de circuits possède en général la structure représentée dans la figure V.1b déduite de son schéma fonctionnel global donné dans la figure V.1a.

Les étapes suivantes s'enchaîneront lors d'une simulation:

La lecture des données, la mise en équation , la résolution et enfin l'exploitation des résultats.

V.1.1 Saisie des données

Celle-ci peut revêtir plusieurs formes selon l'environnement matériel et logiciel dans lequel se situe le simulateur.

Les deux formes les plus courantes sont la saisie textuelle et la saisie graphique, dans les autres cas il peut s'agir de données générées par d'autres applications de CAO comme par exemple les éditeurs de masques.

Dans le mode de saisie graphique, l'utilisation d'un terminal graphique s'impose grâce auquel l'utilisateur place sur l'écran les éléments constitutifs du schéma électrique qu'il désire simuler ainsi que leur interconnexions. Une procédure permet alors d'extraire de cette description un fichier compatible avec le format d'entrée du simulateur électrique.

Les seules commandes restant à définir par l'utilisateur concernent le choix du type d'analyse souhaitée.

Le mode de saisie textuelle est le plus simple car il ne met en jeu qu'un terminal alpha-numérique. Le fichier décrit contient plusieurs types d'informations:

- La définition des modèles de composants où les paramètres définis à ce niveau vont être communs à tous les composants d'un même type représentés dans la topologie du circuit sauf s'ils y sont redéfinis.
- Une description de la topologie du circuit à simuler qui se fait par une suite de commandes dont chacune correspond à un élément du circuit qui peut être un composant élémentaire ou autre.
- Les commandes de simulation qui indiquent le type d'analyse souhaitée ainsi que ses paramètres.

Ce dernier moyen de saisie s'adapte bien à notre situation matérielle et c'est pourquoi nous l'avons retenu.

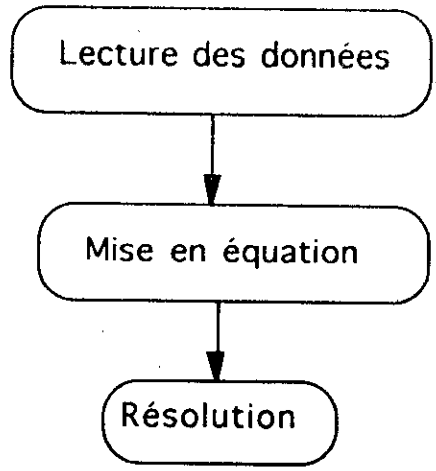


Figure 5a Schéma global d'un simulateur de circuits

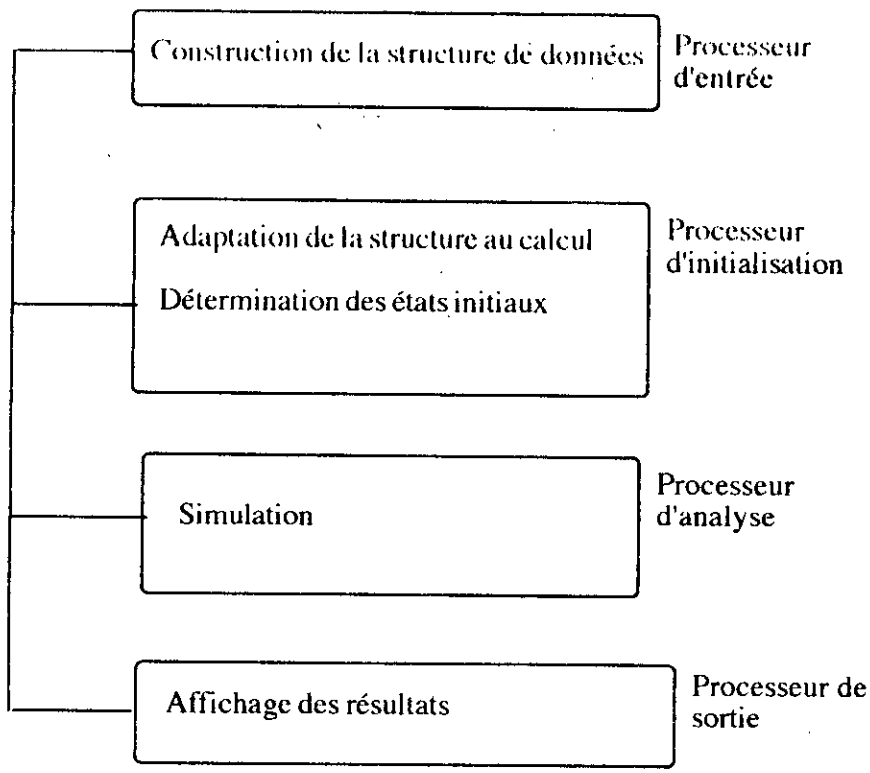


Figure 5b Structure d'un simulateur de circuits

V.1.2 Langage de description source

Nous avons mis au point un langage simple (des exemples sont donnés en annexe 5) , d'utilisation aisée et qui permet de décrire facilement tout circuit électrique.

Au chapitre II, il a été vu que pour représenter un circuit il fallait se servir de schémas composés de symboles qui décrivent chacun un type bien particulier de composants ainsi que les liaisons qui existent entre eux.

Dans un schéma, chaque composant est entièrement déterminé par les trois informations suivantes:

- Son type (inclu dans son nom: la lettre initiale).
- Sa position (la liste de ses connecteurs).
- Sa valeur.

Les résistances, les capacités, les selfs, les sources de courant et les sources de tensions constituent les dipôles de base dont est constitué tout circuit électrique.

La syntaxe de ces éléments est alors comme suit:

NOM CONNECTEURS VALEUR

NOM : c'est une suite alpha-numérique dont le premier terme est alphabétique, il définit le dipôle en question ainsi que son type d'aprés les initiales suivantes:

Résistance → R
Capacité → C
Self → L
Source de tension → V
Source de courant → I

En ce qui concerne les sources de tension indépendantes, elles peuvent être continues DC ou dépendantes du temps. Dans ce cas deux sortes de sources sont disponibles: L'échelon ou le PULSE et la source sinusoîdale ou le SIN.

CONNECTEURS : ce sont deux entiers positifs N_d noeud duquel part le courant et N_a le noeud d'arrivée.

VALEUR: c'est un réel qui peut être suivi de puissance de 10.

En ce qui concerne les diodes et les transistors qui sont des composants non-linéaires, ils seront remplacés par leur modèle mis au point et stocké en bibliothèque, la syntaxe est la suivante:

NOM CONNECTEURS NOM DU TYPE

NOM : c'est une suite alpha-numérique dont le premier terme est alphabétique, il sera noté *D* pour la diode et *T* pour le transistor.

CONNECTEURS :

- N_1^+ N_2^- pour la diode (N_1^+ l'anode et N_2^- la cathode).
- N_C N_B N_E, respectivement les noeuds collecteurs, base et émetteur pour le transistor.

NOM DU TYPE : c'est une suite alpha-numérique dont le premier terme est la lettre *B*. Il permet de retrouver le schéma équivalent du composant dans la bibliothèque ainsi que les valeurs des paramètres.

Le caractère '*' placé en début de ligne indique un commentaire.

Le mot clé END indique la fin de déclaration du circuit.

V.1.3 Lecture des données- Processeur d'entrée

Lors de cette phase, le simulateur reçoit les données contenues dans le fichier d'entrée. Le compilateur *LDCE* (Langage de Description des Circuits Electroniques) effectue sur le fichier source des vérifications syntaxiques et sémantiques puis une fois la lecture terminée, le compilateur contient une description interne codée qui reflète celle du circuit et de ses paramètres [43] (voir annexe 6).

V.1.4 Processeur d'initialisation - Module de liaison LIEN

Ce module transforme la représentation du circuit fournie par le compilateur *LDCE* en une structure de données directement acceptable par le processeur d'analyse.

Pour cela, il doit remplacer les éléments complexes par leurs schémas équivalents qui existent en bibliothèque (voir annexe 6).

Cette dernière serait une suite d'ensemble de paramètres où chaque ensemble est relatif à un type de diode ou de transistor.

Le module LIEN génère une structure reconnue par le processeur d'analyse parmi ces structures fondamentales on distingue :

- La table MATDESC qui est une matrice de description du circuit où chaque élément est l'image d'une branche décrite du circuit en mémoire selon le tableau ci-dessous.

COMP	Na	Nd	COEF	VAL

- COMP : C'est un entier qui détermine le type du composant.
- Na, Nd : Ce sont deux entiers qui désignent respectivement les noeuds de départ et d'arrivée de la branche traitée.
- COEFF : Ce champ est significatif quand la source de tension ou de courant est dépendante, il fournit en réel la valeur du coefficient de dépendance.
- VAL : Valeur réelle du composant.

- Le vecteur VECTNOD est un vecteur qui contient les valeurs relatives des noeuds nommés par l'utilisateur. Il sera exploité ultérieurement par le langage de commande pour retrouver un noeud utilisateur.

- Le vecteur NEW-BR est un vecteur qui contient les noeuds des branches du circuit final à base de dipôles simples résultant de la transformation d'un circuit non-linéaire.

V.1.5 Processeur d'analyse - Résolution

Une fois le circuit décrit, l'utilisateur dispose d'un langage de commande qui lui permet de préciser le fonctionnement qu'il souhaite analyser.

Le processeur d'analyse reconnaît deux modes de fonctionnement correspondant aux commandes *CONT* et *TRAN* :

- Commande CONT - Analyse en continu

Cette commande réalise une analyse en continu, elle permet donc de déterminer le point de fonctionnement du circuit. Une telle analyse est en général effectuée pour initialiser une simulation en mode transitoire (voir annexe 6).

Le but final de l'analyse des circuits électrique est la résolution du système matriciel $A\ X=B$; et l'exploitation du vecteur X de toutes les variables du circuit.

Comme c'est le régime permanent, les variables ne dépendent pas du temps et par conséquent leurs dérivées s'annulent.

L'interprétation de ces lois au niveau de la matrice A, serait sa mise à jour à l'entrée en colonne pour chaque variable et sa mise en forme matricielle séquentiellement pour toute branche (entrée ligne).

La résolution matricielle se fait d'aprés l'algorithme de Crout implémenté sous forme de procédure appelée "SPARSE".

- Commande TRAN - Analyse en transitoire

Cette commande permet une analyse transitoire, ce qui détermine les variables électriques du circuit en fonction du temps pendant une durée donnée et pour des sources d'excitation prises à partir de conditions initiales. Celles-ci peuvent être soit précisées par l'utilisateur soit déduites du calcul d'un état stable correspondant à la valeur de la source d'excitation pour le temps initial.

Dans ce cas d'analyse les dérivées ne s'annulent pas, ce qui impliquera pour le processeur une étape supplémentaire par rapport au continu car il faudra passer par le procédé d'intégration .

Les paramètres à donner sont:

- Le temps initial : T_{IN}
- Le temps final : T_{MAX}
- Le pas de sortie : H

Dans ce cas d'analyse les dérivées ne s'annulent pas, ce qui impliquera pour le processeur une étape supplémentaire par rapport au continu car il faudra passer par le procédé d'intégration.

Le fonctionnement du module $TRAN$ peut être schématisé selon le synoptique donné dans la figure V.2.

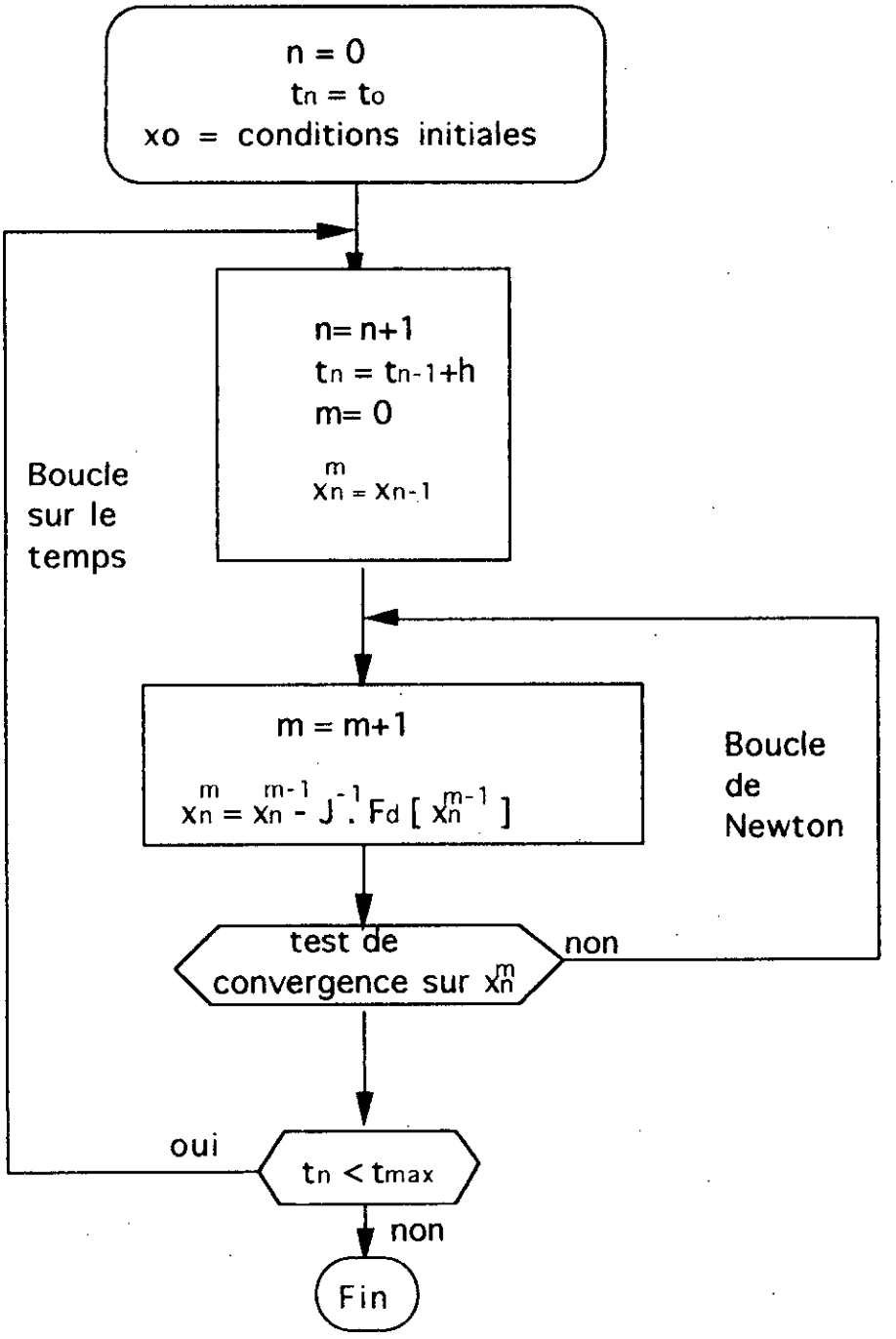


Figure 2 Algorithme de calcul du régime transitoire

V.1.6 Processeur de sortie

Comme la méthode choisie pour la formulation des données d'entrée du circuit à simuler, l'utilisateur dispose donc à la sortie de toutes les variables vu qu'elles ont été regroupées dans le vecteur X.

Ce dernier pourra être: un courant de branche, un potentiel d'un noeud ou une tension de branche.

La commande qui permet la sortie de ces valeurs est la commande *PRINT*.

Ce chapitre a été élaboré en deux parties distinctes:

La première partie a consisté en une présentation complète du programme élaboré sous une forme modulaire où les différents modules s'enchainent de manière logique.

Ainsi ont été mis au point:

1/ Un processeur d'entrée dans lequel la structure des données est construite.

2/ Un processeur d'initialisation qui grâce à un module d'adaptation permet la mise en équation.

2/ Un processeur d'analyse dans lequel la résolution des équations s'effectue.

3/ Un processeur de sortie qui permet l'exploitation des résultats de la simulation.

Comme la méthode du tableau a été choisie comme moyen de formulation des donnée
Les listings des logiciels développés pour chaque module sont donnés en annexe 6.

Dans la seconde partie, nous verrons quelques exemples de simulation de circuits en toutes technologies afin de montrer les possibilités du programme élaboré.

V.2 APPLICATIONS DU PROGRAMME

Cette partie consiste en quelques exemples de simulation de circuits qui illustrent les possibilités du programme. On y présente successivement des résultats obtenus lors de la simulation d'un inverseur, schéma élémentaire à partir duquel toute porte logique pourra être déduite.

Les exemples qui suivront correspondent à la simulation de portes logiques simples dans les différentes technologies bipolaires telle que: la TTL, l'ECL et la technologie I^2L [44,45].

V.2.1 SIMULATION DE L'INVERSEUR

Nous commençons l'application avec l'inverseur élément de base de la majorité des systèmes logiques.

Le symbole logique ainsi qu'un schéma simple mais pratique de l'inverseur sont donnés en figure V.1a.

La description du fonctionnement transitoire du montage a été largement abordée au premier chapitre et dans ce qui suit nous ne donnerons que le résultat de la simulation en figure V.1b.

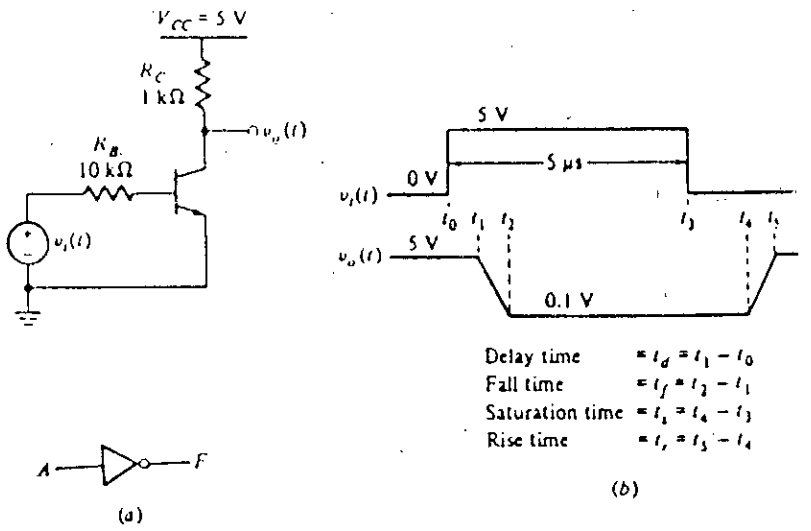


Figure V.1a Description du montage inverseur

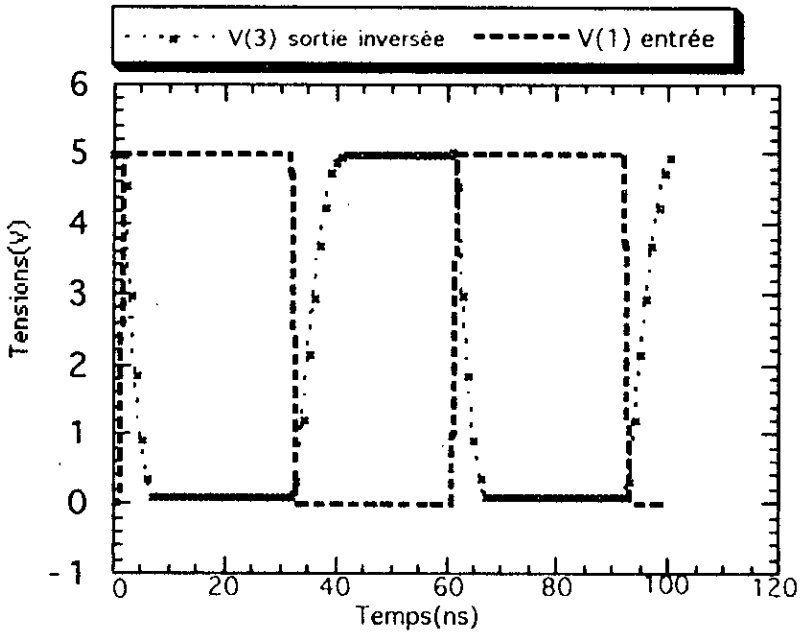


Figure V.1b Résultat de la simulation de l'inverseur

V.2.2 SIMULATION D'UNE PORTE NAND TTL

La technologie TTL (Transistor Transistor Logique) est venue pallier à l'inconvénient majeur de la technologie DTL (Diode Transistor Logique) qui comporte un grand nombre de diodes [41], vu que chaque diode (qui est en réalité un transistor monté en diode) exige pour sa réalisation un caisson isolé, la surface occupée par l'élément logique est assez grande.

Le réseau de diodes logiques et de décalage correspond à la structure d'un transistor, donc à l'aide d'un transistor multi-émetteur ce réseau de diodes sera évité.

La TTL assume la même fonction logique la DTL qui est la fonction NAND.
L'avantage majeur de l'utilisation du transistor multi-émetteur est que le délai de propagation de la porte TTL est plus réduit.

Nous avons considéré pour la simulation, la porte NAND TTL standard de la série logique 54 / 74 qui possède deux entrées A et B.

Le schéma du circuit ainsi que le résultat de la simulation sont donnés en figure V.2a et figure V.2b .

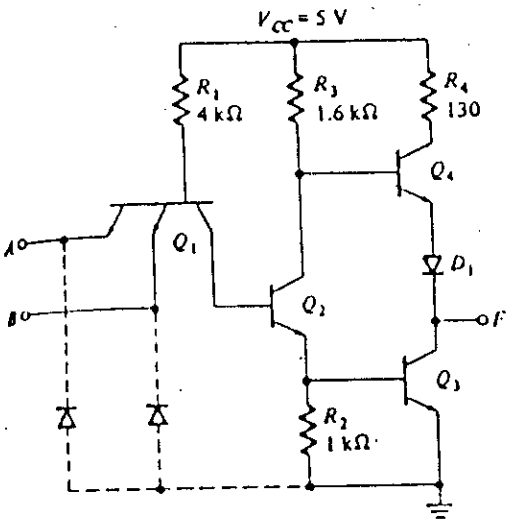


Figure V.2a Description de la porte NAND TTL de la série 54/74

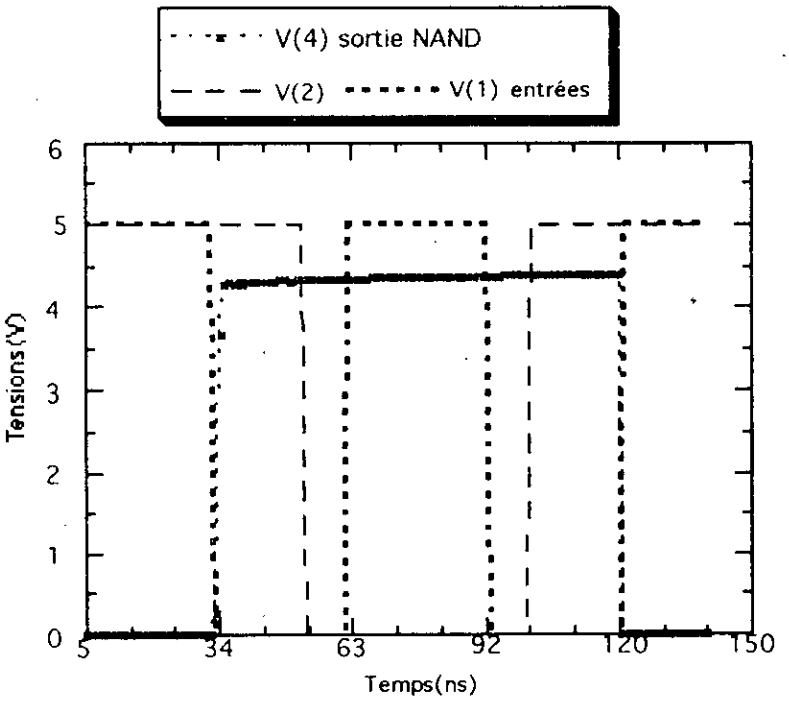


Figure V.2b Résultat de la simulation de la porte NAND TTL de la série 54/74

V.2.3 SIMULATION D'UNE PORTE OR-NOR ECL

Comme les transistors sont conduits jusqu'à la saturation, la TTL souffre d'une limitation fondamentale dans sa vitesse d'opération.

La technologie ECL (Emitter Coupled Logique) est venue corriger cela en maintenant le transistor dans sa région active et par conséquent un faible entrainement à l'entrée sera suffisant pour mener le transistor soit au blocage soit à la conduction.

La famille logique ECL est basée autour d'un étage d'entrée différentiel dont l'une des branches est connectée à une tension de référence Vr, l'autre branche reçoit la tension d'entrée Vi tel que Vi << Vr (voir figure V.3a).

Les transistors Q_1 et Q_3 sont les transistors d'entrée par contre Q_2 représente le transistor de référence qui fournit la tension de référence Vr.

Les sorties sont prises au collecteur à travers des émetteurs suiveurs qui fournissent une faible impédance à la sortie et assurent une bonne sortance de la porte logique.

La porte ECL fournit une sortie NOR et une autre OR, le schéma du circuit ainsi que le résultat de la simulation sont donnés en figure V.3b et en figure V.3c.

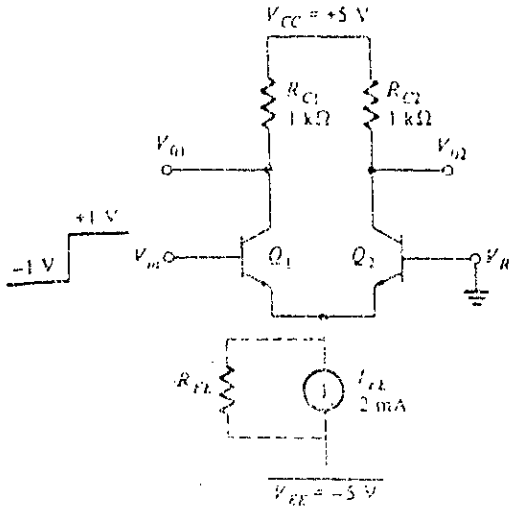


Figure V.3a : Description de l'étage d'entrée différentiel d'une porte ECL

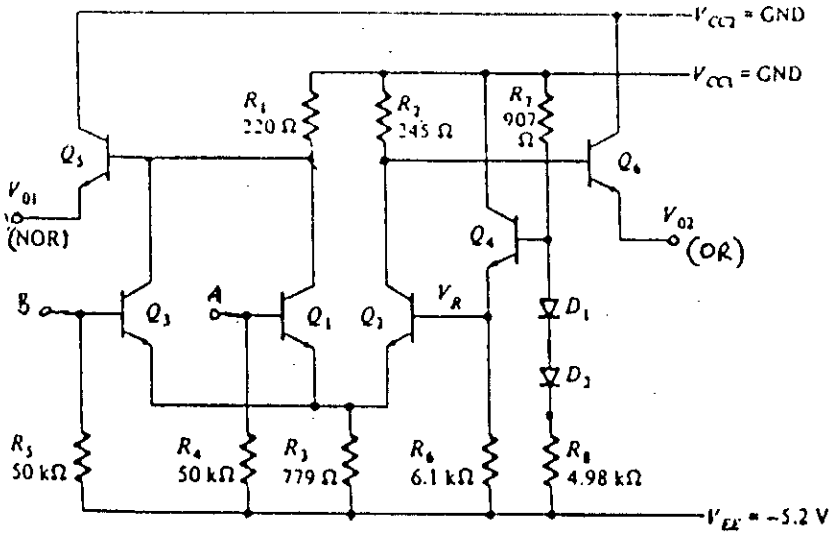


Figure V.3b Description d'une porte OR/NOR ECL

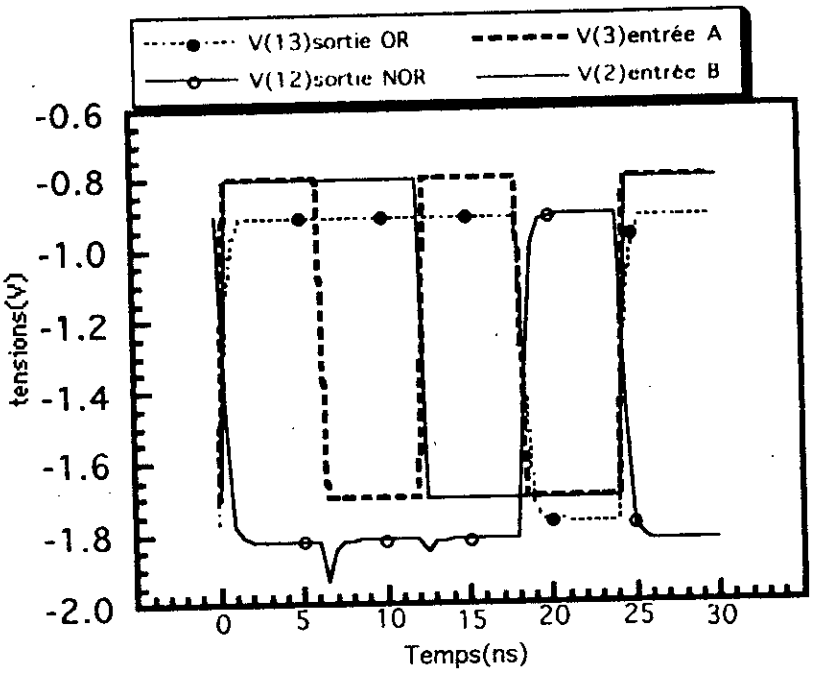


Figure V.3c Résultat de la simulation de la porte OR/NOR ECL

V.2.4 SIMULATION D'UN INVERSEUR I^2L

Les circuits TTL et ECL précédemment décrits sont utilisés dans les circuits intégrés SSI (Small Scale Integration) ou MSI (Medium Scale Integration), mais leur utilisation dans la LSI (Large Scale Integration) est généralement évitée à cause de leur large surface relative et leur puissance de dissipation (elle est de l'ordre de 10 mW pour la TTL, et de 30 mW pour l'ECL).

En effet si un circuit comporte 500 portes TTL, il dissipera une puissance de 1W et couvrira une surface de 25 mm2.

La technologie bipolaire I^2L (Integrated Injection Logique) a répondu à la concurrence faite dans le domaine de la LSI par les transistors MOSFET.

Un circuit logique du type I^2L consiste en trois transistors multi-collecteurs connectés ensemble.

A l'entrée de chaque transistor, il y a une source de courant I0 appelée l'injecteur de courant.

Ce courant provient du collecteur d'un transistor de type PNP qui est connecté à la base du transistor de charge de type NPN.

La technologie I^2L est aussi connue sous le nom de MTL (Merged Transistor Logique), car que les régions de même type ont été réalisées à partir d'une région commune, comme par exemple le collecteur du transistor PNP (l'injecteur) et la base du transistor NPN (la charge).

La simulation d'une porte inverseuse I^2L dont la description donnée en figure V.4a a été effectuée et le résultat est présenté en figure V.4b.

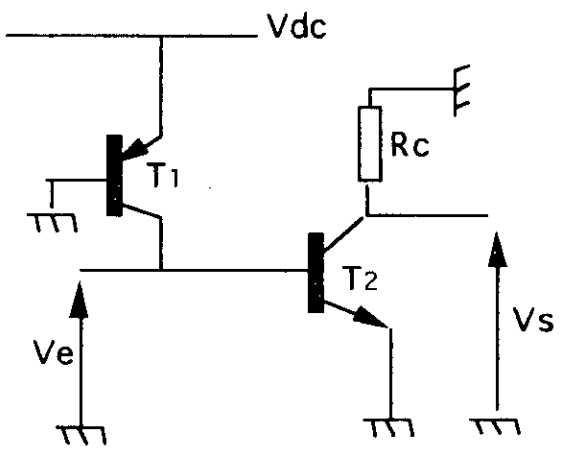


Figure I.4a Description d'un inverseur I^2L

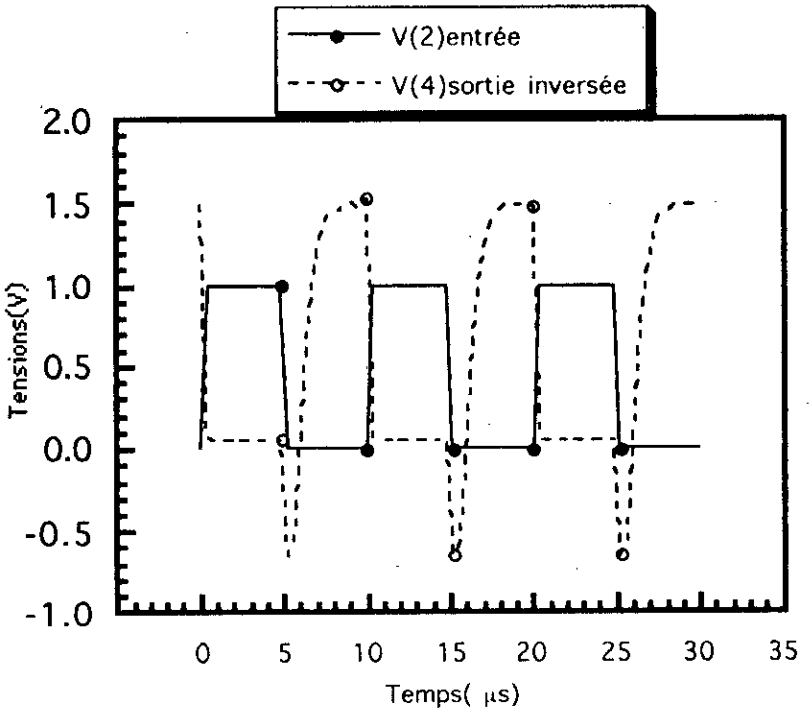


Figure V.4b Résultat de la simulation de l'inverseur I^2L

Dans tous les circuits logiques qui viennent d'être décrits, à tout moment la sortie est directement reliée à l'entrée par une certaine combinaison logique. Cette classe de circuits est connue sous le nom de circuits logiques combinatoires.

Des exemples communs de ces types de circuits sont les portes logiques simples NOR, NAND, etc...

Il existe une autre classe de circuits dits circuits logiques séquentiels dans lesquels les sorties sont affectées par les données d'entrée présentes et précédentes.

Une caractéristique de ces circuits séquentiels est qu'un ou plusieurs noeuds sont intentionnellement connectés aux entrées afin de fournir une boucle de retour positive ou une regénération.

Dans ce qui suivra, un exemple de ces circuits sera présenté avec des transistors bipolaires.

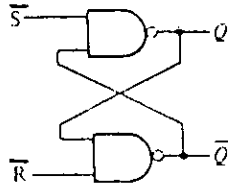
V.2.5 SIMULATION D'UNE BASCULE RS

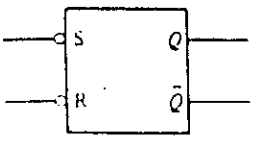
Dans la figure V.5a une bascule RS est représentée avec deux entrées à porte NAND. Une des entrées de cette dernière est utilisée pour un bouclage à la sortie de l'autre porte NAND, pendant que l'entrée fournit le passage d'un état stable à un autre.

Les deux sorties Q et $\overline{Q}$ sont complémentaires par définition, le latch est à l'état SET avec le 0 logique à l'entrée S et à l'état RESET.

La bascule répond à une activation des entrées à l'état bas.

La table de vérité et le schéma de la bascule sont donnés en figure V.5a et le résultat de la simulation donné en figure V.5b.





$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$
1	1	Q	$\overline{Q}$
0	1	1	0
1	0	0	1
0	0	X	X

Figure V.5a Description de la bascule RS

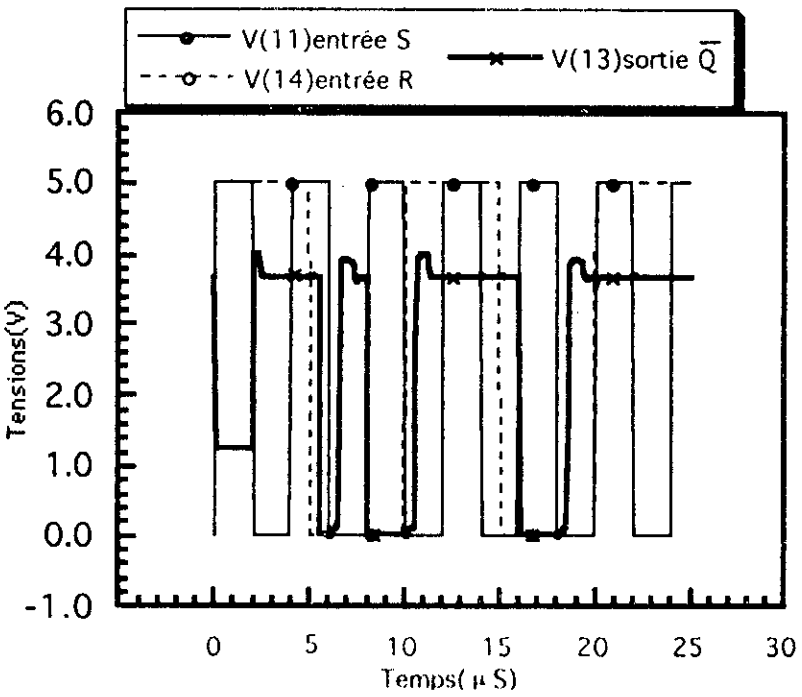


Figure V.5b Résultat de la simulation de la bascule RS (sortie $\overline{Q}$)

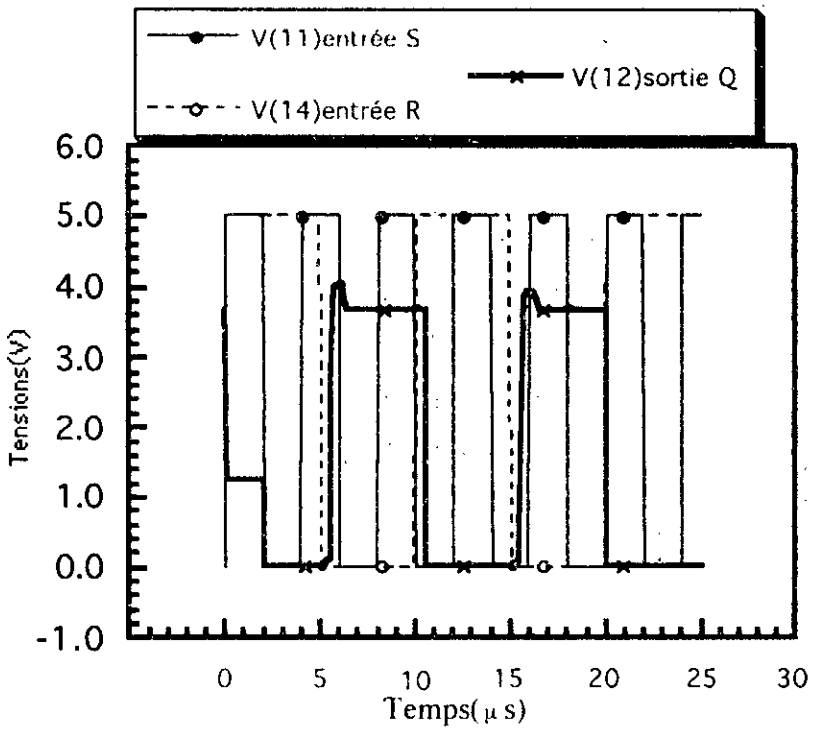


Figure V.5c Résultat de la simulation de la bascule RS (sortie Q)

V.2.6 SIMULATION DU TRIGGER DE SCHMITT

Le trigger de Schmitt est un circuit utilisé comme détecteur de niveaux, qui permet la remise en forme des signaux à front dégradés et pour transformer des signaux sinusoidaux en signaux carrés.

Les deux transistors peuvent passer d'un de leurs deux états stables à l'autre quand le potentiel d'entrée franchit un certain seuil.

Ils basculeront de ce second état à l'état initial quand le potentiel d'entrée franchira en sens inverse un autre seuil.

Le schéma du circuit, sa description ainsi que le résultat obtenu lors de la simulation d'un signal sinusoidal transformé en signal carré sont donnés en figure V.7a et figure V.7b.

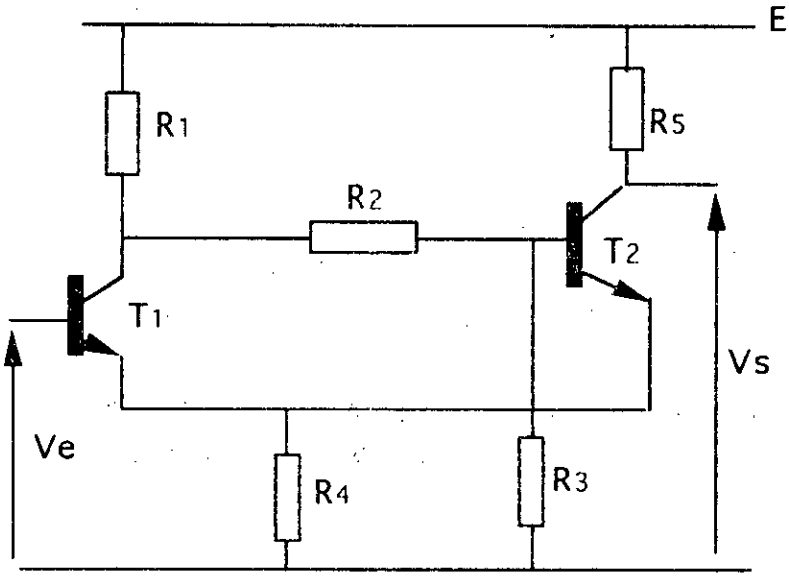


Figure V.6a Description du montage du trigger de Schmitt à simuler

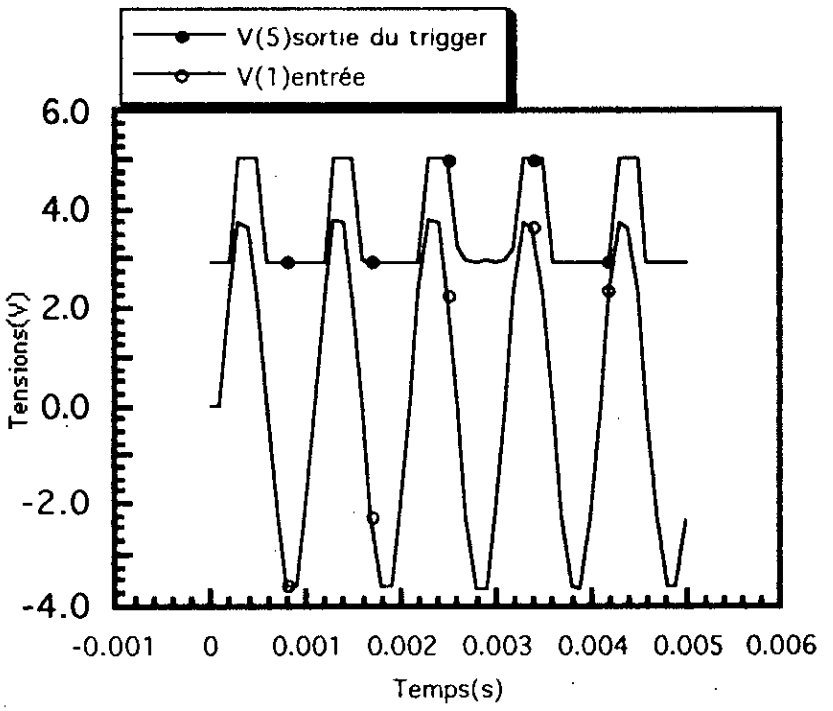


Figure V.6b Résultat de la simulation du trigger de Schmitt

CONCLUSION

Nous avons tout au long de cette thèse présenté le programme sous son aspect informatique et sous l'angle des performances du processeur d'analyse.

Sa réalisation a été effectuée sur la station VAX sous le système VMS. Ainsi ont été mis au point:

- Le langage de description LDCE.
- Le module de liaison LIEN.
- Le simulateur en mode continu CONT.
- Le simulateur en mode transitoire TRAN.
- l'éditeur de résultats PRINT.

Cette structuration modulaire fournit un atout supplémentaire à la diffusion et au développement du programme et de plus assurera dans l'avenir une maintenance informatique assez aisée.

Bien que les résultats obtenus soient satisfaisants, l'amélioration du programme peut être envisagée sur plusieurs plans :

• Sur le plan logiciel

Dans le but d'améliorer le temps d'exécution, il serait possible de reécrire en assembleur certains sous-programmes sollicités plus fréquemment.

• Sur le plan analyse

Dans tous les programmes de simulation, plus de la moitié du temps est consacré à l'évaluation des expressions qui décrivent les composants des éléments qui constituent le circuit à analyser.

Le souci majeur est la diminution du coût de la simulation par l'amélioration des algorithmes numériques de résolution des équations.

Au niveau des méthodes d'intégration, par exemple choisir celles qui permettent une adaptation du pas sur le temps et cela grâce à un test de précision comme la méthode de Gear par exemple.

- Sur le plan utilisateur

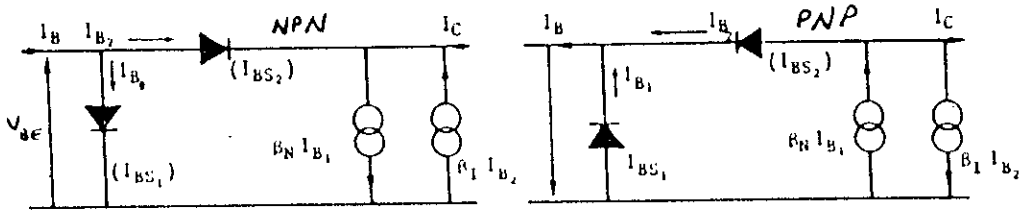
Afin que la description du circuit soit la plus concise possible, une extension du langage LDCE doit être envisagée: cela facilite surtout la description des macro-circuits grâce à la notion de sous-circuits.

Dans le même but, une expansion de la bibliothèque de composants serait propice à une plus large simulation de circuits.

ANNEXES

Annexe 1 Ecriture des expressions des courants du modèle d'Ebers-Moll en émetteur commun

Il est possible de trouver un modèle émetteur-commun, il suffirait de retourner l'élément tripolaire de la base-commune en émetteur-commun. Cependant, il est intéressant d'exprimer les sources liées en fonction du courant d'entrée I_b et de placer ces sources à la sortie afin de mieux indiquer le tansfert direct (voir figure ci-dessous).



$$I_E = I_{ES} \exp\left(\frac{Vbe}{U_T} - 1\right) - \alpha_I I_{CS} \exp\left(\frac{Vbc}{U_T} - 1\right)$$

$$I_C = \alpha_N I_{ES} \exp\left(\frac{Vbe}{U_T} - 1\right) - I_{CS} \exp\left(\frac{Vbc}{U_T} - 1\right)$$

avec : $$\alpha_N I_{ES} = \alpha_I I_{CS}$$

$$I_B = I_S\left(\frac{1 - \alpha_N}{\alpha_N}\right) \exp\left(\frac{Vbe}{U_T} - 1\right) + \frac{I_S}{\alpha_I}(1 - \alpha_I) \exp\left(\frac{Vbc}{U_T} - 1\right)$$

$$I_C = I_S \exp\left(\frac{Vbe}{U_T} - 1\right) - \frac{I_S}{\alpha_I} \exp\left(\frac{Vbc}{U_T} - 1\right)$$

Comme : $\beta_N = \frac{\alpha_N}{1 - \alpha_N}$ $\beta_I = \frac{\alpha_I}{1 - \alpha_I}$

On obtient : $I_{BS1} = \frac{I_S}{\beta_N}$ $I_{BS2} = \frac{I_S}{\beta_I}$

Ce qui permet de vérifier toujours la relation de réciprocité pour un montage base-commune :

$$\beta_N I_{BS1} = \beta_I I_{BS2}$$

On en déduit les équations d'état qui deviennent dans ce cas :

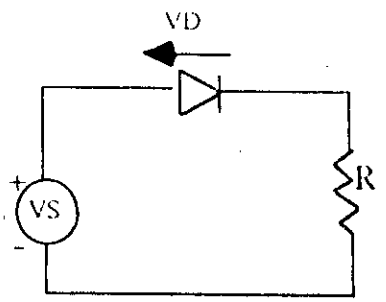
$$I_B = I_{BS1} \text{Exp}\left(\frac{Vbe}{U_T} - 1\right) + I_{BS2} \text{Exp}\left(\frac{Vbc}{U_T} - 1\right)$$

$$I_C = \beta_N I_{BS1} \text{Exp}\left(\frac{Vbe}{U_T} - 1\right) + (\beta_I + 1) I_{BS2} \text{Exp}\left(\frac{Vbc}{U_T} - 1\right)$$

Annexe 2 Linéarisation du modèle de la diode et de la capacité

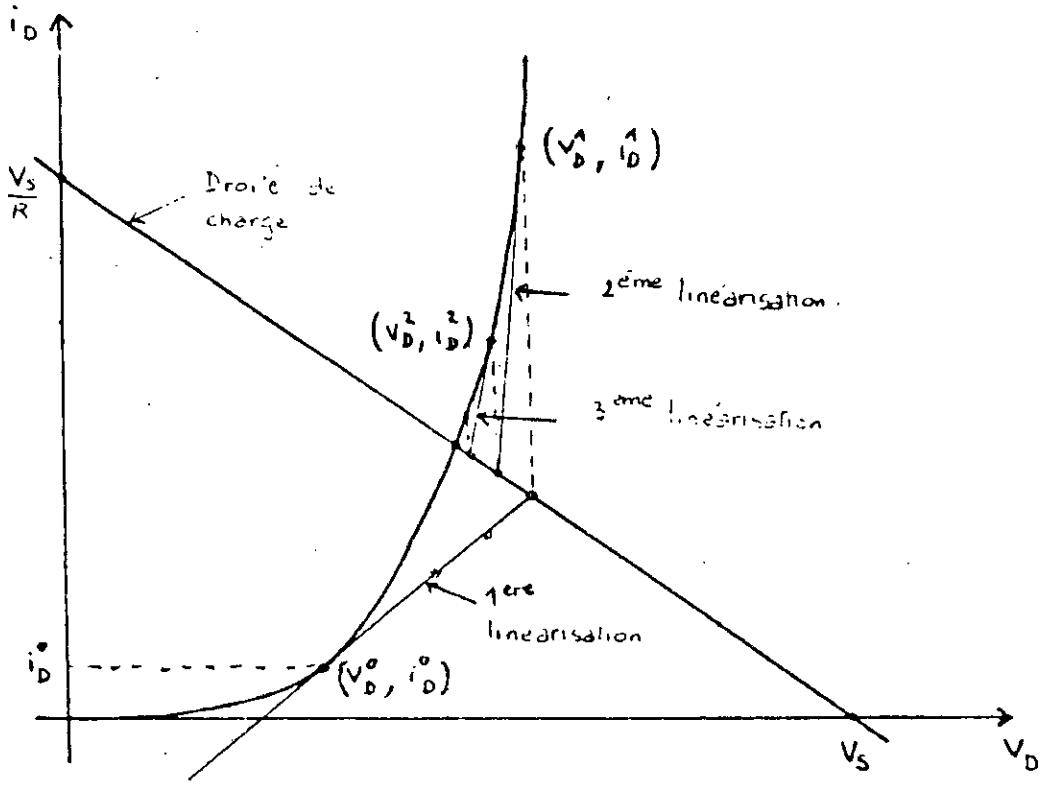
Annexe 2.1 Linéarisation de la caractéristique de la diode

Prenons comme exemple le circuit de la figure ci-dessous:



Nous voulons linéariser la caractéristique de la diode autour du point de fonctionnement.

Du point de vue analytique, la linéarisation de l'équation de la caractéristique de la diode $I_D = f(V_D)$ est obtenue en faisant un développement en série de Taylor au voisinnage du point V_D^o.



$$I_D^{m+1,m+1} = I_D^{m+1,m} + G^{m+1,m}.\ (V_D^{m+1,m+1} - V_D^{m+1,m})$$

$$I_D = Is\,[\exp(\lambda V_D^{o}) - 1] + \lambda\, Is \exp(\lambda V_D^{o})(V_D - V_D^{o})$$

$$I_D = \frac{Vs - V_D}{R} = \frac{Vs - V_D^{o} - \Delta V_D^{o}}{R} \qquad \text{avec} \quad \Delta V_D^{o} = (V_D - V_D^{o}).$$

Ainsi :

$$Is\,[\exp(\lambda V_D^{o}) - 1] + \lambda\, Is \exp(\lambda V_D^{o})\,\Delta V_D^{o} = \frac{Vs - V_D^{o} - \Delta V_D^{o}}{R}$$

Nous en déduisons ΔV_D^{o}, en posant $V_D^{1} = V_D = V_D^{o} + \Delta V_D^{o}$

Une seconde linéarisation au point (I_D^{1} , V_D^{1}) donne ΔV_D^{1} et ainsi de suite. On obtiendra pour chaque nouvelle linéarisation ΔV_D^{2} , ΔV_D^{3}, ..., ΔV_D^{m} .

$$\Delta V_D^{m} = V_D^{m+1} - V_D^{m} = \frac{-R\ Is\,(\exp(\lambda V_D^{m}) - 1) - V_D^{m} + Vs}{1 + \lambda R Is \exp(\lambda V_D^{m})}$$

Nous remarquons maintenant qu'il n'est plus nécessaire de trouver une représentation graphique des itérations. Par contre, nous pouvons simplement écrire et résoudre un ensemble d'équations algébriques en utilisant les lois de kirchoff relatives aux tensions et aux courants (KCL et KVL) avec une extension de la méthode de Newton.

L'équation générale qui décrit la linéarisation de la caractéristique non-linéaire de la diode se traduit par:

$$I_D^{m+1} = I_D^{m} + \left(\frac{\delta I_D}{\delta V_D}\right)_{V_D = V_D^{m}} (V_D^{m+1} - V_D^{m})$$

où m représente le nombre d'itérations.

Cette équation nous permet d'établir le modèle de la diode linéarisé de la façon suivante:

$$G_D^{m} = \left(\frac{\delta I_D}{\delta V_D}\right)_{V_D = V_D^{m}}$$

$$I_D^{m+1} = (I_D^{m} - G_D^{m}\ V_D^{m}) + G_D^{m}\ V_D^{m+1}$$

Nous observons maintenant que le nouveau modèle est linéaire, il est appelé modèle "compagnon" ; le circuit peut donc être résolu par la méthode d'analyse aux noeuds.

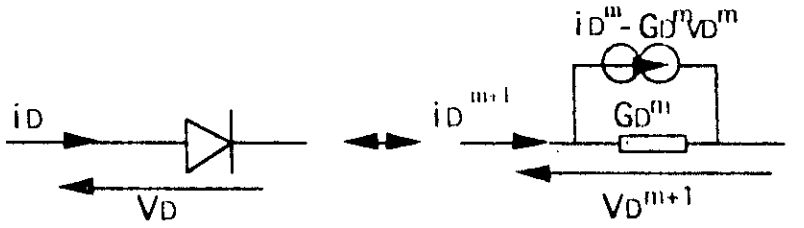


Figure A2.1 Schema équivalent du modèle de la diode

Annexe 2.1 Linéarisation du modèle de la capacité

Afin d'approximer la dérivée, nous utiliserons la formule de Backward-Euler :

$$\left.\frac{dV}{dt}\right|_{t=t^{n+1}} = \frac{V^{n+1} - V^n}{t^{n+1} - t^n} \quad \text{si le pas est constant } t^{n+1} - t^n = T.$$

Quand on applique ce principe au modèle de la capacité dans lequel le courant est proportionnel à la dérivée de la tension à ses bornes:

$$I_c = C \frac{dV}{dt}$$

on obtient les expressions suivantes par linéarisation de la dérivée :

$$I_c = C \frac{V^{n+1} - V^n}{T} = \frac{C}{T} V^{n+1} - \frac{C}{T} V^n$$

En écrivant cette expression sous la forme séparée ci-dessous, on peut établir le schéma suivant du modèle de la capacité linéarisée.

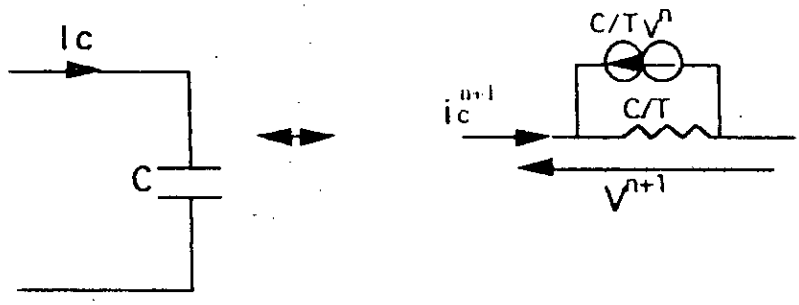


Figure A2.2 Schéma équivalent du modèle de la capacité

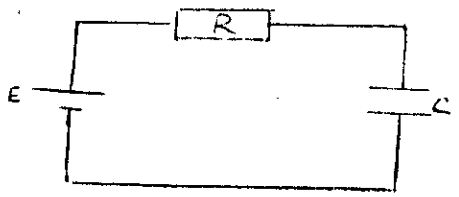
Annexe 3 Programme de simulation de l'inverseur pour la validation du modèle d'Ebers-Moll numérisé

```
510 END IF
520 N1=N
525 MOVE N1,V1
530 V1f=V1
535 V2=0.4
540 V2f=V2
550 Vt=V1
560 FOR N1=N TO N+200
590    T=1.0E-6
600    G2=L*Is*EXP(L*V2)
610    I2=Is*(EXP(L*V2)-1)
620    Cde=Tb*G2
622    I1=Is*(EXP(L*(V2-V1))-1)
624    G1=L*Is*EXP(L*(V2-V1))
625    Cdc=Tb*G1
630    Ctc=C0/(1-(V2-V1)/Fi)^0.5
635    Cte=C0/(1-V2/Fi)^0.5
640    A(1,1)=1/Rc+(Cdc+Ctc)/T+G1
650    A(1,2)=-(Ctc+Cdc)/T+An*G2-G1
660    A(2,1)=-1/Rc-A1*G1
670    A(2,2)=-1/Rb-(Cte+Cde)/T+A1*G1-G2
680    B(1)=E/Rc-An*(I2-G2*V2)+I1-G1*(V2-V1)-(Cdc+Ctc)/T*(V2f-V1f)
690    B(2)=-Vb/Rb-A1*(I1-G1*(V2-V1))-E/Rc+I2-G2*V2-(Cte+Cde)/T*V2f
710    D=INV(A)
720    X=D MPY B
730    V1=X(1)
740    V2=X(2)
750    IF (V1-Vt)^2<1.0E-8 THEN
760       V1f=V1
770       V2f=V2
780       Vt=V1
790       DRAW N1,V1
800    NEXT N1
810    HOME
820 ELSE
830    Vt=V1
840    GO TO 600
850 END IF
860 Vb=-2
865 V2=0.5
870 Vt=V1
875 N=N1
880 GO TO 560
890 END
```

```
90 PAGE
100 PAGE
101 VIEWPORT 0,80,0,80
102 WINDOW 1,500,0,8
105 AXIS 10,1
106 DIM A(2,2),B(2),X(2),D(2,2)
110 C0=1.0E-9
130 T=1.0E-9
140 Rc=1000
150 Rb=5000
160 Tb=1.0E-5
170 Te=1.0E-5
180 Is=7.0E-12
190 L=40
200 An=0.987
205 Fi=0.7
210 Ai=0.8
220 Vb=4
230 E=4
240 V1=4
250 V2=-2
260 V1f=V1
270 V2f=V2
280 N=1
290 IF V2>0 THEN
300     PRINT "FIN DU REGIME BLOQUE"
310     GO TO 530
330 ELSE
340     Ctc=C0/(1-(V2-V1)/Fi)^0.5
350     Cte=C0/(1-V2/Fi)^0.5
360     A(1,1)=Ctc/T+1/Rc
370     A(1,2)=-Ctc/T
380     A(2,1)=-Ctc/T
390     A(2,2)=1/Rb+(Ctc+Cte)/T
400     B(1)=Ctc/T*(V1f-V2f)+E/Rc
410     B(2)=-(Ctc/T)*V1f+V2f/T*(Ctc+Cte)+Vb/Rb
420     D=INV(A)
430     X=D MPY B
440     V1=X(1)
450     V2=X(2)
460     V1f=V1
470     V2f=V2
480     DRAW N,V
490     N=N+1
500     GO TO 290
510 END IF
```

Annexe 4 Exemple de mise en équation d'un circuit RC

Soit le circuit suivant :



On écrira les équations de ce circuit en appliquant les différentes lois KVL et KCL ainsi que les relations constitutives:

1°/ Loi des noeuds:

noeud 1: $i_1 + i_2 = 0$

noeud 2 : $-i_2 + i_3 = 0$

Le système matriciel peut alors s'écrire:

$$\begin{bmatrix} +1 & +1 & 0 \\ 0 & -1 & +1 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

c'est à dire : $A.i = 0$, A étant la matrice d'incidence

2°/ Loi des mailles:

branche b1: $v_1 = u_1$ => $v_1 - u_1 = 0$

branche b2 : $v_1 - v_2 = 0$ => $v_1 - v_2 - u_2 = 0$

branche b3 : $v_2 = u_3$ => $v_2 - u_3 = 0$

On obtiendra le système matriciel suivant:

$$\begin{bmatrix} +1 & 0 \\ +1 & -1 \\ 0 & +1 \end{bmatrix} \begin{bmatrix} v_1 & v_2 \end{bmatrix} - \begin{bmatrix} +1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & +1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

On voit bien que : $B.V - I.U = 0$ et $B = A^t$

3°/ relations constitutives:

Elles expriment la loi d'Ohm sous toutes ses formes dans les branches du circuit et s'appliquent aux composants de base que reconnait le simulateur:

- La tension aux bornes d'une résistance R parcourue par un courant i :

$$u = R.i$$

- La charge stockée dans une capacité C ayant une tension u :

$$Q = C \cdot u$$

- La tension débitée par une source de tension E :

$$E = u$$

Le système matriciel global sera alors le suivant:

V_1	V_2	i_1	i_2	i_3	v_1	v_2	v_3	q_3
		$+1$	$+1$					
			-1	$+1$				
$+1$					-1			
$+1$	-1					-1		
	$+1$						-1	
					-1			
			$R(v_2)$			$i_2 \dfrac{\partial R}{\partial v_2} - 1$		
							C	-1
				1				$-\alpha_0$

Annexe 5 Fichier source de description des exemples considérés pour la simulation

```
*inverseur simple
VCC 4 0 DC 5
T1 3 2 0 BT
R1 3 4 1K
R2 1 2 10K
V1 1 0 PULSE (0 5 1N 1N 1N 30N 60N)
TRAN 1N 100N
PRINT V(3) V(1)
END
```

```
*NAND TTL
VCC 6 0 DC 5
T1 10 7 1 BT
T2 5 10 3 BT
T3 4 3 0 BT
T4 9 5 8 BT
T5 10 7 2 BT
D1 8 4 BD
D2 0 2 BD
D3 0 1 BD
R1 6 7 4K
R2 3 0 1.6K
R3 6 5 1.6K
R4 6 9 0.13K
V1 1 0 PULSE (0 5 1N 1N 1N 30N 60N)
V2 2 0 PULSE (0 5 1N 1N 1N 50N 100N)
TRAN 1N 140N
PRINT V(4), V(2),V(1)
END
```

```
*OR/NOR ECL
vee 7 0 dc -5.2
v14 14   0 dc -2
T1  1 3  4 BT
T2  6  8 4 BT
T3  1  2 4 BT
T4  0  9 8 BT
T5  0  1 12 BT
T6  0  6 13 BT
R1 0 1  .22k
R2 0 6  0.245k
R3 7 4  0.779k
R4 3 7  50k
R5  2 7  50k
R6  8 7  6.1k
R7 9 0  0.907k
R8 11 7  4.98k
R9 13 14  0.05k
R10 12 14  0.05k
D1 9 10 BD
D2 10 11 BD
VA  3 0 pulse (-1.7  -0.8  0.1N  0.1N  0.1N 6N 12N)
VB  2 0 pulse (-1.7  -0.8  0.1N  0.1N  0.1N 12N 24N)
TRAN  0.5N  30N
PRINT  v(13),V(12),V(3),V(2)
END



*OR 12L
vdc 1 0  dc 1.5
T1  2 0 1  BT
T2  4 2 0 BT
R1  1 4  50k
VA  2 0 pulse (0 1  0N  1N  1N 5u 10u)
TRAN 400N 30u
PRINT  v(2) V(4)
END
```

```
*Bascule
vcc 8 0 dc -5.2v
T1  2 1 9 BT
T2  2 4 9 BT
T3  3 2 5 BT
T4 3 6 4 BT
T5 6 5 8 BT
T6 6 7 10 BT
R1  1 12 .450k
R2 4 3  0.640k
R3 6 8  0.450k
R4 10  3  0.450k
R5 4  8  0.640k
R6  2 11 0.450k
VR  1 0 pulse (-1 1  0.1N  0.1N  0.1N 10N 20N)
VS  2 0 pulse (-1 1  0.1N  0.1N  0.1N 20N 40N)
TRAN  1N 60N
PRINT  v(8),v(3),v(1),v(2)
END
```

```
*Trigger de Schmidt
VCC 6 0 DC 5
T1 2 1 4 BT
T2 5 3 4 BT
R1 6 2 6.8K
R2 2 3  12K
R3 3 0 27K
R4 4 0 3K
R5 6 5 2.7K
Ve 1 0 sin(0 4 1KHZ 0.1MS 0)
TRAN 0.1MS 5MS
PRINT  V(5) V(1)
END
```

Annexe 6 Listings des différents modules du simulateur

```
[INHERIT ('VIDEO.PEN','PROJET.PEN')]
MODULE GESBIB(INPUT,OUTPUT);
[GLOBAL]PROCEDURE  GESBIS;
 (*-------------------------------------------------------------------------
 (*     PROGRAMME DE GESTION DES BIBLIOTHEQUES UTILISEES PAR SICAL
 (*     ==========================================================
 (*-------------------------------------------------------------------------
VAR
 TYPCOMP,CARTRAIT,CAR:CHAR;
 NOM,PARAM,VALUE   :T1;
 TRANSISTOR : TRANSIS;
 DIODE   : DIO;
 VAL:REAL;
 K : INTEGER;
 VECTDIO : ARRAY[1..5] OF T1;
 VECTRAN : ARRAY[1..25] OF T1;
 I,NBCAR : INTEGER;
 ENTREE,SORTIE : T1;
 COMPLEX : TEXT;
 FTRAN : FILE OF TRANSIS;
 FDIO  : FILE OF DIO;
 PROCEDURE DESCRIPTION;EXTERNAL;
BEGIN
 (*     OUVERTURE DU FICHIER DES TRANSISTORES    *)
 [-----------------------------------------]
 OPEN(FILE_VARIABLE := FTRAN,
      FILE_NAME     := 'TRAN.DAT',
      HISTORY       := OLD,
      ACCESS_METHOD := KEYED,
      RECORD_TYPE   := FIXED,
      ORGANIZATION  := INDEXED,
      ERROR         := CONTINUE);
 OPEN(FILE_VARIABLE := FDIO,
      FILE_NAME     := 'DIODE.DAT',
      HISTORY       := OLD,
      ACCESS_METHOD := KEYED,
      RECORD_TYPE   := FIXED,
      ORGANIZATION  := INDEXED,
      ERROR         := CONTINUE);
 (*     OUVERTURE DU FICHIER DES DIODES          *)
 [-----------------------------------------]

CAR := '0';
WHILE (CAR ='0') DO
BEGIN
TTINIT;EFFACETOUT;POSIT(5,5);SOULIGNER;
TTWRITE('VOUS VOULEZ TRAITER UN :');POSIT(7,10);NORMAL;
TTWRITE('1_ DIODE.');POSIT(8,10);
TTWRITE('2_ TRANSISTOR.');POSIT(9,10);
TTWRITE('3_ CIRCUIT COMPLEXE.');POSIT(11,5);
TTWRITE('TAPEZ LE NUMERO DU TYPE DU COMPOSANT A TRAITER');TTREADV(TYPCOMP
 EFFACETOUT;POSIT(5,5);SOULIGNER;
TTWRITE('VOUS VOULEZ FAIRE UNE :');POSIT(7,10);NORMAL;
TTWRITE('1_ CREATION.');POSIT(8,10);
```

```
TTWRITE('2_ MISE A JOUR.');POSIT(9,10);
TTWRITE('3_ SUPPRESSION.');POSIT(11,5);
TTWRITE('TAPEZ LE NUMERO DU TYPE DE TRAITEMENT DESIRE');TTREADNE(CARTRAIT);
CASE TYPCOMP OF
'1':
    CASE CARTRAIT OF
    '1':BEGIN
        EFFACETOUT;
        POSIT(2,10);
        SOULIGNER;
        REVERSE;
        TTWRITE('CREATION D"UN TYPE DE DIODE');
        NORMAL;
        POSIT(4,5);
        TTWRITE('DONNEZ LE NOM DU TYPE DE LA DIODE CREE :');
        POSIT(4,45);
        TTREADN(NOM,4,40);(* PARAMETRES DIODES A TROUVER*)
        DIODE.NOM:=NOM;
        NORMAL;
        POSIT(6,4);
        TTWRITE('PARAMETRES DE LA DIODE :');
        POSIT(8,4);
        TTWRITE('VALEUR DE LA SOURCE :');
        TTREADN(VALUE,6,30);
        CONVERTIR(VAL,VALUE);
        DIODE.PARAMETRE[1]:=VAL;
        POSIT(8,4);
        TTWRITE('VALEUR DE LA RESISTANCE :');
        TTREADN(VALUE,8,35);
        CONVERTIR(VAL,VALUE);
        DIODE.PARAMETRE[2]:=VAL;
        WRITE(FDIO,DIODE);
        END;
    '2':BEGIN
        EFFACETOUT;
        POSIT(2,10);
        REVERSE;
        SOULIGNER;
        TTWRITE('MISE A JOUR D"UNE DIODE.');POSIT(4,5);
        NORMAL;
        TTWRITE('DONNEZ LE NOM DE LA DIODE A METTRE A JOUR :');
        POSIT(4,50);
        TTREADN(NOM,4,50);
        MAJUSCULE(NOM);
        FINDK(FDIO,0,NOM,EQL);
        IF STATUS(FDIO) <> 0 THEN
         BEGIN
          POSIT(6,5);
          TTWRITE('LA DIODE DE CE NOM N''EXISTE PAS');
         END
         ELSE
         BEGIN
          READ(FDIO,DIODE);
          CAR := 'N';
          WHILE ((CAR='N') OR (CAR='N')) DO
```

```
        NORMAL;
       END
      ELSE DELETE (FTRAN);
      END;
    END;
'3':
    CASE CARTRAIT OF
   '1':BEGIN
       EFFACETOUT;
       POSIT(3,10);
       REVERSE;
       SOULIGNER;
       TTWRITE('CREATION D"UN CIRCUIT COMPLEXE');
       POSIT(4,5);
       TTWRITE('LE FICHIER SOURCE DECRIVANT LE CIRCUIT SERA COMPILE' );
       POSIT(5,5);
       TTWRITE(' IL NE DOIT COMPRENDRE QUE LES ELEMENTS DE BASE ');
       POSIT(8,4);
       TTWRITE('DONNER LE NOMBRE DE NOEUDS EXTERNES : ');RESTAURE;
       READLN(NBEXT);
       POSIT(10,9);
       REVERSE;SOULIGNER;
       TTWRITE('INTRODUCTION DES NOEUDS EXTERNES');RESTAURE;
       FOR IM := 1 TO NBEXT DO
       BEGIN
        WRITELN('NOEUD',IM:2,' >');READLN(NOEUD[IM]);
        NBNOEUD := NBNOEUD + 1;
       END;
       DESCRIPTION;(* APPEL DE LOCE POUR COMPILER LE FICHIER SOURCE *)
       IF NOT SUISEM THEN (* COMPILATION CORRECTE*)
        BEGIN
         POSIT(6,4);
         TTWRITE('DONNER LE NOM DU CIRCUIT : ');
         POSIT(6,50);TTREADN(ENTREE,6,50);
         TYPER(ENTREE,SORTIE,'T','X','T');
         OPEN(COMPLEX,SORTIE,NEW);
         (* MISE A JOUR DES PARAMETRES LOCAUX AU CIRCUIT');
         WRITELN(COMPLEX,NBEXT);
         NBINT := NBNOEUD - NBEXT;
         WRITELN(COMPLEX,NBINT);
         WRITELN(COMPLEX,INAM);
         FOR IM:=1 TO NBNOEUD DO WRITELN(NOEUD[IM]);
         PBRANCH := FIRSTCIR;
         (* INTRODUCTION DES BRANCHES DANS LA BIBLIOTHEQUE *)
         WHILE(PTRBR<>NIL) DO
         BEGIN
          WRITE(COMPLEX,PTRBR^.NA,PTRBR^.ND,PTRBR^.VAL,PTRBR^.TYP);
          IF (PTRBR^.PTR <> NIL) THEN
           BEGIN
            PTRNOD := PTRBR^.PTR;
            WRITELN(COMPLEX,PTRNOD^.VAL);
           END;
          PTRBR := PTRBR^.BRSUI;
          WRITELN(COMPLEX);
         END;
```

```
    SOULIGNER;
    REVERSE;
    POSIT(4,5);
    TTWRITE('LES VALEURS DES PARAMETRES DU TRANSISTOR EN REGIME CONTINU');
    POSIT(6,5);
    TTWRITE('* BE :');TTREADN(VALUE,6,15); CONVERTIR(VAL,VALUE);
    POSIT(7,5);
    TRANSISTOR.PARAMETRE[11]:=VAL;
    TTWRITE('* BC :');TTREADN(VALUE,7,15); CONVERTIR(VAL,VALUE);
    POSIT(8,5);
    TRANSISTOR.PARAMETRE[12]:=VAL;
    TTWRITE('* RC :');TTREADN(VALUE,8,15); CONVERTIR(VAL,VALUE);
    POSIT(9,5);
    TRANSISTOR.PARAMETRE[13]:=VAL;
    TTWRITE('* R1 :');TTREADN(VALUE,9,15); CONVERTIR(VAL,VALUE);
    POSIT(10,5);
    TRANSISTOR.PARAMETRE[14]:=VAL;
    TTWRITE('* R2 :');TTREADN(VALUE,10,15); CONVERTIR(VAL,VALUE);
    POSIT(11,5);
    TRANSISTOR.PARAMETRE[15]:=VAL;
    TTWRITE('* I1 :');TTREADN(VALUE,11,5); CONVERTIR(VAL,VALUE);
    POSIT(12,5);
    TRANSISTOR.PARAMETRE[16]:=VAL;
    TTWRITE('* I2 :');TTREADN(VALUE,12,5); CONVERTIR(VAL,VALUE);
    POSIT(13,5);
    TRANSISTOR.PARAMETRE[17]:=VAL;
    TTWRITE('* IALI2 :');TTREADN(VALUE,13,5); CONVERTIR(VAL,VALUE);
    POSIT(14,5);
    TRANSISTOR.PARAMETRE[18]:=VAL;
    REVERSE;TTWRITE('* IALI1 :');POSIT(14,15);TTREADN(VALUE,14,15); CONVERTIR
    TRANSISTOR.PARAMETRE[19]:=VAL;
    POSIT(15,5);TTWRITE('* IZMI8 :');POSIT(15,15);
    TTREADN(VALUE,15,15); CONVERTIR(VAL,VALUE);
    TRANSISTOR.PARAMETRE[20]:=VAL;
    WRITE(FTRAN,TRANSISTOR);
  END;
'2':BEGIN
    ERASETOUT;
    POSIT(2,10);
    SOULIGNER;
    REVERSE;
    TTWRITE('MISE A JOUR D"UN TRANSISTOR');
    POSIT(4,5);
    NORMAL;
    TTWRITE('DONNEZ LE NOM DU TRANSISTOR A METTRE A JOUR');
    POSIT(4,50);
    TTREADN(NOM,4,50);
    FINDK(FTRAN,0,NOM,EQL);
    IF STATUS(FTRAN) <> 0 THEN
     BEGIN
      POSIT(5,20);
      CLIGNOTER;
      TTWRITE('LE TRANSISTOR N"EXISTE PAS');
     END
    ELSE
```

```
-SL-
  '2':
      CASE CARTRAIT OF
      '1':BEGIN
          EFFACETOUT;
          POSIT(2,10);
          REVERS;
          SOULIGNER;
          TTWRITE('CREATION D"UN TYPE DE TRANSISTOR.');
          POSIT(4,5);
          NORMAL;
          TTWRITE('LE NOM DU NOUVEAU TYPE DU TRANSISTOR CREE :');
          POSIT(4,60);
          TTREADN(NOM,4,60);
          TRANSISTOR.NOM:=NOM;
          EFFACETOUT;
          REVERS;
          SOULIGNER;
          POSIT(6,5);
          TTWRITE('LES VALEURS DES PARAMETRES DU TRANSISTOR EN BASSES FREQUEN
          POSIT(8,5);
          TTWRITE('* RH11 :');POSIT(8,15);TTREADN(VALUE,8,15); CONVERTIR(VAL,
          POSIT(9,5);
          TRANSISTOR.PARAMETRE[1] := VAL;
          TTWRITE('* VBETA :');TTREADN(VALUE,9,15); CONVERTIR(VAL,VALUE);
          POSIT(10,5);
          TRANSISTOR.PARAMETRE[2]:=VAL;
          REVERSE;TTWRITE('* RH22 :');TTREADN(VALUE,10,15); CONVERTIR(VAL,VAL
          TRANSISTOR.PARAMETRE[3]:=VAL;
          EFFACETOUT;
          REVERS;
          SOULIGNER;
          POSIT(4,5);
          TTWRITE('LES VALEURS DES PARAMETRES DU TRANSISTOR EN HAUTES FREQUEN
          POSIT(6,5);
          TTWRITE('* RBB :');TTREADN(VALUE,6,15); CONVERTIR(VAL,VALUE);
          POSIT(7,5);
          TRANSISTOR.PARAMETRE[4]:=VAL;
          TTWRITE('* RBE :');TTREADN(VALUE,7,15); CONVERTIR(VAL,VALUE);
          POSIT(8,5);
          TRANSISTOR.PARAMETRE[5]:=VAL;
          TTWRITE('* CB :');TTREADN(VALUE,8,15); CONVERTIR(VAL,VALUE);
          POSIT(9,5);
          TRANSISTOR.PARAMETRE[6]:=VAL;
          TTWRITE('* CCS :');TTREADN(VALUE,9,15); CONVERTIR(VAL,VALUE);
          POSIT(10,5);
          TRANSISTOR.PARAMETRE[7]:=VAL;
          TTWRITE('* RCS :');TTREADN(VALUE,10,15); CONVERTIR(VAL,VALUE);
          POSIT(11,5);
          TRANSISTOR.PARAMETRE[8]:=VAL;
          TTWRITE('* VB :');TTREADN(VALUE,11,15); CONVERTIR(VAL,VALUE);
          POSIT(12,5);
          TRANSISTOR.PARAMETRE[9]:=VAL;
          TTWRITE('* RCE :');POSIT(12,15);TTREADN(VALUE,12,15); CONVERTIR(VAL
          TRANSISTOR.PARAMETRE[10]:=VAL;
          EFFACETOUT;
```

```
    BEGIN
     READ(FTRAN,TRANSISTOR);
     CAR := 'N';
     WHILE ((CAR='N') OR (CAR='N')) DO
     BEGIN
      POSIT(6,5);
      EFFACELIGNE;
      POSIT(7,5);
      EFFACELIGNE;
      POSIT(10,5);
      EFFACELIGNE;
      POSIT(6,5);
      TTWRITE('DONNEZ LE NOM DU PARAMETRE A METTRE A JOUR');
      POSIT(6,50);
      TTREADN(PARAM,6,50);
      POSIT(7,5);
      K := ORDRE(NOM,15,VECTRAN);
      IF K<> 0 THEN
      BEGIN
       TTWRITE('DONNEZ SA NOUVELLE VALEUR :');
       POSIT(7,35);
       TTREADN(VALUE,7,35); CONVERTIR(VAL,VALUE);
       TRANSISTOR.PARAMETRE[K]:=VAL;
      END
      ELSE
      BEGIN
       CLIGNOTER;
       POSIT(12,6);
       TTWRITE('CE PARAMETRE N''EXISTE PAS');
       NORMAL;
      END;
      POSIT(10,5);
      TTWRITE('AVEZ-VOUS TERMINE LA MISE A JOUR DU TRANSISTOR ? (O/N)');
      TTREADN(CAR);
     END;
     WRITE(FTRAN,TRANSISTOR);
    END;
    END;
'3':BEGIN
    EFFACETOUT;
    POSIT(2,10);
    REVERSE;
    SOULIGNER;
    TTWRITE('SUPPRESSION D''UN TRANSISTOR');
    NORMAL;
    POSIT(4,5);
    TTWRITE('DONNEZ LE NOM DU TRANSISTOR A SUPPRIMER :');
    POSIT(4,46);
    TTREADN(NOM,4,46);
    FINDK(FTRAN,0,NOM,EQL);
    IF STATUS(FTRAN) <> 0 THEN
     BEGIN
      POSIT(10,46);
      CLIGNOTER;
      TTWRITE('CE TYPE DE TRANSISTORE N''EXISTE PAS');
```

```
7            BEGIN
7             POSIT(6,5);
7             EFFACELIGNE;
7             POSIT(7,5);
7             EFFACELIGNE;
7             POSIT(10,5);
7             EFFACELIGNE;
7             POSIT(6,5);
7             NORMAL;
7             TTWRITE('DONNEZ LE NOM DU PARAMETRE A METTRE A JOUR :');
7             POSIT(6,55);
7             TTREADN(NOM,6,55);
7             POSIT(7,5);
7             K := ORDRE(NOM,2,VECTDIO);
7             IF K<>0 THEN
              BEGIN
               TTWRITE('DONNEZ SA NOUVELLE VALEUR :');POSIT(7,35);
               TTREADN(VALUE,7,35);
               CONVERTIR(VAL,VALUE);
               DIODE.PARAMETRE[K]:=VAL;
              END
              ELSE
              BEGIN
               POSIT(10,5);
               CLIGNOTER;
               TTWRITE('CE PARAMETRE N''EXISTE PAS');
              END;
              POSIT(12,5);
              TTWRITE('AVEZ-VOUS TERMINE LA MISE A JOUR DE LA DIODE ?(O/N)');
              TTREADNE(CAR);
              END;
              WRITE(FDIO,DIODE);
            END;
        END;
    '3':BEGIN
        EFFACETOUT;
        POSIT(2,10);
        REVERSE;
        SOULIGNER;
        TTWRITE('SUPPRESSION D''UNE DIODE');
        POSIT(4,5);
        NORMAL;
        TTWRITE('DONNEZ LE NOM DE LA DIODE A SUPPRIMER :');
        POSIT(4,45);
        TTREADN(NOM,4,45);
        FINDK(FDIO,0,NOM,EQL);
        IF STATUS(FDIO) <>0 THEN
         BEGIN
          POSIT(10,45);
          CLIGNOTER;
          TTWRITE('CE TYPE DE DIODE N''EXISTE PAS');
         END
        ELSE  DELETE(FDIO);
        END;
    END;
```

```
        END;
      CLOSE(COMPLEX);
      END;
   '2':BEGIN
      EFFACETOUT;POSIT(2,10);REVERSE;SOULIGNER;
      TTWRITE('MISE A JOUR D"UN CIRCUIT COMPLEXE');
      POSIT(6,4);
      NORMAL;
      TTWRITE('UTILISER L"EDITEUR DE TEXTES POUR TOUTE MISE A JOUR');
      POSIT(7,4);
      TTWRITE('D"UN CIRCUIT COMPLEXE');
      END;
   '3':BEGIN
      EFFACETOUT;POSIT(2,10);REVERSE;SOULIGNER;
      TTWRITE('SUPPRESSION D"UN CIRCUIT COMPLEXE');NORMAL;
      POSIT(4,5);
      TTWRITE('DONNER LE NOM DU CIRCUIT : ');
      POSIT(4,55);TTREADN(NOM,4,55);
      TYPER(ENTREE,SORTIE,'T','X','T');
      OPEN(COMPLEX,SORTIE,OLD,DISPOSITION:=DELETE);
      IF (STATUS(COMPLEX) <> 0) THEN
       BEGIN
        POSIT(20,4);CLIGNOTER;
        TTWRITE('CIRCUIT INEXISTANT EN BIBLIOTHEQUE');
       END;
      END;
     END;
    END;
    NORMAL;
    POSIT(22,10);
    TTWRITE('AVEZ-VOUS TERMINE DE TRAVAILLER AVEC LA BIBLIOTHEQUE ? (O/N)');
    TTREADN(CAR);
  END;
END;
END.
```

```
-L-SL-

  0  0 [INHERIT('VIRGO.PEN','PROJET.PEN')]
  0  0 MODULE CACHE(INPUT,OUTPUT);
  0  0
  1  0 PUBLIC PROCEDURE STOR_OBJ(NOM : T1);
  1  0   (*-------------------------------------------------------------------
  1  0   (*          PROCEDURE DE STOKAGE DU MODULE OBJET PRODUIT
  1  0   (*               PAR LOCE DANS LA BIBLIOTHEQUE
  1  0   (*-------------------------------------------------------------------
  1  0   VAR   OBJET       : TEXT;
  1  0         NAME        : T1;
  1  0         I           : INTEGER;
  1  1   BEGIN
  1  1    TYPER(NOM,NAME,'L','O','C');
  1  1    OPEN(OBJET,NAME,NEW);
  1  1    REWRITE(OBJET);
  1  1
  1  1    (*--------   SAUVEGARDE DES SOUS CIRCUITS  ---------*)
  1  1
  1  1    WRITELN(OBJET,NSOUS);
  1  1    SCIRCOUR := FIRSTSCI;
  1  1    WHILE(SCIRCOUR<>NIL) DO
  1  2    BEGIN
  1  2     WRITELN(OBJET,SCIRCOUR^.NOM);
  1  2     WRITELN(OBJET,SCIRCOUR^.NBEXT);
  1  2     WRITELN(OBJET,SCIRCOUR^.NBINT);
  1  2     WRITELN(OBJET,SCIRCOUR^.NBRANCH);
  1  2     FOR I:= 1 TO (SCIRCOUR^.NBINT+SCIRCOUR^.NBEXT) DO
  1  2      WRITELN(OBJET,SCIRCOUR^.VECTNOD[I]);
  1  2     WRITELN(OBJET);
  1  2     FOR I:=1 TO SCIRCOUR^.NBRANCH DO
  1  2      WRITELN(OBJET,SCIRCOUR^.VECTNOM[I]);
  1  2     WRITELN(OBJET);
  1  2
  1  2     (*-----   SAUVEGARDE DES BRANCHES DE CHAQUE SOUS CIRCUIT   ----*)
  1  2
  1  2     BRCOUR := SCIRCOUR^.PTDESC;
  1  2     FOR I:=1 TO NBRANCH DO
  1  3     BEGIN
  1  3      WRITE(OBJET,BRCOUR^.NA,BRCOUR^.ND);
  1  3      WRITE(OBJET,BRCOUR^.TYP,BRCOUR^.VAL);
  1  3      WRITE(OBJET,BRCOUR^.NBIB);
  1  3      IF (BRCOUR^.PTR <> NIL) THEN
  1  4      BEGIN
  1  4       PTRNOD := BRCOUR^.PTR;
  1  4       WRITE(OBJET,PTRNOD^.VAL);
  1  3      END;
  1  3      WRITELN(OBJET);
  1  3      BRCOUR := BRCOUR^.BRSUI;
  1  2     END;
  1  2     SCIRCOUR := SCIRCOUR^.SCIRSUI;
  1  1    END;
  1  1
  1  1    (*----------  SAUVEGARDE DES BRANCHES DU CIRCUIT  ------------*)
  1  1    WRITELN(INAM);
  1  1    BRCOUR := FIRSTCIR;
```

```
-SL-
  1     FOR I:= 1 TO INAM DO WRITELN(OBJET,NAMES[I]);
  1     WHILE(ERCOUR<>NIL) DO
  2      BEGIN
  2       WRITE(OBJET,PRCOUR^.NA,ERCOUR^.ND);
  4       WRITE(OBJET,ERCOUR^.TYP,ERCOUR^.VAL);
  3       WRITE(OBJET,ERCOUR^.NBIS);
  1       IF (ERCOUR^.PTR <> NIL) THEN
  1       BEGIN
  2        PTRNOD := ERCOUR^.PTR;
  2        WRITE(OBJET,PTRNOD^.VAL);
  2       END;
  2       WRITELN(OBJET);
  1       ERCOUR := ERCOUR^.ERSUI;
  1      END;
  0   END;
  0
  0         (*----  FIN DE LA PROCEDURE STOR_OBJ  ----*)
  0  END.
```

```
      PROGRAM FILL
C     CE PROGRAMME RECHERCHE LES REMPLISSAGES DUS A LA 1ERE IT..
      COMMON  A(50,50),DIAG(50),LPACK(200),UPACK(200),UCOL(200),
     *,K(50)
      INTEGER HIGH,HI,UCOL,UROWST
      REAL LPACK
      READ(5,1) NNODE
      WRITE(6,5)NNODE
    1 FORMAT(I5)
      READ(5,2)((A(I,J),J=1,NNODE),I=1,NNODE)
      WRITE(7,3)((A(I,J),J=1,NNODE),I=1,NNODE)
    2 FORMAT(5F10.0)
    3 FORMAT(1X,5F12.4)
      LENGHT=0
      K=0
      DO 50 I=1,4
         DIAG(I)=A(I,I)
         WRITE(6,7)I,DIAG(I)
    7    FORMAT(1X,'DIAG(',I2,')=',F12.4)
         M=1
         DO 20 J=I+1,5
         IF(M.NE.1) GOTO 30
      IF (A(I,J).EQ.0.0) GOTO 6
      GOTO 9
    6 IF(A(I,1).NE.0.0.AND.A(1,J).NE.0.0) GOTO 9
      GOTO 20
    9 LENGHT=LENGHT+1
      UROWST(I)=LENGHT
      WRITE(6,4)I,UROWST(I)
    4 FORMAT(1X,'UROWST(',I2,')=',I5)
      WRITE(6,5)LENGHT
    5 FORMAT(1X,I2,///)
      LPACK(LENGHT)=A(I,J)
      UCOL(LENGHT)=J
      UPACK(LENGHT)=A(J,I)
      WRITE(6,8)LENGHT,LPACK(LENGHT),LENGHT,UCOL(LENGHT)
      WRITE(6,10)LENGHT,UPACK(LENGHT)
    8 FORMAT(1X,'LPACK(',I2,')=',F12.4,1X,'UCOL(',I2,')=',I5)
   10 FORMAT(1X,'UPACK(',I2,')=',F12.4)
      K=J
      M=0
   20 CONTINUE
   30 IF( K .GE.5)GOTO 50
      DO 30 J=K+1,5
      IF (A(I,J).EQ.0.0) GOTO 12
      GOTO 60
   12 IF(A(1,J).NE.0.0.AND.A(I,1).NE.0.0) GOTO 60
      GOTO 30
   60 LENGHT=LENGHT+1
      LPACK(LENGHT)=A(I,J)
      UPACK(LENGHT)=A(J,I)
      UCOL(LENGHT)=J
      WRITE(6,61) LENGHT,LPACK(LENGHT),LENGHT,UCOL(LENGHT)
      WRITE(6,62)L,UPACK(LENGHT)
   61 FORMAT(1X,'LPACK(',I2,')=',F12.4,1X,'UCOL(',I2,')=',I5)
   62 FORMAT(1X,'UPACK(',I2,')=',F12.4)
   30 CONTINUE
   50 CONTINUE
      DIAG(5)=A(5,5)
      WRITE(6,11)DIAG(5)
   11 FORMAT(1X,'DIAG(5)=',F12.4)
C *** ECRITURE DES RESULTATS OBTENUS ***
```

```
      IF (HIGH.LT.LOW) GOTO 4
      DO 3 MU=LOW,HIGH
      UPACK(MU)=UPACK(MU)/DIAG(N)
      CONTINUE

  SUBSTRACTION OF PRODUCTS

      DO 6 ML=LOW,HIGH
      I=UCOL(ML)
      DO 6 MU=LOW,HIGH
      J= UCOL(MU)
      IF(I-J) 5,3,5
      DIAG(J)=DIAG(J)-UPACK(MU)*LPACK(ML)
      GOTO 6

  SEARCH FOR LOCATION IN UPACK CORRESPONDING
  TO (I,J) ELEMENT IN U
      LO=UROWST(I)
      HI=UROWST(I+1)-1
      IF (HI.LT.LO) GOTO 6
      DO 4 LOC=LO,HI
      IF (UCOL(LOC).EQ.J) GOTO 5
      CONTINUE
      UPACK(LOC)=UPACK(LOC)-UPACK(MU)*LPACK(ML)
      LPACK(LOC)=LPACK(LOC)-LPACK(MU)*UPACK(ML)
      DO 3 MU=LOW,HIGH
      CONTINUE
      RETURN
      END


      SUBROUTINE  SOLV (NNODE,LENGHT)
      COMMON DIAG(50),UPACK(200),LPACK(200),UCOL(200),UROWST(50),
     +      X(50)
      INTEGER HIGH,UCOL,UROWST
      REAL LPACK
      SOLV THE LOWER TRIANGULAR SYSTEM

      NM1=NNODE-1
      UROWST(NNODE)=LENGHT+1
      DO 1 J=1,NM1
      X(J)=X(J)/DIAG(J)
      LOW=UROWST(J)
      HIGH=UROWST(J+1)-1
      IF (HIGH.LT.LOW) GOTO 1
      DO 5 K=LOW,HIGH
      MCOL=UCOL(K)
      X(MCOL)=X(MCOL)-LPACK(K)*X(J)
      CONTINUE
      CONTINUE
      X(NNODE)=X(NNODE)/DIAG(NNODE)
      WRITE(6,13)
      WRITE(6,14)X
      FORMAT(/'RESULT OF LY=U'/)
      FORMAT(1X,5F12.4)
 SOLVE THE UPPER TRIANGULAR SYSTEM

      DO 2 J=2,NNODE
      K=NNODE-J+1
      LOW=UROWST(K)
      HIGH=UROWST(K+1)-1
      IF (HIGH.LT.LOW) GOTO 2
      DO 4 M=LOW,HIGH
      MCOL=UCOL(M)
      X(K)=X(K)-UPACK(M)*X(MCOL)
      CONTINUE
      CONTINUE
```

```
PROGRAM SOLVES THE EQUATIONS ...
----------------------------------------------------------------
READ(7,13)LENGHT,NNODE
READ(7,14)(DIAG(J),J=1,NNODE)
READ(7,13)(UROWST(J),J=1,NNODE)
READ(7,14)(LPACK(J),J=1,LENGHT)
READ(7,13)(UCOL(J),J=1,LENGHT)
READ(7,14)(UPACK(J),J=1,LENGHT)
WRITE(8,15)LENGHT,NNODE
WRITE(8,16)(DIAG(J),J=1,NNODE)
WRITE(8,16)(UPACK(J),J=1,LENGHT)
WRITE(8,16)(LPACK(J),J=1,LENGHT)
WRITE(8,19)(UCOL(J),J=1,LENGHT)
NN1=NNODE-1
WRITE(8,19)(UROWST(J),J=1,NNODE)


CALL LUFACT (NNODE,LENGHT)
WRITE(8,17)
WRITE(8,16)(DIAG(J),J=1,NNODE)
WRITE(8,16)(UPACK(J),J=1,LENGHT)
WRITE(8,16)(LPACK(J),J=1,LENGHT)
WRITE(8,19)(UCOL(J),J=1,LENGHT)
WRITE(8,19)(UROWST(J),J=1,NN1)
READ(7,14)(X(J),J=1,NNODE)
WRITE(8,21)
WRITE(8,16)(X(J),J=1,NNODE)


CALL SOLV (NNODE,LENGHT)

WRITE(8,22)
WRITE(8,16)(X(J),J=1,NNODE)

 FORMAT (2I5)
 FORMAT(8F10.0)
 FORMAT(/'INPUT DATA'/8I10/)
 FORMAT(1X,8F12.4)
 FORMAT( // 'RESULT OF LU FACT' /
       'ELEMENTS OF DIAG,UPACK,UCOL,LPACK AND UROWST' /)
 FORMAT(16I8)
FORMAT(/'CONSTANT VECTOR'/)
FORMAT(/'SOLUTION VECTOR'/)
END
SUBROUTINE LUFACT (NNODE,LENGHT)
COMMON DIAG(50),UPACK(200),LPACK(200),UCOL(200),UROWST(50)
INTEGER HIGH,HI,UCOL,UROWST
REAL LPACK
NN1=NNODE-1
UROWST(NNODE)=LENGHT+1
DO 8 N=1,NN1

...N ONE ROW OF THE U MATRIX

LOW=UROWST(N)
HIGH=UROWST(N+1)-1
IF (HIGH.LT.LOW) GOTO 8
DO 1 NU=LOW,HIGH
UPACK(NU)=UPACK(NU)/DIAG(N)
CONTINUE

SUBTRACTION OF PRODUCTS

DO 8 NL=LOW,HIGH
...
```

```
MAT(5F10.3)
MAT(1X,5F10.2)
GHT=0

DO I=1,N
DIAG(I)=A(I,I)
WRITE(6,7)I,DIAG(I)
FORMAT(1X,'DIAG(',I2,')=',F10.2)
M=1
DO 20 J=I+1,N
IF(M.NE.1) GOTO 30
(A(I,J).EQ.0.0) GOTO 6
O 5

A(I,1).NE.0.0.AND.A(1,J).NE.0.0) GOTO 5
O 20
GHT=LENGHT+1
WST(I)=LENGHT
TE(6,4)I,UROWST(I)
MAT(1X,'UROWST(',I2,')=',I5)
TE(6,5)LENGHT
MAT(1X,I2)
CK(LENGHT)=A(I,J)
L(LENGHT)=J
CK(LENGHT)=A(J,I)
TE(6,8)LENGHT,LPACK(LENGHT),LENGHT,UCOL(LENGHT)
TE(6,10)LENGHT,UPACK(LENGHT)
MAT(1X,'LPACK(',I2,')=',F10.2,1X,'UCOL(',I2,')=',I5)
MAT(1X,'UPACK(',I2,')=',F10.2)


TINUE
(K .GE. N)GOTO 60
DO J=K+1,N
(A(I,J).EQ.0.0) GOTO 12
O 60
A(1,J).NE.0.0.AND.A(I,1).NE.0.0) GOTO 60
O 60
GHT=LENGHT+1
CK(LENGHT)=A(I,J)
CK(LENGHT)=A(J,I)
L(LENGHT)=J
TE(6,61) LENGHT,LPACK(LENGHT),LENGHT,UCOL(LENGHT)
TE(6,62)L,UPACK(LENGHT)
MAT(1X,'LPACK(',I2,')=',F10.2,1X,'UCOL(',I2,')=',I5)
MAT(1X,'UPACK(',I2,')=',F10.2)
TINUE
TINUE
G(5)=A(5,5)
TE(6,11)DIAG(5)
MAT(1X,'DIAG(5)=',F10.2)

ITURE DOS RESULTAS OBTIDUS***
-------------------------------
TE(7,*)NNODE,LENGHT
TE(7,40)(DIAG(N),N=1,NNODE)
TE(7,40)(UPACK(N),N=1,LENGHT)
TE(7,40)(LPACK(N),N=1,LENGHT)
TE(7,41)(UCOL(N),N=1,LENGHT)
TE(7,41)(UROWST(N),N=1,NNODE-1)
MAT(1X,5F10.2)
MAT(1X,16I5)
E(7)
N(7,FILE='COMP.DAT',STATUS='OLD')
```

```
 UCOL(MU)
(I-J) 3,3,5
AG(J)=DIAG(J)-UPACK(MU)*LPACK(ML)
T. 8

CH FOR LOCATION IN UPACK CORRESPONDING
I,J) ELEMENT IN C
=UROWST(I)
=UROWST(I+1)-1
 (HI.LT.LO) GOTO 6
 LOC=LO,HI
 (UCOL(LOC).EQ.J) GOTO 5
NTINUE
ACK(LOC)=UPACK(LOC)-UPACK(MU)*LPACK(ML)
ACK(LOC)=LPACK(LOC)-LPACK(MU)*UPACK(ML)

NTINUE
TURN
D


BROUTINE  SOLV (NNODE,LENGHT)
MMON  DIAG(50),UPACK(200),LPACK(200),UCOL(200),UROWST(50),
  X(50)
TEGER HIGH,UCOL,UROWST
AL LPACK
LV THE LOWER TRIANGULAR SYSTEM

1=NNODE-1
ROWST(NNODE)=LENGHT+1
 1 J=1,NM1
J)=X(J)/DIAG(J)
W=UROWST(J)
GH=UROWST(J+1)-1
 (HIGH.LT.LOW) GOTO 1
 2 K=LOW,HIGH
OL=UCOL(K)
COL)=X(ICOL)-LPACK(K)*X(J)
NTINUE
NTINUE
NNODE)=X(NNODE)/DIAG(NNODE)
ITE(6,13)
ITE(6,14)X
RMAT(/'RESULT OF LY=U'/)
RMAT(1X,5F12.4)
THE UPPER TRIANGULAR SYSTEM

 2 J=2,NNODE
NNODE-J+1
W=UROWST(K)
GH=UROWST(K+1)-1
 (HIGH.LT.LOW) GOTO 2
 4 M=LOW,HIGH
OL=UCOL(M)
K)=X(K)-UPACK(M)*X(ICOL)
NTINUE
NTINUE
TURN
```

```
FOR I:=N DOWNTO 1 DO
  BEGIN
  S:=0;
  pivmax:=a[lig[i],i];
  FOR J:=I+1 TO N DO
   BEGIN
    S:=S+(A[Lig[i],J]*X[J]);
    END;
    X[I]:=(B[lig[i]]-S)/pivmax;
  END;
END;{GAUSS2}
(*************************************************************************
(*
(*          CORPS DU PROGRAMME PRINCIPAL DU SIMULATEUR EN REGIME CONTINU
(*          ==================================================================
(*
(*************************************************************************

Begin
  (* Lecture de la matrice de description *)
  For i:=1 to 400 do  B[i] := 0;
  i := 1; (*Indice des branches *)
  j := 1; (*Indice des elements reactifs *)
(* Parcours de tout le circuit branche a branche *)
(*----------------------------------------------------------*)
While ( i <= Branche ) do
 Begin
  (* Remplissage de la sous arice A *)
  (*-------------------------------*)
  If Matdesc[i].Nd <> 0 then
  Begin
  a[Nbnoeud+i,Matdesc[i].Nd+Branche]:=1;
  a[Matdesc[i].Nd, i]:=1;
  End;

  (* Remplissage de la sous arice At *)
  (*---------------------------------*)
  If Matdesc[i].na <> 0 then
  Begin
  a[Nbnoeud+i,Matdesc[i].Na+Branche]:=-1;
  a[Matdesc[i].Na,i] := -1;
  End;

  (* Remplissage de la sous arice -I *)
  (*---------------------------------*)
  a[Nbnoeud+i,Nbnoeud+Branche+i] := -1;
  (* Loi d'Ohm appliquee au circuit *)
  (*-------------------------------*)
  Case Matdesc[i].Compose_of
   (* Voir de quel composant il s'agit *)
   1 :(* Cas d'une resistance *)
   (*--------------------------*)
   Begin
   a[Nbnoeud+Branche+i,i] := Matdesc[i].Val;
   a[Nbnoeud+Branche+i,Nbnoeud+branche+i] := -1;
```



```
   End;
```

```
                    begin
                    m:=a[i,k]/pivmax;
                    for j:=(k+1) to n do
                    begin
                    a[i,j]:=a[i,j]-(m*a[indic,j]);
                    end;
                    b[i]:=b[i]-(m*b[indic]);
                    end;
                    end;
                    end;
                    i:=1;
                    arret:=false;
                    while ((i<= n) and (not arret)) do
                     begin
                     parc:=false;
                     j:=1;
                     while ((j<= n) and (not parc)) do
                      begin
                      if i=lig[j] then
                       parc:=true
                           else
                       j:=j+1;
                      end;
                      if not parc then
                       begin
                        arret:=true;
                        lig[n]:=i;
                       end
                      else
                       i:=i+1;
                     end;
                     FOR I:=N DOWNTO 1 DO
                     BEGIN
                      S:=0;
                      pivmax:=a[lig[i],i];
                      FOR J:=I+1 TO N DO
                      BEGIN
                       S:=S+(A[Lig[i],J]*X[J]);
                      END;
                      X[I]:=(B[Lig[i]]-S)/pivmax;
                     END;
                     END;
(*-------------------------------------*)
(* Corps De La Procedure Transit  *)
(*-------------------------------------*)
Begin
For i:=1 to (no+(2*br)+q) do
Begin
 For j:=1 to (no+(2*br)+q) do
  Begin
  Awl[i,j]:=Al[i,j];
  END;
 Bwl[i]:=Bl[i];
END;
n:=1;
```



```
b[n]:=b[n-1]*h;
```

```
  a[Nbnoeud+Branche+i,Nbnoeud+Branche+j] := Rabase[i].val;
  a[Nbnoeud+2*Branche+j,i] := 1;
  j := j +1;
  End;
Procedure Transit(Br,No,Q:integer;Var Lim:integer);
(*$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$*)
VAR I,inter,nbr,J,K,N,lig,iter:INTEGER;
    s,p,S1,S2,ak,e,ek,eta:REAL;
    Aw:array[1..400,1..400] of real;
    bk:array[0..60] of real;
    Bw:ARRAY[1..400] OF REAL;
        procedure gauss(var n:integer);
                VAR cour,l,indic,J,K,R:INTEGER;
                     parc,arret:boolean;
                     PIVmax,S,M:REAL;
                     lig:array[1..400] of integer;
                PROCEDURE PIVOT(var ind:integer;var piv:real);
                var cour,r:integer;
                     trait:boolean;
                     max,save:real;
                     sauv:array[1..400] of real;
                begin
                 max:=0;
                 for r:=1 to n do
                 begin
                 trait:=false;
                 for cour:=1 to (k-1) do
                 begin
                 if lig[cour]=r
                 then begin
                  trait:=true;
                      end;
                 end;
                 if not trait then begin
                 if abs(a[r,k])>max
                 then begin
                  max:=abs(a[r,k]);
                  piv:=a[r,k];
                  ind:=r;
                  end;
                end;
               end;
              end;
                BEGIN
                for k:=1 to (n-1) do
                begin
                pivot(indic,pivmax);
                lig[k]:=indic;
                for i:=1 to n do
                begin
                parc := false;
                for cour:=1 to k do
                begin
                if lig[cour] =i then
                 parc:=true;
```



```
                end;
```

```
    a[Nbnoeud+i,Nbnoeud+Branche+i] := -1;
    (* Loi d'Ohm appliquee au circuit *)
    (*-----------------------------------*)
    Case Matdesc[i].Compose of
     (* Voir de quel composant il s'agit *)
     1 :(* Cas d'une resistance *)
     (*-------------------------------*)
     Begin
     a[Nbnoeud+Branche+i,i] := Matdesc[i].Val;
     a[Nbnoeud+Branche+i,Nbnoeud+branche+i] := -1;
     End;

     2 : (* Cas d'une inductance *)
     (*-------------------------------*)
          Bobine;
     3 :(* Cas d'une capacite *)
     (*-------------------------------*)
          Capacite;

     4 :(* Cas d'une source de courant *)
     (*-------------------------------------*)
     begin
        if Matdesc[i].Coef = 0
         then
          Begin
           B[Nbnoeud+Branche+i]:=Matdesc[i].val;
           a[Nbnoeud+Branche+i,i] := 1;
          End
         else
          Begin
           a[Nbnoeud+Branche+i,i] := 1;
           Co := TRUNC(Matdesc[i].Coef);
           a[Nbnoeud+Branche+i,Co] := Matdesc[i].Val;
          End;
    End;
     5 : (* Cas d'une source de tension *)
     (*-------------------------------------*)
     Begin
        if Matdesc[i].Coef = 0
         then
          Begin
           B[Nbnoeud+Branche+i] := Matdesc[i].Val;
           a[Nbnoeud+Branche+i,Nbnoeud+Branche+i] := 1;
          End
         else
          Begin
           a[Nbnoeud+Branche+i,Nbnoeud+Branche+i] := -1;
           Co := TRUNC(Matdesc[i].Coef);
           a[Nbnoeud+Branche+i,Nbnoeud+Branche+Co] := Matdesc[i].Val;
          End;
      End;

    End;
  (* Fin du choix du composant *)
   i := i + 1;
```

```
                Source Listing                  15-Nov-1986 09:04:06    VAX-
                                                13-Oct-1986 [illegible]

end;
```

```
Begin
  for nbr:=1 to q do
    begin
    lig:=nbr+(2*br)+no;
    a[lig,lig]:=-1/h;
    b[lig]:=-xt[n-1,lig]/h;
    end;
     inter:=no+q+(2*br);
     gauss(inter);
     for i:=1 to ((2*br)+no+q) do
      begin
      xt[n,i]:=x[i];
      end;
     n:=n+1;
     t[n]:=t[n-1]+h;
     for i:=1 to ((2*br)+no+q) do
     begin
     for j:=1 to ((2*br)+no+q) do
     begin
     a[i,j]:=aw[i,j];
     end;
     B[i]:=Bw[i];
     end;
    end;
  Lim := n-1;
  end;
(*****************************************************************************)
(*                                                                           *)
(*            Programme Principal Du Regime Transitoire                      *)
(*                                                                           *)
(*****************************************************************************)
Begin
  For i:=0 to 400 do  B[i] := 0;
  i := 1; (*Indice des branches *)
  j := 1; (*Indice des elements reactifs *)
(* Parcours de tout le circuit branche a branche *)
(*-------------------------------------------------*)
While ( i <= Branche ) do
 Begin
  (* Remplissage de la sous matrice A *)
  (*-----------------------------------*)
  If Matdesc[i].Nd <> 0 then
  Begin
  a[Nbnoeud+i,Matdesc[i].Nd+Branche]:=1;
  a[Matdesc[i].Nd, i]:=1;
  End;

  (* Remplissage de la sous matrice At *)
  (*-----------------------------------*)
  If Matdesc[i].Na <> 0 then
  Begin
  a[Nbnoeud+i,Matdesc[i].Na+Branche]:=-1;
  a[Matdesc[i].Na,i] := -1;
  End;
```



```
  (* Remplissage de la sous matrice -1 *)
```

```
(* Programme qui remplit la matrice des equations du circuit
   a partir de la matrice decomposee du circuit *)
Type
  vec2 =Array[1..400] of real;
  vec1 = array[1..400,1..400] of real;
Var
   Reset,i,k1,j,l,co,inter,time,br,limit :  integer;
   inter1:real;
   a : Array[1..400,1..400] of real;
   B :array[1..400] of real;
   car:char;
Procedure Bobine;
(*$$$$$$$$$$$$$$$$$$*)
  Begin
    If Matdesc[i].Coef = 0  (* Inductance simple *)
     then
      Begin
      (* Mise a zero de l'intensite *)
      a[Nbnoeud+Branche+i,i] := Matdesc[i].Val;
      a[Nbnoeud+Branche+i,Nbnoeud+2*Branche+j] := -1;
      a[Nbnoeud+2*Branche+j,Nbnoeud+branche+i]:=1;
      End
     else
      Begin
      (*  Cas d'une inductance mutuelle *)
      a[Nbnoeud+Branche+i,i] := Matdesc[i].Val;
      a[Nbnoeud+Branche+i,i+1] := Matdesc[i].Coef;
      a[Nbnoeud+Branche+i,Nbnoeud+2*Branche+j] := -1;
      a[Nbnoeud+2*Branche+j,Nbnoeud+Branche+i] := 1;
      i:= i + 1;
      (* Mise a jour de la branche suivante *)
      j := j + 1;
      If Matdesc[i].Nd <> 0 then
      Begin
       a[Nbnoeud+i,Branche+Matdesc[i].Nd] := 1;
      a[Matdesc[i].Nd,i] := 1;
      End;

      If Matdesc[i].Na <> 0 then
      Begin
       a[Nbnoeud+i,Branche+Matdesc[i].Na] := -1;
      a[Matdesc[i].Na,i] :=  -1;
      End;
      a[Nbnoeud+i,Nbnoeud+Branche+i] := -1;
      a[Nbnoeud+Branche+i,i] := Matdesc[i].val;
      a[Nbnoeud+Branche+i,i-1] := Matdesc[i].Coef;
      a[Nbnoeud+Branche+i,Nbnoeud+2*Branche+j] := -1;
      End;
  a[Nbnoeud+2*Branche+j,Nbnoeud+Branche+i] := 1;
  j := j +1;
  End;
  Procedure Capacite;
 (*$$$$$$$$$$$$$$$$$$*)
  Begin
```

15-Nov-1986 09:0?:05
13-Oct-1986 ??:??:??

Source Listing

```
  (* Mise a zero de la tension *)
```

```
  2 : (* Cas d'une inductance *)
  (*------------------------*)
       Bobine;
  3 :(* Cas d'une capacite *)
  (*-----------------------*)
      Capacite;

  4 :(* Cas d'une source de courant *)
  (*--------------------------------*)
  Begin
    If Matdesc[i].Coef = 0
     then
      Begin
       R[Nbnoeud+Branche+i]:=Matdesc[i].val;
       a[Nbnoeud+Branche+i,i] := 1;
      End
     else
      Begin
       a[Nbnoeud+Branche+i,i] := 1;
       Co := TRUNC(Matdesc[i].Coef);
       a[Nbnoeud+Branche+i,Co] := Matdesc[i].Val;
      End;
 End;
  5 : (* Cas d'une source de tension *)
  (*-------------------------------*)
  Begin
     If Matdesc[i].Coef = 0
      then
       Begin
        R[Nbnoeud+Branche+i] := Matdesc[i].Val;
        a[Nbnoeud+Branche+i,Nbnoeud+Branche+i] := 1;
       End
      else
       Begin
        a[Nbnoeud+Branche+i,Nbnoeud+Branche+i] := -1;
        Co := TRUNC(Matdesc[i].Coef);
        a[Nbnoeud+Branche+i,Nbnoeud+Branche+Co] := Matdesc[i].val;
       End;
   End;

  End; (* Fin du choix du composant *)
 i := i + 1;
end;
 k1 := 2*Branche + Nbnoeud ;
 j := j -1;
 (* remise a zero des charges des elements reactifs dans le cas du continu
 For i := k1+1 to k1+j do
 Begin
   For l := k1+1 to k1+j do
    Begin
     a[i,l] := 0;
    End;
  End;
  i := Nbnoeud+2*Branche+j;
  GAUSS2(i);
```



```
 End;(CONTINU)
```

Bibliographie

1] Y.J PARK,T.W TANG "Monte-Carlo study of junction transistors"IEEE on electron devices vol ED 31,pp 1724-1729;1984

2] A.MALMBERG,F.L CORNELL"NET 1 Network analysis program " rapport LA 3119

3] M.JACOLIN "Les programmes IMAGI et IMAGII " l'onde electrique vol 49,fasc1 pp20-27;Janvier 1969

4] J.D PLUMMER,S.E HANSEN "VLSI Process Modeling-SUPREMIII"IEEE Trans Electron Devices ED 11 , pp1438-1453;1983

5] G.D HATCHEL,R.K BRAYTON,F.G GUSTAVSON "The sparse tableau approach to network analysis and design"IEEE Trans on circuit theory vol CT-18,n°1pp101-113;Janvier 1971

6] E.M BUTURLA, P.E COTRELL "Two-dimensionnal static and transient simulation of mobile carrier transport in a semi-conductor" Solid state Electron n°23 ,pp 331-334;1980

7] G.W BROWN,B.W LINDSAY "The numerical solution of Poisson's equations for two- dimensionnal semi-conductors devices" Solid state Electron n°19,pp 991-992 1976

8] B.T BROWN,J.J MILLER" Numerical analysis of semi-conductors devices" Proceeding of the NASECODE I Conférence,boole.Press,Dublin Ireland,1979

9] G.D HATCHEL,M.MACK,R.O'BRIEN "Semi-conductors device analysis via finite element" presnted at conference on circuits and systems,pacific Grove CA ;1974

10] B.SPEELPENNING "Semi-conductor analysis using finite élément" IBM J Res Develop. n°25 pp232-245 ; 1981

11] E.M BUTURLA, P.E COTTRELL "Two-dimensionnal finite élément analysis of semi-conductor phénomèna"Présented at the international conférence on numérical méthods in electric and magnétic field problems;Italy 1976

12] T.ADACHI, A.YASHII,T.SUDO "Two-dimensionnal semi-conductor analysis using finite élément méthod "IEEE Trans Electron Devices ED 26 pp1026-1031;1979

13] B.T MURPHY "Diode and transistor Self-analogues for circuit Analysis" B.S.T.J.,47,n°4 pp 487-502; avril 1968

14] E.M BUTURLA, P.E COTTRELL " Simulation of semiconductor transport using coupled and decoupled solution techniques" IEEE Journal of solid state circuits,vol 23 pp 331-334, 1980.

15] J.W SLOTBOOM"Computer -aided two-dimensionnal analysis of bipolar transistor"IEEE Trans Electron Devices ED 20 , pp669-679;1973

16] E.M BUTURLA ,P.E COTTRELL "FIELDAY -Three dimensionnal simulation of semi-conductor devices" IBM J Res Dev vol 25 n°4 ;Juillet 1981

17] L.J GIACOLETTO -RCA Review pp 506-562 ;Decembre 1964

18] J.J EBERS,J.L MOLL "Large signal behaviour of junction transistors"Proc IRE 42 n°12 pp1761-1772;Décembre 1954

19] R.BEAUFOY ,J.J SPARKES "The junction transistor as a charge controlled device" ATE J vol 13 n°4 pp310-324;octobre 1957

20] H.K GUMMEL "A charge contrôl relation for bipolar transistors"BSTJ vol 49 n°1 pp115-120 ;Janvier 1970

21]T.K OHTSUKI,T.K KANI "A unified modeling schéme for semi-conductor device with application for state variable analysis"IEEE Trans on Circuit Theory,vol 17 n°1 pp26-32;février 1970

22] J.G LINVILL "Lumped models of diodes and transistors" Proc IRE 46 n°6 pp1141-1152;juin 1958

23] J.M EARLY "Effect of space charge layer widenning in junction transistor" Proc IRE vol 40 n°11 pp1401-1406;Novembre1952

24] C.T SAH, R.N NOYCE ,W.SHOCKLEY "Carrier,génération and recombination in PN junction caractéristics" Proc IRE vol 45 n°9 pp1228-1243;Septembre 1957

25] Jr C.T KIRK "A theory of transistor cutoff frequency (ft) falloff at high current densities"IEEE Trans Electron Devices ED 9 pp 164-174;1962

26] R.CARTON, R.MICOLET "Modèles de transistors pour analyse de circuits par ordinateur" l'onde électrique vol 49 fasc 3 pp 295-308;Mars 1969

27] A.MALMBERG,F.L CORNELL"NET 1 Network analysis program " rapport LA 3119

28] SPICE : Version 2G6 User's guide Université of california, Berkeley ; 1981

29] SHOCKLEY "Electrons and holes in semionductors" D.Van Nostrand, Princeton N.J 1950.

30] J.BOUCHER,J.S SIMONNE"Principes et fonctions de l'electronique intégrée" Cepadues-editions pp 175-178;1977

31] P.ANTOGNETTI, G.MASSOBRIO "Semiconductor device modeling with SPICE" Mac Graw Hill; 1987

32] A.HAAS "Mesures electroniques-méthodes pratiques de mesure des montages electroniques et de tous leurs éléments constitutifs" ED.Radio ;1976

33] G.REY,K.LEMAIRE "Identification et caractérisation du comportement d'un transistor bipolaire en régime statique" l'onde electrique vol 50, fasc 6 pp503-516;1970

34] D.A CALAHAN "Computer aided network design" Mac Graw Hill;1972

35] F.H BRANIN,G.R HOGSET "ECAP2-A new electronic circuit analysis program"IEEE Journal of solid state circuits,vol sc-6 n°4,Aout 1971

36| L.O CHUA, P.M LIN "Computer aided analysis of electronic circuits- algorithms and computational techniques" Prentice hall inc,1975

37| P.HENRICI "Discrète variable methods in ordinary différential équation"John wiley N.Y 1962

38| KAHANER,MOLER,NASH "Numérical méthods and software"Prentice Hall ,1988

39| M.SIBONY,J.MARDIN "Systèmes linéaires et non-linéaires-Analyse numérique 1",Hermann 1982

40| R.K BRAYTON "A new efficient algorithm for solving différential algébraic system using implicit backward differentiation formulas".Proc of IEEE, vol 60 n°1 pp98-108;1975

41| C.W GEAR "Simultaneous numérical solution of differential algebraic equations"IEEE Trans on CT vol 18 n°1 pp89-95;Juin 1971

42| M.BOUMAHRAT,A.GOURDIN "Méthodes numériques appliquées" Publication OPU, 1978

43| N.CHAOURAR,H.FOUCHAL "SICEL.Logiciel d'aide à la mise au point de circuits électroniques" thèse d'ingéniorat CEN 1986

44| I.STEPANENKO"Principes de la micro-électronique " Edition Mir-Moscou;1980

45|A.HODGES ,ST JACKSON G.HORACE "Analysis and design of digital IC's" Departement of electrical engineering and computer sciences Berkeley ;1986