IMAG 3 : un système de simulation et d'optimisation de circuits electroniques

Jean-Claude Reynaud

THESE

présentée à

Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR DE 3ème CYCLE

Génie Informatique

par

Jean-Claude REYNAUD

IMAG3 : UN SYSTEME DE SIMULATION ET D'OPTIMISATION DE CIRCUITS ELECTRONIQUES

Thèse soutenue le 10 mai 1978 devant la Commission d'Examen :

Président : P.J. LAURENT

Examinateurs : J. BOREL
J. KUNTZMANN
C. LE FAOU
J. MERMET
D. VANDORPE

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Monsieur Philippe TRAYNARD : Président

Monsieur Pierre-Jean LAURENT : Vice Président

PROFESSEURS TITULAIRES

MM.	BENOIT Jean	Radioélectricité
	BESSON Jean	Electrochimie
	BLOCH Daniel	Physique du solide
	BONNETAIN Lucien	Chimie minérale
	BONNIER Etienne	Electrochimie et électrométallurgie
	BOUDOURIS Georges	Radioélectricité
	BRISSONNEAU Pierre	Physique du solide
	BUYLE-BODIN Maurice	Electronique
	COUMES André	Radioélectricité
	DURAND Francis	Métallurgie
	FELICI Noël	Electrostatique
	FOULARD Claude	Automatique
	LESPINARD Georges	Mécanique
	MOREAU René	Mécanique
	PARIAUD Jean-Charles	Chimie-Physique
	PAUTHENET René	Physique du solide
	PERRET René	Servomécanismes
	POLOUJADOFF Michel	Electrotechnique
	SILBER Robert	Mécanique des fluides

PROFESSEUR ASSOCIE

M.	ROUXEL Roland	Automatique

PROFESSEURS SANS CHAIRE

MM.	BLIMAN Samuel	Electronique
	BOUVARD Maurice	Génie mécanique
	COHEN Joseph	Electrotechnique
	LACOUME Jean-Louis	Géophysique
	LANCIA Roland	Electronique
	ROBERT François	Analyse Numérique
	VEILLON Gérard	Informatique fondamentale et appliquée
	ZADWORNY François	Electronique

MAITRES DE CONFERENCES

MM.	ANCEAU François	Mathématiques appliquées
	CHARTIER Germain	Electronique
	GUYOT Pierre	Chimie minérale
	IVANES Marcel	Electrotechnique
	JOUBERT Jean-Claude	Physique du solide
	MORET Roger	Electrotechnique nucléaire
	PIERRARD Jean-Marie	Mécanique
	SABONNADIERE Jean-Claude	Informatique fondamentale et appliquée
Mme.	SAUCIER Gabrièle	Informatique fondamentale et appliquée

MAITRE DE CONFERENCES ASSOCIE

M.	LANDAU Ioan	Automatique

CHERCHEURS DU C.N.R.S. (Directeur et Maîtres de Recherche)

MM.	FRUCHART Robert	Directeur de Recherche
	ANSARA Ibrahim	Maître de Recherche
	CARRE René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	MATHIEU Jean-Claude	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

Je tiens à remercier le Professeur P.J. LAURENT, professeur à l'U.S.M.G. et Directeur de l'E.N.S.I.M.A.G., qui a bien voulu me faire l'honneur de présider le jury de cette thèse. Ses remarques m'ont permis d'entrevoir la rigueur du Mathématicien.

Il est certain que ce travail doit beaucoup au Professeur J. KUNTZMANN. D'abord parce qu'il a pris l'initiative, dès 1965, d'intéresser une équipe de l'I.M.A.G. aux problèmes posés par l'analyse de circuits électroniques sur ordinateur. Ensuite parce qu'il a su m'encourager, tout au long de la rédaction, par ses conseils, son exemple et sa rigueur. Qu'il croie à ma reconnaissance la plus vive et à l'honneur qu'il me fait de participer au jury.

Je remercie Monsieur J. MERMET, Maître de recherches au C.N.R.S., qui a bien voulu m'accueillir dans son équipe. Par son intérêt constant pour la C.A.O., il a toujours su y encourager les recherches et les développements.

Monsieur BOREL, responsable du Laboratoire de micro-électronique appliquée du L.E.T.I. et ses collaborateurs, n'ont pas été rebutés par l'utilisation des premières versions du programme. Sans leur collaboration amicale et efficace, la mise au point du système IMAG aurait sûrement été difficile. Je remercie Monsieur BOREL d'avoir accepté de participer au jury.

L'intérêt de Monsieur D. VANDORPE, Maître de conférences à l'Université de LYON 1, pour la C.A.O. en électronique ne s'est jamais démenti. Nos conversations, souvent passionnées, toujours passionnantes, sont chaque fois source d'inspiration. Je le remercie d'avoir bien voulu juger ce travail.

Sans Monsieur C. LE FAOU, Ingénieur de recherches au C.N.R.S., ce travail n'aurait jamais été mené à bien. D'abord parce qu'il y a participé activement. Ensuite parce qu'il m'a introduit au domaine à la fois si particulier et si vaste de la simulation de circuits électroniques. Sa compétence et son expérience ont écarté bien des voies sans issue. Qu'il croie à ma reconnaissance la plus vive et à toute mon amitié.

Un système tel qu'IMAG3 s'appuie nécessairement sur l'expérience accumulée par une équipe. Aussi, je ne saurais trop souligner l'apport essentiel de celle qui a développé IMAG2 et qui, outre Monsieur LE FAOU, comprenait Messieurs JACOLIN, PIMORT, VERAN, CARRY.

La Société d'Etudes des Systèmes Automatiques s'est toujours attachée à la diffusion et au développement des systèmes IMAG2 et IMAG3. Elle a pris une part importante, par son aide matérielle, à la réalisation de ce travail. En particulier, je tiens ici à remercier Monsieur J.P. SICOT.

Enfin, je voudrais remercier

Mademoiselle G. BOULESTEIX, qui a assuré la dactylographie de cette thèse avec beaucoup de gentillesse, de soin et de goût ;

Le service de reprographie, qui a réalisé avec compétence le tirage de cet ouvrage.

Que ceux,nombreux,qui ne sont pas cités ne se croient pas oubliés. Ils sont l'amicale et indispensable ambiance de laquelle s'informe et s'interprète ce travail.

A mes Soeurs,

A la limite extrême où
l'obscur devient son propre rêve,
MIRACLE! ,ce rêve est la lueur.
Toujours vous avez su renouer
l'éphémère abîme.

TABLE DES MATIERES

- Chapitre III

"Il y a plus à faire à interpreter les interprétations qu'à interpréter les choses"

MONTAIGNE.

INTRODUCTION

Le cadre dans lequel nons nous plaçons est celui de la Conception Assistée par Ordinateur (C.A.O.) de circuits électroniques.

Actuellement, l'électronicien dispose d'un certain nombre d'outils informatiques destinés à l'assister dans son processus de création. L'utilisation de ces outils permet à l'ingénieur de mettre au point des ensembles de plus en plus complexes dans des temps minimaux, et surtout, lui permet 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.

Dans le domaine de la conception assistée par ordinateur, c'est probablement la conception de circuits électroniques qui a suscité jusqu'ici le plus d'efforts et connu les développements les plus importants. En particulier l'apparition des circuits intégrés a rendu obligatoire l'utilisation et le développement de ces outils.

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 de circuits intégrés pour lesquels les processus de production sont très complexes.

Ainsi sont apparus des outils permettant :

- la simulation des phénomènes physiques (diffusion, ...) se produisant dans les dispositifs semi-conducteurs. [VAN 71, Heyd. 72].
- * - la simulation du fonctionnement de composants discrets.
- la simulation analogique et l'optimisation de circuits complets.
- la simulation de circuits logiques [Merm. 73, Borr. 76].
- l'implantation des dispositifs, dessins de masques
- l'implantation sur plaquettes, dessins de circuits imprimés.

Ces outils couvrent de manière plus ou moins efficace toute la chaîne des processus de conception et de réalisation.

L'évolution qui s'affirme chaque jour vers une complexité plus grande des circuits intégrés, exige une adaptation et un perfectionnement des outils

existants ainsi que l'élaboration d'outils nouveaux.

Les travaux que nous décrivons ici se situent dans ce contexte et portent sur la simulation et l'optimisation de circuits analogiques.

A l'initiative de Monsieur KUNTZMANN, dès les années 65-66, une équipe de l'IMAG comprenant Messieurs JACOLIN et LE FAOU portait ses efforts sur l'analyse de circuits électroniques par ordinateur.

A cette époque, quelques programmes rudimentaires existaient aux Etats-Unis. Cependant, ceux-ci étaient très coûteux d'utilisation et souvent, ils ne pouvaient être utilisés que par l'auteur du programme lui-même.

L'équipe de l'IMAG parvenait à mettre au point un programme de simulation de circuits qui prenait le nom d'IMAG I (1968).

IMAG I ne permettait pas à l'utilisateur de définir des éléments non-linéaires quelconques. Il avait seulement à sa disposition - prédéfini dans le programme - un modèle de diode et un modèle de transistor.

Cette équipe se renforçait de Messieurs PIMORT et VERAN, et poursuivait ses travaux pour aboutir en 1970, au système IMAG 2, lequel prenait en compte des éléments non-linéaires quelconques.

Après une période de confrontation avec une utilisation industrielle qui permit de parfaire le produit et de juger de ses limites, les travaux se développèrent dans trois directions :

- restructuration des données de manière à permettre la simulation de circuits de dimension quelconque. (Le Faou, Véran).
- calcul des sensibilités et mise en oeuvre d'algorithmes d'optimisation (Carry, Le Faou, Véran).
- introduction d'algorithmes d'analyse en régime continu (méthode de Newton-Raphson avec paramètre) et en régime transitoire (méthode de Gear d'ordre 2) plus performants (Le Faou).

Ces travaux étaient en cours lorsque j'arrivai au laboratoire avec l'objectif d'améliorer les coûts d'analyse du système IMAG 2.

J'y introduisis une méthode de Gear à ordre variable (1 à 6), ce qui

me permit de me familiariser avec les problèmes numériques rencontrés dans l'analyse des circuits. Cette méthode permit un gain de l'ordre de 20 % sur l'analyse transitoire, ce qui pouvait paraître intéressant. Cependant, à examiner de plus près les raisons de ce gain, on s'apercevait qu'il était dû beaucoup plus au fait que la matrice de Jacobi du système n'était pas réévaluée à chaque pas d'intégration, qu'à la méthode elle-même.

J'acquis alors la conviction que l'introduction d'algorithmes locaux ne pourrait conduire à une diminution importante des coûts d'analyse et que le problème devait être reconsidéré dans son ensemble. Ceci conduisit à l'élaboration du système IMAG 3.

Dans le premier chapitre, après une étude succinte des différents types de circuits électroniques et des méthodes de représentation de ces circuits, j'examine le langage de description d'IMAG 2/3 à la lumière des développements récents dans le domaine des langages de programmation modulaire. Puis, après avoir exposé la méthode d'écriture des équations du circuit utilisée dans IMAG 2, je montre comment elle a été modifiée dans IMAG 3 et l'intérêt de cette transformation à la fois :

- par sa plus grande généralité
- par la possibilité d'en déduire des propriétés du circuit, destinées à faciliter son analyse.

Le chapitre II est consacré aux problèmes numériques rencontrés en simulation de circuits et en particulier dans le système IMAG 3. Pour le calcul des états stables, je propose une approche permettant d'aborder de manière plus rigoureuse ce calcul dans le cas de composants complexes. D'autre part, les problèmes posés par l'optimisation y sont examinés.

En particulier, une méthode de calcul du gradient d'une fonction objectif de la forme,

$$\phi(p) = \int_0^{T(p)} F(x, y, \frac{dy}{dt}, p, t)\, dt$$

plus simple à mettre en oeuvre que les méthodes de Hachtel et Rohrer ou du réseau adjoint, y est introduite.

Dans le chapitre III, nous tenterons de situer IMAG 3 dans l'ensemble des programmes similaires existants actuellement. En particulier, on trouvera une comparaison avec les systèmes ASTAP [Week 73] et IMAG 2.

Le système IMAG 3 est maintenant utilisé par de nombreux laboratoires publics et privés, aussi bien en France que dans le Monde. Il est devenu pour ces laboratoires un outil essentiel de travail. C'est dire que le passage du stade prototype au stade opérationnel a été effectué avec les nécessités qu'il comporte. En particulier un effort important de rigueur dans la mise en oeuvre et la rédaction d'une documentation a été nécessaire pour assurer la fiabilité du programme et en faciliter la maintenance.

De nombreux tests ont été effectués à partir d'exemples réels fournis par les électroniciens.

L'expérience montre que les cas réels, par leur diversité, leurs dimensions et leurs caractéristiques souvent imprévisibles, posent de sérieuses difficultés à des systèmes mis au point sur des cas théoriques. Alors que dans ces derniers, les difficultés se présentent la plupart du temps de manière isolées, celles-ci se combinent dans les exemples réels et c'est justement cette combinaison qu'il est difficile de maîtriser.

Ceci peut aller jusqu'à la remise en cause de certaines parties du système et l'abandon de solutions qui pouvaient paraître séduisantes, mais qui se sont révélées difficiles à appliquer dans un contexte d'utilisation réelle.

De tels tests ne peuvent s'effectuer qu'en contact étroit avec des utilisateurs ayant eux-mêmes une bonne expérience de l'outil et de ce que l'on peut en attendre.

De ce point de vue, nous avons été favorisés par la présence à Grenoble d'équipes travaillant dans le domaine de la conception de circuits intégrés. Sans elles, une mise au point efficace aurait certainement été impossible.

Signalons enfin que la présence à Grenoble du système CP/CMS a permis aux auteurs d'IMAG 2 de développer un système entièrement conversationnel, lui donnant ainsi dès les années 70, toutes les caractéristiques d'un système de C.A.O. moderne.

CHAPITRE I

De façon générale, les systèmes de C.A.O. peuvent s'envisager de deux manières distinctes selon que l'on s'intéresse à l'aspect utilisation ou à l'aspect conception interne.

Dans l'esprit du concepteur de système C.A.O., ces deux aspects sont liés puisque c'est en fonction du premier qu'il élabore le second. Aussi nous consacrerons la première partie de ce chapitre à introduire les propriétés essentielles des composants d'un circuit électronique. Cette étude sera brève et superficielle, mais nécessaire à une bonne compréhension de la suite de l'exposé.

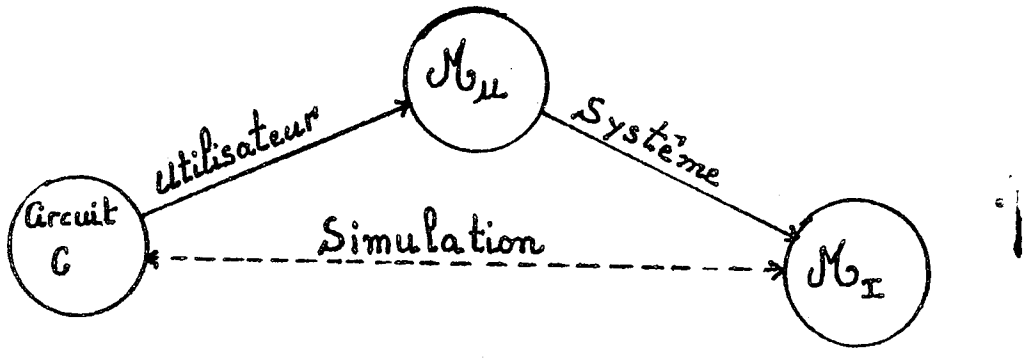
IMAG 3 est une extension d'IMAG 2 guidée principalement par la recherche du maximum d'efficacité, et on retrouve dans les deux systèmes de nombreuses caractéristiques communes.

Pour exposer la structure des systèmes IMAG 2 et IMAG 3 et montrer leurs différences, nous introduirons la notion d'invariant.

L'utilisateur construit un modèle du circuit qu'il désire simuler : M_u, et le communique au système. Cependant, pour que le système soit agréable à utiliser, ce modèle doit être le plus simple et le plus descriptif possible.

Le rôle du système est de fournir à l'utilisateur, à partir de M_u, un modèle M_I capable de simuler le fonctionnement du circuit réel C si on lui communique des ordres de simulation. Ce second modèle doit donc être capable d'interpréter ces ordres.

On se trouve en présence de la situation suivante :



D'un point de vue informatique, le modèle M_I peut se représenter par une machine abstraite M dont le registre d'instructions correspond aux ordres de simulation.

La réalisation de M est constituée d'une partie active algorithmique P_A et d'un ensemble de données caractérisant le circuit considéré = P_D.

Si l'on suppose que la partie algorithmique ne dépend pas du circuit considéré, mais est donné à priori, seule P_D est une caractéristique du circuit.

L'ensemble P_D doit caractériser le circuit quel que soit l'ordre de simulation fourni à M. Il doit donc être indépendant de la simulation elle-même, c'est-à-dire ne faire intervenir aucun renseignement tiré de cette simulation.

Ainsi, P_D sera constitué de renseignements que l'on peut déduire du modèle M_u du circuit et de ses propriétés formelles.
Ces renseignements seront appelés des <u>invariants</u> du circuit.

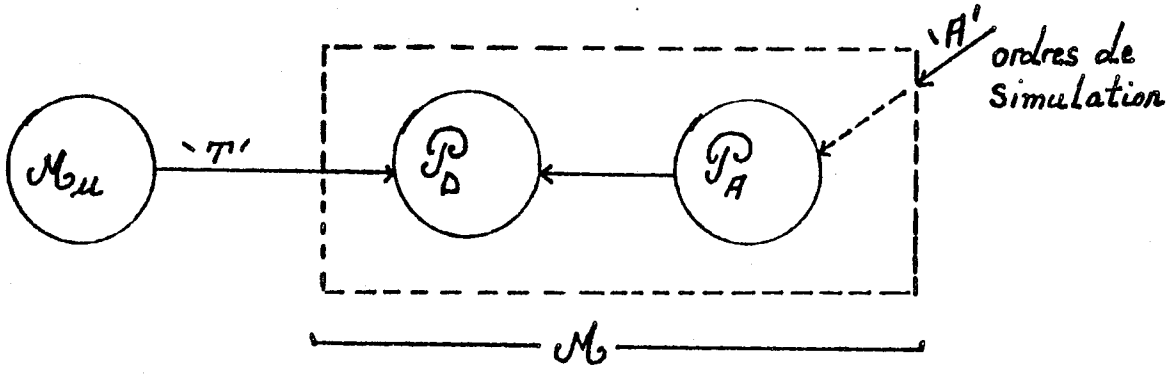
Nous pouvons distinguer deux phases dans le processus conduisant à la simulation :

- la première consiste à compléter la machine abstraite M à partir du modèle M_u.
- la seconde se rapporte à la simulation proprement dite, l'utilisateur active la machine M en lui communiquant des ordres de simulation.

Nous appellerons la première phase <u>traduction</u> et la seconde <u>analyse</u>.

Dans la phase de traduction T, le système effectue une analyse formelle du modèle M_u de manière à générer l'ensemble P_D.

Le processus global peut se schématiser de la façon suivante :



Bien entendu, du point de vue des possibilités et de l'efficacité du système, les parties P_D et P_A ne sont pas indépendantes.
Ainsi, parmi les différents couples possibles (P_D^i, P_A^j), il importe de choisir celui qui satisfera le mieux à certains critères que l'on peut se fixer. (Voir chapitre III).

Il nous est impossible d'effectuer une étude détaillée du système IMAG 2. Nous examinerons seulement les caractéristiques qui subsistent dans le système IMAG 3* et celles qui nous ont conduit à le transformer. Ces caractéristiques sont localisées dans la phase de traduction.
Ainsi, nous exposerons d'abord les caractéristiques communes aux deux systèmes, puis nous mettrons en évidence la différence entre les deux approches en montrant les points qui sont apparus insuffisants dans IMAG 2. Nous nous limiterons dans ce chapitre à la phase de traduction, renvoyant l'étude des algorithmes d'analyse au chapitre II.

* Les caractéristiques que nous supposerons communes ont en réalité évolué plus ou moins profondément. Un système tel qu'IMAG 2 ou IMAG 3 est un système vivant qui évolue constamment grâce à :

  - l'expérience acquise auprès des utilisateurs.
  - les progrès accomplis dans des disciplines utilisées par le système (compilation, etc...).

I - LES COMPOSANTS D'UN CIRCUIT ELECTRONIQUE

Le non-spécialiste se fait souvent une idée intuitive d'un circuit électronique sous la forme d'un ensemble de fils reliant entre eux des composants qui se présentent sous la forme de boîtes.
Bien que cette idée soit juste, elle recouvre des réalités très diverses. En fait, les types de circuit que l'on peut rencontrer sont si nombreux qu'il est nécessaire d'établir des critères permettant de les classer.

On peut distinguer les circuits électroniques par le type des composants qui les constituent, ou bien par leur comportement, que l'on peut préciser :

- par des mesures
- par la résolution des équations qui les décrivent.

Un composant électronique peut être défini comme une boîte avec deux ou plusieurs points de sortie accessibles au monde extérieur.
A chaque point de sortie j, on associe :

- une variable intensité i_j
- une variable tension v_j
- une variable charge q_j
- une variable flux ϕ_j

Chaque tension v_j est mesurée par rapport à une sortie de référence v_n.

Les variables q_j et ϕ_j sont liées aux intensités et tensions de sortie par les relations :

$$q_j(t) = \int_{-\infty}^{t} i_j(\tau)\, d\tau = q_j(t_0) + \int_{t_0}^{t} i_j(\tau)\, d\tau$$

$$\phi_j(t) = \int_{-\infty}^{t} v_j(\tau)\, d\tau = \phi_j(t_0) + \int_{t_0}^{t} v_j(\tau)\, d\tau$$

où t désigne la variable temps.

Les composants sont interconnectés entre eux par leurs points de sortie pour constituer un circuit électronique.

Dès lors, un circuit électronique se représente naturellement par un réseau [KUNTZ 72] où les articulations figurent les composants et les connecteurs les points de sortie.

I. 1 - Dipole

Un dipole est un composant qui ne possède que deux points de sortie. On distingue trois dipoles de bases :

- les résistances
- les self-inductances
- les capacités.

Les résistances sont entièrement déterminées par une fonction $f_r(v,i)$, les self-inductances par $f_i(i,\phi)$ et les capacités par $f_c(v,q)$.

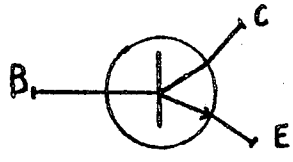
Si la fonction f associée au composant est linéaire, celui-ci est dit linéaire, dans le cas contraire, il est dit non-linéaire.

La fonction f peut dépendre d'un ou de plusieurs paramètres physiques. Dans ce cas, le dipole est dit contrôlé.

Si ce paramètre est le temps t, le dipole est "variant avec le temps" (time-varying).

I. 2 - N-poles

Un N-poles est un composant qui possède N points de sorties (N>2).
ex. : un transistor est un 3-poles :



Par analogie, on peut définir 3 N-poles de base :

- des N-résistances : caractérisées par un ensemble de relations portant sur les variables $(i_1, i_2, \ldots, i_{N-1}, v_1, v_2, \ldots v_{N-1})$
- des N-self-inductances caractérisées par un ensemble de relations portant sur les variables $(i_1, i_2, \ldots, i_{N-1}, \phi_1, \ldots \phi_{N-1})$
- des N-capacités caractérisées par un ensemble de relations portant sur les variables $(v_1, v_2, \ldots v_{n-1}, q_1, \ldots q_{n-1})$

I. 3 - Les sources d'excitations

Les sources sont des dipoles particuliers déterminés par les relations suivantes :

. $f(v, i) = v - E = 0$. Dans ce cas, on a une source de tension

. $f(v, i) = i - J = 0$. On a ici une source de courant

Si E et J sont des constantes, les sources sont dites indépendantes, dans le cas contraire, elles sont dites contrôlées.

I. 4 - Les modes de fonctionnement d'un circuit

On distingue 3 modes de fonctionnement pour un circuit électronique caractérisés par le type de réponse qu'il présente à l'effet de sources d'excitations d'entrée.

- Réponse en régime stabilisé (D.C.) : c'est la réponse d'un circuit linéaire résistif (constitué de résistances) à des sources d'excitations constantes. Ce type de réponse caractérise également le fonctionnement d'un circuit non-linéaire dont les sources sont constantes et qui a atteint un état stable.

- Réponse en fréquence (A.C.) : c'est la réponse d'un circuit de résistances, capacités, etc... (linéaires) à un signal d'entrée sinusoïdal.
- Réponse en transitoire : c'est la réponse en fonction du temps d'un circuit quelconque.

Dans IMAG 2 et IMAG 3, il est possible de prendre en compte ces trois modes de fonctionnement.

I. 5 - Classification des circuits électroniques

A l'aide des définitions précédentes, nous pouvons donner une classification des circuits électroniques en fonction des composants qu'ils contiennent et de leur comportement.

Nous reprenons, en l'adaptant au vocabulaire introduit, la classification donnée par Calahan [Ca 1972].

* Le tableau suivant contient une classification des circuits à la fois par les types de réponse associés et par les types d'éléments qu'ils contiennent.

Type du circuit	Propriétés	Composant	Réponse en courant direct	Réponse en fréquence	Réponse transitoire
Linéaire	Réponse proportionnelle à l'excitation	$\frac{v_R}{i_R} = c^{te}$, $\frac{\phi l}{il} = c^{te}$, $\frac{qc}{Vc} = c^{te}$	oui	oui	oui
Non-linéaire	Réponse non porportionnelle à l'excitation	$\frac{v_R}{i_R} \neq c^{te}$, $\frac{\phi i}{i_i} \neq c^{te}$, $\frac{qc}{Vc} \neq c^{te}$	oui	non	oui
Résistif (sans mémoire)	Pas d'énergie stockée	Résistances, sources contrôlées par des résistances	oui	oui	non
Dynamique	Stockage d'énergie	Selfs, Capacités, inductances	oui	oui	oui
Invariant avec le temps		f indépendant de t	oui	oui	oui
Variant avec le temps		f dépend de t	non	non	oui
Circuits à constantes localisées	Circuit pouvant se décrire par des équations algébriques ou différentielles				
Circuit à constantes réparties	Circuit décrit par des équations aux dérivées partielles				

I. 6 /- Schéma équivalent de N-poles

Le calcul direct de circuit contenant des N-poles est un domaine qui, à l'heure actuelle, reste encore ouvert.
La difficulté de ce problème tient essentiellement à la détermination de l'ensemble de relations caractérisant le N-poles.
Aussi, tous les programmes de simulation de circuits électroniques travaillent à partir de schémas équivalents.

On appelle schéma équivalent d'un N-pole un circuit électronique constitué uniquement de dipoles et présentant les mêmes caractéristiques de fonctionnement.

L'importance de l'étude de schémas équivalents de N-poles est évidente si l'on remarque que la précision d'analyse d'un circuit par un programme de simulation est directement fonction de la précision avec laquelle les schémas équivalents représentent les N-poles.

Il n'est pas dans notre intention de faire ici une étude même superficielle de cette discipline. Il nous paraît cependant indispensable de donner quelques éléments de base pour faciliter la compréhension de la suite de l'exposé.

Il existe deux approches classiques pour l'étude des N-poles :

- la première que l'on peut dénommer microscopique, appartient à la physique du solide. Les équations décrivant les composants contiennent des caractéristiques telles que :
  surface des jonctions, niveaux de dopages, mobilité des porteurs, etc... Les équations mises en jeu sont des équations aux dérivées partielles (équation de continuité, de distribution des charges, etc...). Cette approche devient inutilisable pour des circuits contenant plus de un ou deux composants parce que trop complexe et coûteuse.

- la seconde s'intéresse aux caractéristiques électriques que l'on peut observer sur les points de sorties. Le but de cette approche

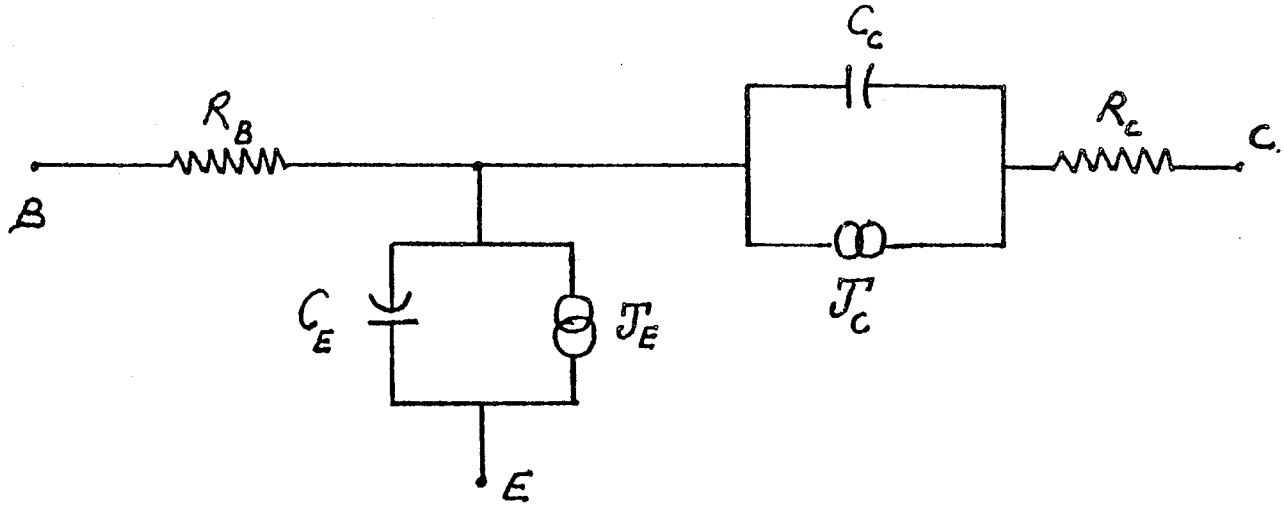
est de déterminer une expression mathématique pour traduire chaque caractéristique électrique. Nous appellerons cette approche électrique.

A partir des résultats de ces deux études, on peut définir des schémas équivalents de composants physiques comme les transistors (bipolaires, MOS,à effet de champ, etc...), les diodes, etc..., ou de circuits déjà réalisés : amplificateur, ... et considérés comme des boîtes noires. La topologie du schéma équivalent est définie par l'interconnexion de ses dipoles (Résistances, Selfs, Capacités, Sources).
Quand on le peut, chaque dipole (ou sous-ensemble de dipoles) est choisi pour représenter une caractéristique du composant. Ceci permet de déterminer le comportement du circuit en fonction des paramètres physiques de celui-ci, mais cette correspondance n'est pas toujours possible à réaliser.

Dans le cas d'un transistor bipolaire, les équations de la physique du solide montrent que les jonctions emetteur-base et collecteur-base peuvent se caractériser (entre autres) par des "capacités de jonction", et on utilisera des capacités pour représenter ces caractéristiques.

Il convient de souligner qu'un schéma équivalent donné ne représente généralement un composant complexe que dans un certain domaine de fonctionnement. Cette limitation est de la plus grande importance dans l'étude des circuits électroniques par ordinateur.

A titre d'exemple, nous donnons un schéma équivalent d'un transistor bi-polaire dérivé du schéma d'EBERS et MOLL. Nous utiliserons ce schéma par la suite.



$$J_E = I_{ej} - L_i . I_{cj}$$

$$J_c = I_{cj} - L_N . I_{ej}$$

$$I_{ej} = I_{eo} \{ e^{\frac{q}{kt} \cdot V_{b'e}} - 1 \}$$

$$I_{cj} = I_{co} \{ e^{\frac{q}{k.t.nc} V_{b'c'}} - 1 \}$$

$$C_E = T_e \cdot \frac{q}{k.t.ne} (J_E + I_{eo}) + \frac{CTEO.ke}{(V_{\phi E} - V_{b'e})}$$

$$C_B = T_c \cdot \frac{q}{k.t.nc} (J_c + 1_{co}) + \frac{CTCO.kc}{(V_c - V_{b'c'})}$$

q/kT : constante dépendant de la température

n_e, I_{eo} : constante et courant de fuite diode collecteur base

n_c, I_{co} : constante et courant de fuite diode collecteur base

L_1, L_N : gain en courant base commune en sens normal et inverse

R_B : résistance de base

R_C : résistance de collecteur

T_e : constante de diffusion

CTEO : constante de transition

$V_{\phi E}$: barrière de potentiel

Re : constante

} Diode émetteur-base

T_c : constante de diffusion

CTCO : constante de transition

$V_{\phi c}$: barrière de potentiel

kc : constante

} Diode collecteur-base

Notons pour terminer que de nombreuses recherches se poursuivent sur les schémas équivalents dans les buts de :

- simplifier ces schémas pour en faciliter l'analyse par ordinateur
- construire des schémas de N-poles plus complexes (nouveaux composants physiques ou circuits complets).
- construire des schémas plus précis.

Une équipe de Lyon animée par D. VANDORPE et avec laquelle nous collaborons s'intéresse à ces problèmes. Elle tente de représenter le comportement de N-poles complexes par des relations mathématiques choisies pour contenir un nombre minimum de variables tout en traduisant le fonctionnement du circuit avec une précision suffisante. Les premiers résultats de cette

étude sont encourageants [AZEN 77, LE FAOU 77].
L'étude des schémas équivalents est une discipline essentielle et préalable à l'étude des circuits par ordinateur, et sans elle, des systèmes comme IMAG 2 ou IMAG 3 perdraient une grande partie de leur intérêt.

II - Caractéristiques communes aux systèmes IMAG 2 et IMAG 3

Les points communs entre les deux systèmes sont essentiellement :

- le langage d'accès au système qui en est une caractéristique externe
- l'obtention des équations de fonctionnement du circuit (caractéristique interne).

Nous examinerons successivement ces deux points en montrant ce qui en fait l'originalité.

II. 1 - Caractéristiques externes :

L'utilisateur accède au système par un langage d'utilisation. On distingue deux niveaux dans ce langage :

- le niveau "description du circuit".
- le niveau "mode et condition d'analyse".

Le langage du premier niveau - que nous appellerons L_D - permet à l'utilisateur de décrire le circuit qu'il désire étudier et de le communiquer au système. (modèle M_u).

Le langage du second niveau - L_C - précise le mode d'analyse, ses conditions, ainsi que les résultats attendus par l'utilisateur : (ordres de simulation donnés à M).

Cette séparation est naturelle pour un électronicien puisqu'elle correspond aux deux étapes suivantes :

- construction d'une maquette
- prise de mesures sur la maquette en fonctionnement.

Bien que les concepts fondamentaux des langages L_D et L_C n'aient pas été remis en cause, ces langages ont été étendus par MM. LE FAOU et VERAN. Cette extension a été effectuée à la suite de l'expérience acquise de l'utilisation d'IMAG 2 par des électroniciens et dans le but de mieux répondre à leurs problèmes. Les résultats de ces travaux sont exposés dans [SESA 75].

Ce que nous dirons ici de ces langages est valable pour les deux systèmes.

Nous tentons de considérer le langage L_D à la lumière des recherches récentes dans le domaine des langages de programmation. Aussi, l'exposé pourra-t-il dérouter les utilisateurs d'IMAG 2 et d'IMAG 3 qui n'y retrouveront pas toujours la syntaxe à laquelle ils sont habitués.

II. 1-1- Le langage de description

Le problème, pour les concepteurs d'IMAG 2, était de définir un langage souple, simple à utiliser et permettant de décrire aisément un circuit électronique constitué des composants que nous avons examinés au paragraphe précédent.

Pour représenter des circuits, les électroniciens se servent de dessins qui ne sont autres que des représentations géométriques de réseaux. Il était donc naturel de construire un langage qui permette de décrire ces représentations géométriques.

Dans le réseau, chaque composant (articulation) est entièrement déterminé par les quatre informations suivantes :

- son type
- son nom
- sa position, c'est-à-dire la liste de ses connecteurs
- une liste de valeurs de paramètres (physiques ou électriques).

Nous étudierons successivement la représentation des types de base et les possibilités d'extension.

II. 1-1-1- Les types de base

Les dipoles de base (Résistance, capacité, self-inductances) et les sources (courant et tension) constituent les types de base, c'est-à-dire les types dont la définition est contenue dans le langage (prélude standard).

Donnons les définitions des types de base :

- une résistance est un dipole déterminé par l'équation caractéristique :

$$v_R = r. i_R .$$

- une capacité est un dipole déterminé par :

$$i_e = \frac{d_{qe}}{dt} = C. \frac{dV_c}{dt}$$

- une self-inductance est un dipole déterminé par :

$$v_L = \frac{d\phi}{dt} = l.\frac{di_L}{dt}$$

- une source de tension est un dipole caractérisé par :

$V_E = e$, quelque soit i_E déterminé par le circuit

- une source de courant est un dipole caractérisé par :

$i_T = j$, quelque soit v_J déterminé par le circuit.

i, c, l, e et j sont des paramètres, éventuellement variables, et dont la valeur détermine le fonctionnement du dipole.
Cet ensemble de relations est familier à l'électronicien.

Remarque : ces définitions peuvent sembler "aller de soi", cependant, on remarque qu'elles sont plus restrictives que les définitions données au paragraphe I. 1.
Une définition plus générale, pour une capacité par exemple, pourrait être la suivante :

$$\begin{cases} q_e = q\ (\rho) \\ v_c = v\ (\rho) \\ i_c = \dfrac{dqc}{dt} = \dfrac{dq.(\rho)}{d\rho}\ \dfrac{d\rho}{dt} \end{cases}$$

ou ρ est un paramètre.

L'avantage de cette représentation - outre sa généralité - est sa souplesse à aborder les difficultés numériques rencontrées lors de la résolution des équations du circuit.
En effet, toute transformation bi-univoque

$\rho' = g\ (\rho)$ conduit à un nouveau paramètrage

Dans ce sens, chaque dipole admet une infinité de représentations équivalentes, ce qui permet de modifier la représentation courante dans le cas de difficultés numériques.

Pour notre part, nous avions envisagé de redéfinir le langage dans ce sens. Cependant, outre les difficultés de mise en oeuvre considérables qu'elle présentait, cette approche risquait de dérouter l'utilisateur électronicien.

. Chaque type de base est nommé par une lettre :

- R : Résistance
- C : Capacité
- L : Self-inductance
- J : Source de courant
- E : Source de tension

. Les types de base permettent de définir des composants élémentaires qui en sont des exemplaires particuliers.

La syntaxe de la définition d'un composant élémentaire est la suivante :

< Composant élémentaire > : <nom de type> <identificateur>

(< liste de connecteurs >) < valeur du paramètre > $

exemple : R1 (1, 2) 10 $ C1 (A, B) 0.01 $

. L'identificateur est une suite de caractères alpha-numériques

exemples :

CTONC (E, B) 1 $ R BASE (B, BP) 100 $

. La liste de connecteur se réduit à deux éléments, chaque élément est représenté :

- par un entier
- par un identificateur alpha-numérique commençant par une lettre.

. La valeur du paramètre peut s'exprimer :

- par un nombre (entier ou réel)
- par une expression.

. Une expression peut être :

- une expression arithmétique
- une expression SI

. Les expressions arithmétiques sont analogues à celles que l'on rencontre dans le langage FORTRAN.

Les opérateurs admis sont les suivants :

- \+ Addition
- \- Soustraction
- * Multiplication
- / Division
- ** Puissance
- Des fonctions définies par l'utilisateur du système et écrites en FORTRAN
- Des fonctions standard (SIN, COS, etc...)

Les opérandes peuvent être :

. des nombres (entiers ou réels)

. des variables permettant de représenter des grandeurs attachées au circuit.

ex : tension ou intensité en un point du circuit.

. un composant élémentaire ; dans ce cas, sa partie valeur est utilisée dans le calcul de l'expression.

. des tables qui permettent de référencer des listes de valeurs définissant des courbes point par point (abscisse, ordonnée).

exemples : C1 (A, B) BE * ZE * IEO * AE + CTEO / ((V PHIE - V1) ** KE) $

E1 (1, 2) SIN (OMEGA * T + PHIE) $

E2 (3, 4) 'RAMPE' (T - 3) $

('RAMPE' est définie comme une table)

R2 (A, MASSE) R1 $

. Les expressions SI sont définies par :

< expression SI > : SI (< expression logique >) (< expression 1 >) (< expression 2 >) $

Les expressions logiques sont analogues, dans leurs constructions, à celles de FORTRAN mais n'utilisent que les opérateurs de relations SUPER et INFER.

. <expression 1> et <expression 2> sont des expressions qui peuvent être :

- des expressions arithmétiques
- des expressions SI

L'expression logique est d'abord évaluée ; le résultat de l'expression SI correspond au calcul de l'expression 1 si l'expression logique est "vraie" et au calcul de l'expression 2 dans le cas contraire.

exemples :

```
T1 (1, 2) SI (V1.SUPER. VZ) (0) (V1/R3) $

C1 (2, 3) SI (A. INFER.B) (SI (C.SUPER.D) (0.1) (2)) (V2**2) $
```

L'utilisation des expressions dans la partie "valeur" d'un composant élémentaire permet de représenter les variations de son paramètre caractéristique habituellement rencontrées.

Les unités utilisées pour exprimer ces valeurs sont celles du système M.K.S.A.

II. 1.1.2. Extensions

A partir des types de base définis ci-dessus, nous avons généré des composants élémentaires .

Sur l'ensemble des composants élémentaires, est implicitement défini dans le langage une opération de connexion C.

Deux composants élémentaires sont connectés, s'ils ont un connecteur commun, par ce connecteur.

exemple :

R1 (A, B) 10 $ R2 (B,C) 100 $

sont connectés par leur connecteur commun B.

Remarque : cette opération joue, dans le langage L_D, le même rôle qu'une opération définie entre deux éléments de "type" déterminé dans un langage de programmation.

entier A, B

C ← A + B

R1 (A, B)10$ R2 (B, C) 1$

R1, R2 ← R1 et R2 connecté par B

A l'aide de cette opération implicite C, on peut définir des assemblages de composants élémentaires pour former de nouveaux types admis par le langage.

Les auteurs d'IMAG 2 ont appelé ces types des modèles. Cette dénomination a été choisie parce qu'elle correspond à celle qu'emploient souvent les électroniciens pour désigner les schémas équivalents (assemblage de dipoles). Toutefois, bien que ce mot subsiste dans le langage, nous emploierons de préférence le mot TYPE et ceci pour deux raisons :

- rester cohérent avec la terminologie adoptée par les personnes travaillant dans le secteur des langages.

- ne pas introduire de confusion sur le mot MODELE tel qu'il est utilisé dans ce document.

Ces types sont définis par une déclaration et un corps.

- Déclaration :

TYPE :< nom de type > (< liste de connecteurs formels >) < liste de paramètres formels > $

. Le nom du type est un identificateur alphabétique choisi par l'utilisateur.

. Ces types sont généralement destinés à générer des N-poles, il est donc nécessaire de leur associer une liste de connecteurs formels destinée à recevoir les connecteurs réels des composants générés à partir d'eux.

. De la même manière qu'à un type de base on associe un paramètre, on associe à ces types généraux une liste de paramètres.

Ces paramètres peuvent être :

- des variables
- des composants (élémentaires ou non).

exemple : TYPE : TRAN (E, B, C) EPS, QKT, NE, NC, ALPHAN, ALPHAI, IEO, ICO, ZE, ZC, CTEO, VPHIE, KE, CTCO, VPHIC, KC, R1, R2 $

- Corps :
le corps de la définition contient la description de l'assemblage de composants utilisé pour construire le nouveau type.

- Génération :
à partir des types ainsi définis, nous pouvons générer des composants complexes. Cette génération s'effectue en deux étapes :

- spécification du composant

exemple :

un transistor BSW48

- l'implantation dans le circuit. Cette implantation peut être multiple.

Cette séparation permet de ne spécifier qu'une seule fois un composant que l'on désire implanter en plusieurs exemplaires dans un circuit, ce qui simplifie la description.

La spécification d'un composant s'effectue de la manière suivante :

COMPOSANT :'<nom du composant>'(<nom du type>)<liste de paramètres effectifs> $$

: Le nom du composant est un identificateur alpha-numérique commençant par une lettre

. La liste des paramètres effectifs doit correspondre à la liste des paramètres formels du type.

exemple :

COMPOSANT : ' TOTO ' (M) 1.3, 4, 'LULU' $$

Remarque : le lecteur habitué au langage de description d'IMAG 2 et 3 constatera une certaine liberté prise ici avec la syntaxe - mais non avec la sémantique - dans le but de garder une terminologie cohérente avec les travaux actuels sur les langages.

Ainsi défini, un composant peut s'implanter dans le circuit :

<nom du type> <numéro dans le circuit> (<liste de connecteurs effectifs>)
' < nom du composant> ' $

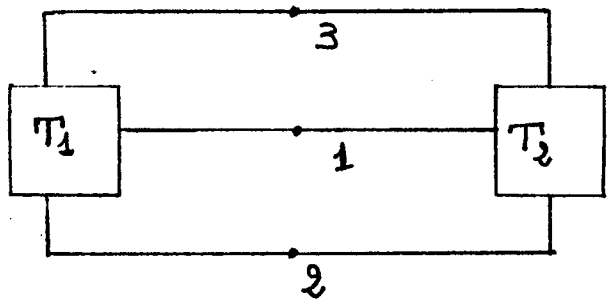
exemple :

M1 (1, 2, 3, 4) 'TOTO' $

- Les composants générés à partir de types généraux peuvent être assemblés par la même opération de connexion C que les composants élémentaires (connecteurs communs).

exemple :

T1 (1, 2, 3,) 'BSW48' $ T2 (1, 2, 3) 'LULU'



- Un type peut être défini à l'aide de composants complexes. On réalise ainsi une extension à plusieurs niveaux.

Ces possibilités d'extension font du langage de description un langage modulaire, c'est-à-dire un langage permettant de décrire des modules que l'on peut assembler entre eux.
Cette propriété est très intéressante et permet à l'utilisateur :

- de décrire et d'introduire aisément des composants décrits par des schémas équivalents
- de faciliter la mise au point d'un circuit de manière progressive (modules par modules).

II. 1.1.3. Déclaration de variables

On distingue deux catégories de variables :

- les variables électriques
- les variables générales

. Quand un électronicien prend des mesures sur un circuit, il utilise des appareils de mesures (voltmètres, ampèremètres).
Le langage doit permettre de décrire ces appareils.

Cette description s'effectue par l'intermédiaire des grandeurs qu'ils mesurent (tension, intensité) et par leur position dans le circuit.
Les variables électriques correspondent à cette notion.

<Déclaration de variable électrique> : <nom> <connexions>

- connexions désigne soit une liste de deux connecteurs quelconque, soit un composant bipolaire.

Si la variable est une tension, elle exprime la valeur (en volts) de la différence de potentiel entre les deux connecteurs désignés ou entre les deux connecteurs du composant bipolaire.

Si c'est un courant, elle exprime l'intensité (en ampères) du courant dans le composant. Seuls les composants appartenant aux types de base sont autorisés.

exemple :

$$R3\ (1,\ 3)\ \$ \qquad \begin{cases} V_1\ (R3)\ \$ \\ V_1\ (1,\ 3)\$ \end{cases} \qquad \{\ I1\ (R3)$$

- les variables générales sont implicitement de type réel. Elles n'ont pas à priori de signification particulière (dans ce sens, elles sont équivalentes aux variables d'un langage de programmation) et sont utilisées pour représenter des grandeurs physiques ou des quantités intermédiaires dans des expressions.

II. 1.1.4. Instruction d'affectation

<Affectation> : a = b

a désigne un composant élémentaire (dans ce cas, c'est la partie valeur qui est affectée) ou bien une variable générale.

b est un nombre, une variable, une expression arithmétique, une expression SI, un composant élémentaire.

exemple : R2 = R1 $ R1 = 3 $ AE = EXP (Q/CK*T) (NE* V1) $

J1 = SI (V1.SUPER.VZ) (o) (V1/R3) $

Remarques :.les affectations peuvent être données dans un ordre quelconque. C'est le système qui détermine l'ordre d'évaluation des instructions.

.tout bouclage est interdit. C'est-à-dire que le processus d'évaluation d'un élément ne doit pas utiliser (directement ou indirectement) la valeur de cet élément.

ex : A = B + 1 $

B = 2 * C $

C = EXP (A) $

est interdit.

II. 1.1.5. Description d'un circuit

La description d'un circuit est annoncée par le mot-clé DESC.

On distingue trois parties dans cette description :

- Définition des types utilisés (autres que les types de base)
- Génération des composants complexes à partir des types définis.
- Construction effective du circuit.

Une description de circuit se présentera de la façon suivante :

```
DESC
  * Définition des types $
  TYPE : A (          ) .....
         ..... $ $
  TYPE : B (          ) .....
         ..... $ $
  .....
  * Circuit  $
     E1 (1, 2) 10  $  R1 (2, 3) 100 $
     A1 (3, 4, 6) 'LULU'  $  .....
        ..... $ $
  * Composant  $
  COMPOSANT : 'LULU' (A) 10, ..... $
            : 'TOTO' (B) 1, .....  $
              ..... $ $     $
```

Le symbole * indique un commentaire et $ est un séparateur.

La définition des types doit précéder les deux autres parties, par contre, l'ordre d'apparition des blocs "circuit" et "composant" est arbitraire.

II. 1.1.6. Parallèle avec les recherches actuelles sur les langages de programmation

Défini dans les années 1966-69, le langage de description d'IMAG 2 rejoint, dans une certaine mesure, les travaux les plus récents sur les langages de conception modulaire de programmes : comme ALPHAR D [WULF 76] ou CLU [LISKOV 74].

Ces langages sont construits autour de la notion d'objet abstrait.

Un objet abstrait se définit par une certaine structure de données et par l'ensemble des opérations que l'on peut effectuer sur cette structure.

exemple : (terminologie ALPHARD)

```
form ISTACK (n : integer) =
        beginform
        specifications
        function
        push (...paramètres...) ;        } opérations définies
        pop  (.....) ;                   } sur l'objet
        top  (.....) ;                   } abstrait.
        representation
        ..... déclaration des variables internes...
        implementation
        ..... description des fonctions listées dans specifications...
        endform ;
```

Les variables internes sont des objets générés à partir d'objets abstraits déjà définis par le programmeur ou faisant partie des objets de base.

L'objet abstrait ISTACK ainsi décrit est une pile de profondeur n. Il peut être utilisé pour générer des piles particulières de profondeur 1, 2, 3, etc... utilisables dans un programme (ou dans la définition d'autres objets abstraits).

Il est bien évident qu'un langage comme ALPHARD ne poursuit pas le même but que le langage de description d'IMAG 2 et que sa sémantique est plus générale (en particulier il est algorithmique alors que le langage d'IMAG 2 est seulement descriptif). Cependant, les mécanismes mis en jeu sont analogues.

Nous donnons ci-après la description du schéma équivalent d'EBERS et MOLL pour un transistor, ce qui nous permettra de développer la comparaison.

```
TYPE : T (1, 2, 3) EPS, QKT, NE, NC, ALPHAN, ALPHAI, IEO, ICO, ZE, ZC,
                   CTEO,VPHIE, KE, CTCO, VPHIC, KC, R1, R2 $

R1 (2, 4) $   R2 (5, 3) $
J1 (4, 1) $
J2 (4, 5) $
V1 (J1)  $  V2 (J2) $
C1 (J1)  $  C2 (J2) $

BE = QKT/NE  $ BC = QKT/NC $
AE = EXP (BE * V1) $   AC = EXP (BC * V2) $
T1 = IEO * (AE - 1) - ALPHAI * ICO * (AC - 1) $
J2 = CIO * (AC - 1) - ALPHAN * IEO * (AE - 1) $
C1 = BE  * ZE  * IEO *  AE + CTEO / ((VPHIE - V1)**KE) $
C2 = BC  * ZC  * ICO *  AC + CTCO / ((VPHIC - V2)**KC) $$
```

Le lecteur pourra rapprocher cette description de la description formelle du schéma d'EBERS et MOLL faite au paragraphe I-5.

Ce schéma est un "transistor abstrait" qui sera utilisé pour générer des transistors particuliers (en fixant des valeurs pour EPS, QKT, etc...) qui entreront dans la définition d'autres "composants abstraits" ou dans la construction d'un circuit.

Nous dressons ci-après un tableau mettant en évidence les mécanismes communs à ALPHARD et au langage de description IMAG 2 tels qu'ils apparaissent dans les deux exemples ci-dessus.

	ALPHARD	IMAG 2
Mot-clé annonçant la définition d'un objet abstrait	FORM	TYPE
Fonctions définies sur l'objet	PUSH, TOP, POP	C : (connexion) définie implicitement
Paramètres des fonctions	paramètres de PUSH,...	connecteurs : 1, 2, 3.
Paramètres de l'objet	n	EPS, IEO, etc...
Variables internes	variables définies dans la partie représentation	J1, J2, V1, etc...
Description des fonctions	Dans la partie implémentation	Connexions des composants internes
	Programme	Description du circuit

Il semble que la différence essentielle réside dans l'impossibilité - dans le cas d'IMAG 2 - de définir des fonctions sur l'objet abstrait.

Cette restriction est naturelle dans le cas d'un circuit électronique car les seules relations qu'il est utile de définir sont les relations de connexions.

La souplesse, la puissance et la facilité d'emploi du langage de description est l'un des points essentiels des systèmes IMAG 2 et IMAG 3. Il a grandement contribué à leur succès auprès des utilisateurs. Ceux-ci y ont trouvé la possibilité d'accéder aisément à un ordinateur et de l'utiliser dans leur processus de conception.

Nous ne pouvons examiner ici les langages de description d'autres systèmes équivalents. Cependant, une étude attentive montre qu'il y est souvent conçu davantage comme un codage de données que comme un véritable langage de description.

II. 1.2. Le langage de commande

Une fois le circuit décrit, l'utilisateur dispose d'un langage de commande qui lui permet de préciser : le fonctionnement qu'il souhaite étudier, les mesures qu'il entend faire (c'est-à-dire les résultats qu'il désire obtenir en sortie du système), ainsi que des spécifications relatives au fonctionnement et des ordres particuliers aux algorithmes d'analyse. C'est le langage interprétable par la machine M.

Le système reconnaît 3 modes de fonctionnement correspondant aux commandes : (1)

- CONT
- TRAN
- ALTE

(1) Nous n'aborderons pas ici la commande OPTI (optimisation) que nous développerons en détail dans le chapitre II.

II. 1.2.1. La commande CONT

La commande permet d'obtenir les états stables d'un circuit.

Les circuits non-linéaires peuvent posséder plusieurs états stables ou aucun.

ex : une bascule possède deux états stables.

Si un circuit se trouve dans un état stable, seule une variation des sources d'excitations peut modifier cet état.

Généralement, on s'intéresse aux états stables d'un circuit pour des valeurs différentes des sources d'excitations, des composants ou de certains paramètres physiques qui lui sont associés.

Le langage de commande permet de spécifier de telles listes de valeurs. Voir [SESA 75].

On peut préciser les résultats attendus du calcul, ainsi que la forme sous lesquels on désire qu'ils soient présentés.

Il est possible d'obtenir les valeurs :

- des variables (électriques ou générales) déclarées dans la description.
- de composants élémentaires.

Les résultats peuvent être obtenus sous forme de courbes, ou de tableaux de valeurs (ou les deux à la fois).

Ces possibilités de sorties de résultats existent pour les trois commandes. (CONT, TRAN, ALTE).

II. 1.2.2. La commande TRAN

Elle permet d'obtenir la réponse transitoire du circuit - c'est-à-dire

les courbes de variations des variables du circuit en fonction du temps - pour des sources d'excitation données et à partir de conditions initiales éventuellement précisées-

Les paramètres à préciser sont les suivants :

- temps initial : TIN
- temps final : TMAX
- pas de sortie : HS

Le pas de sortie est l'intervalle de temps séparant deux points voisins d'une courbe de sortie.

D'autres paramètres peuvent être spécifiés. Leur signification exige la connaissance des méthodes d'analyses utilisées.
Nous les indiquerons lors de l'étude de ces méthodes au chapitre II.

Les conditions initiales sont :

- soit précisées par l'utilisateur [SESA 75]
- soit déterminées par le calcul d'un état stable correspondant à la valeur des sources d'excitation pour le temps TIN.

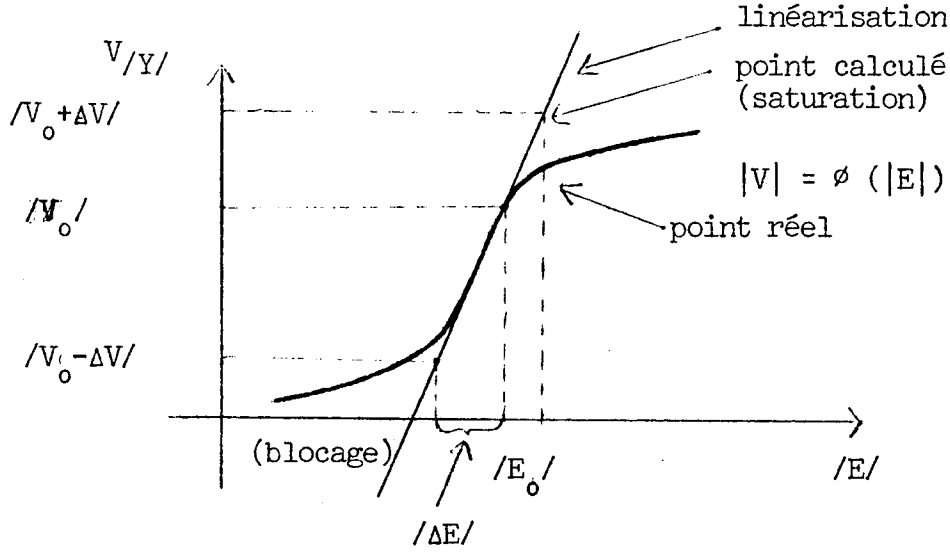
II. 1.2.3. La commande ALTE

Elle permet de calculer le comportement du circuit soumis à des excitations sinusoïdales pour une ou plusieurs fréquences données.

Lorsque le circuit comprend des éléments non-linéaires (diodes, transistors, etc...), il y a linéarisation autour du point de fonctionnement (ceci quelle que soit l'amplitude des signaux alternatifs).

Si l'on désire tenir compte des non-linéarités (effets de blocage ou de saturation), il est indispensable d'utiliser la commande TRAN.

En effet, soit V une variable fonction d'une source E, E_o et V_o correspondant au point de fonctionnement :



point calculé

Il est évident, sur cette figure, que les zones de blocage et de saturation ne peuvent être prises en compte.

Le langage permet de définir une excitation sinusoïdale par son module et sa phase ainsi qu'une suite de fréquences pour lesquelles le calcul devra être effectué. [SESA 75].

II. 1.2.4. L'instruction sensibilité (SENS).

L'instruction sensibilité peut s'utiliser avec les trois modes de fonctionnement.

Elle permet de calculer "les sensibilités" - c'est-à-dire la variation d'une quantité par rapport à la variation d'une autre quantité - de n'importe quelle variable du circuit par rapport à n'importe quel paramètre ou valeur de composant élémentaire constant dans la description.

$$(\text{SENS})\ V_{/a} = \frac{dV}{da}$$

Remarque : dans le cas d'une étude en régime transitoire, la sensibilité sera bien-entendu une fonction du temps.

Elle sera donc calculée pour chaque pas de sortie des résultats.

- On calcule également la dispersion globale de V. Si le calcul des sensibilités a été effectué par rapport à chaque élément a d'un ensemble A de paramètres, le calcul de la dispersion s'effectue de la manière suivante :

σ_a est la dispersion supposée gaussienne de a.

(son écart type est de 0,1. Il peut toutefois être modifié par l'utilisateur).

$$Disp_V = \sqrt{\sum_A \left(\frac{dV}{da} \cdot a \cdot \sigma_a\right)^2}$$

On obtient ainsi une répartition Gaussienne de $Disp_V$.

Remarque : le choix d'une répartition Gaussienne s'explique de la manière suivante :

- ce type de calcul est généralement utilisé pour déterminer l'influence des aléas de fabrication des composants sur le comportement du circuit et la répartition des valeurs réelles des éléments construits est fréquemment Gaussienne.

II. 2. Les aspects internes

Les points internes communs aux systèmes IMAG 2 et IMAG 3 sont localisés dans la phase de traduction T.

Nous considérons que la compilation du langage d'accès est identique. Signalons toutefois qu'elle a évolué - grâce à M. VERAN - dans le but de prendre en compte l'extension réalisée sur le langage L_D.

L'obtention des équations de fonctionnement du circuit est également identique. Cependant, là encore, les algorithmes utilisés ont été modifiés par M. LE FAOU pour optimiser le coût de cette obtention et prendre en compte des circuits plus volumineux.

Comme nous le verrons dans le paragraphe III, la forme finale des équations est différente dans les deux systèmes, aussi nous n'exposerons pas dans cette partie les transformations finales effectuées sur ces équations.

II. 2.1. Compilation du langage d'accès

Pour une étude plus approfondie de cette partie, nous renvoyons le lecteur à [JAC 70].

Une fois le circuit décrit, sa description est compilée en un langage intermédiaire. Cette phase utilise le transformeur de grammaire réalisé à l'IMAG par MM. Griffiths et Peltier. [GRIFF. 69].

Le choix d'utiliser un langage intermédiaire est rendu nécessaire par les deux points suivants :

- la description est lue et analysée instruction par instruction. Si une erreur est détectée, l'instruction fausse n'est pas prise en compte et l'utilisateur peut la réécrire aussitôt sans avoir à recompiler tout le programme (avec un système conversationnel).

- il est facile de modifier un circuit initial (changements de valeurs, suppression ou adjonction de composants) au niveau de la description.

La compilation est conversationnelle et incrémentielle.

Le langage intermédiaire est interprété pour fournir :

- les chaînes codées topologiques traduisant les interconnexions des composants.

- les expressions utilisées dans la description du circuit en écriture post-fixée.

Toute modification du circuit entraîne une modification correspondante du programme en langage intermédiaire. Si la modification affecte la topologie (suppression ou adjonction d'éléments), la phase d'interprétation se déroule à nouveau.

A partir du résultat de l'interprétation, les équations de fonctionnement peuvent être obtenues.

II. 2.2. Equations du circuit

Les équations de fonctionnement utilisées dans IMAG 2 et IMAG 3 reposent sur la notion d'état.

L'état d'un système à l'instant $t = t_o$ et ses entrées pour $t>t_o$ permettent de déterminer le comportement du système pour $t>t_o$.

Nous considérons un circuit électronique composé uniquement de dipoles des types R, C, L et éventuellement de sources E et J.

Les N-poles étant ramenés au cas ci-dessus grâce à des schémas équivalents.

Bashkow [BASK 1957] a démontré que l'état d'un tel circuit se caractérise par (1) :

- les tensions aux bornes des capacités
- les courants dans les selfs.

Cet ensemble de variables sera dit :

"ensemble de variables dynamiquement indépendantes".

(1) Ceci s'explique par le fait que l'énergie stockée par le circuit peut s'exprimer à l'aide de ces 2 types de variables seulement.

Les équations écrites par rapport à ces variables sont des équations différentielles du 1° ordre que l'on peut mettre sous forme canonique :

$$\frac{dy}{dt} = F(y,t) \quad (II.1)$$

Les avantages d'une telle forme sont les suivants :

- le nombre de variables est minimum et les équations peuvent être traitées par des méthodes numériques classiques.
- les conditions initiales du système différentiel correspondent à l'état initial du circuit : $Y(t_o) = Y_o$

L'obtention de la forme (I.1) se décompose en deux étapes :

- écriture des équations du circuit à l'aide de sa topologie
- élimination des variables non dynamiquement indépendantes.

La première étape s'inspire de l'algorithme de BRYANT (BRYA 59,62) mais s'en distingue par :

- l'introduction de certaines tensions de connecteurs.
- l'expression des lois de courants et de tensions.
- la possibilité de traiter directement des éléments de type E et J.

Ces choix ont été guidés par un souci d'efficacité du traitement permettant d'obtenir le système (II.1).

Ainsi, ils permettent d'effectuer l'élimination presque entièrement de manière formelle.

Le circuit ne contenant que des dipoles, son réseau représentatif se réduit à un graphe ordinaire orienté G. L'articulation est alors une

branche et ses deux connecteurs des noeuds.

Nous supposons le lecteur familiarisé avec les notions élémentaires de la théorie des graphes : matrice d'incidence, des cycles, des coupures, arbres, etc...

II. 2.2.1. Ecriture des équations

A chaque dipole sont affectées deux quantités :

- le courant qui le traverse
- la tension entre ses points de sorties.

Ce sont ces quantités qu'il s'agit de déterminer. A un circuit de NB dipoles seront attachées 2*NB inconnues. Il faut donc définir 2*NB équations indépendantes entre elles.

Les lois régissant le fonctionnement d'un circuit électronique sont : [Seshu 61]

- la loi de Kirchhoff aux coupures (LKN) : la somme algébrique des courants incidents à une coupure quelconque I est nulle.

$$\sum i_{b_I} = o$$

- la loi de Kirchhoff pour les cycles (mailles) (LKC) : la somme algébrique des potentiels de l'ensemble des branches de tout cycle C du graphe représentatif est nulle :

$$\sum v_{b_C} = o$$

- les lois de branche caractéristiques des composants :

$$V_R = R.\, i_R \;,\; i_c = C\,\frac{dVc}{dt},\; V_l = l\,\frac{dil}{dt} \;,\; V_E = E,\; i_J = J^{te}.$$

Ces relations de branches sont au nombre de NB et sont indépendantes entre elles puisque se rapportant chacune à un composant différent.

Les deux premières lois sont uniquement topologiques et peuvent être écrites par inspection du graphe G représentatif du circuit.

L'ensemble de ces trois lois va permettre d'écrire les équations de fonctionnement.

On peut montrer qu'en introduisant la matrice des cycles B et la matrice des coupures Q de G, les lois de Kirchhoff s'écrivent :

$$B \cdot V_b = o \quad (II.2) \quad \text{et} \quad Q.I_b = o \quad (II.3)$$

où V_b et I_b sont les vecteurs colonnes des tensions et courants de branches.

Nous noterons B_a (Q_a) une matrice de cycles (de coupures) dont le rang est égal à la dimension de l'espace des cycles (des coupures).

(II.2) et (II.3) s'expriment par

$$B_a.V_b = o \ (II.4) \quad \text{et} \quad Q_a.I_b = o \ (II.5).$$

Le problème consiste à choisir des matrices B_a et Q_a particulières permettant d'exprimer les équations de fonctionnement en fonction de V_b et I_b de manière aisée.

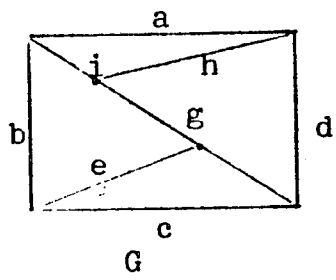
II. 2.2.1.1. <u>Définitions préliminaires</u>

Soit T un arbre de G, c'est-à-dire un sous graphe de G, connexe, contenant tous les noeuds de G et sans cycle.

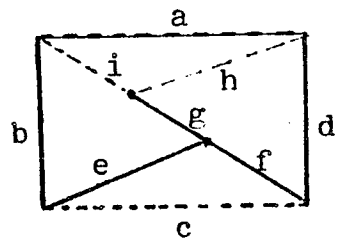
La construction d'un arbre T sur G divise l'ensemble des branches de G en deux parties :

- les branches de T : b_T
- les branches n'appartenant pas à T que l'on appelle des <u>maillons</u> : b_M.

exemple :



G



T (trait plein)

Branches de T : b,e,f,d,g

Maillons : a,c,i,h

Cette distinction permet de définir des F-cycles et des F-coupures :

- F-cycle : cycle de G formé d'un maillon et du chemin (unique) de l'arbre reliant les deux noeuds du maillon.

- F-coupure : coupure formée d'une branche de l'arbre et de l'ensemble des maillons dont les F-cycles associés contiennent la branche.

Etant donné un arbre T particulier, on peut montrer que les matrices des F-cycles et des F-coupures associées constituent des matrices B_a et Q_a particulières. (B_F et Q_F).

Par construction des matrices B_F et Q_F, les relations (II.4) et (II.5) s'écrivent :

$$\left[I_1 \cdot F \right] \begin{bmatrix} V_{T_M} \\ V_{T_A} \end{bmatrix} = o \quad \text{(II.6)} \qquad \left[K \; I_2 \right] \begin{bmatrix} i_{T_M} \\ i_{T_A} \end{bmatrix} = o \quad \text{(II.7)}$$

où I_1 et I_2 sont des matrices unitaires, F et K des matrices topologiques.

II. 2.2.1.2. Choix d'un arbre de G

Les propriétés précédentes sont utilisées pour formuler les LKN et LKC.

Les variables dynamiquement indépendantes étant les V_c et i_l, au vue de (II.6, 7), il est intéressant de construire l'arbre T de telle manière qu'il contienne le maximum de capacités et le minimum de selfs.

Un tel arbre sera dit arbre propre modifié (notation de Bryant).

On ne tient pas compte des sources de courant qui sont "ouvertes", ni des sources de tensions qui sont "court-circuitées", c'est-à-dire l'arbre propre modifié doit contenir toutes les sources de tension et aucune source de courant.

Nous utiliserons la notation de Bryant pour désigner les branches du graphe.

Maillons : α : capacités

β : résistances

γ selfs, J

Branches de l'arbre propre modifié :

δ : capacités

ε : résistances

$\mathfrak{g}$: selfs, E

A l'aide de cette notation, F_{xy} est une matrice représentant les branches de T de type y appartenant aux F-cycles formés sur les maillons de type x.

Ainsi, on peut vérifier que la matrice F dans (II.6) s'écrit :

$$F = \begin{bmatrix} F_{\alpha\delta} & 0 & 0 \\ F_{\beta\delta} & F_{\beta\varepsilon} & 0 \\ F_{\gamma\delta} & F_{\gamma\varepsilon} & F_{\gamma\gamma} \end{bmatrix} \qquad (II.8)$$

II. 2.2.1.3. Loi des cycles

Bryant utilise (II.6) comme loi des cycles. On l'exprime dans IMAG 2 un utilisant des tensions de noeuds. En effet, une partie du processus d'élimination consistant à exprimer des tensions de branches en fonction d'autres tensions de branches (V_δ), il est plus commode de passer par l'intermédiaire de tensions nodales.

Les tensions de branches se retrouvent immédiatement par différence.

-Soit NN le nombre de branches du graphe G.

Propriété : une condition nécessaire et suffisante pour que v_b satisfasse $B_a.v_b = o$ est qu'il existe NN - 1 variables V_{nj} telles que :

$V_b = Q^T V_{nj}$ ou Q est une matrice de coupures ayant NN - 1 lignes indépendantes.

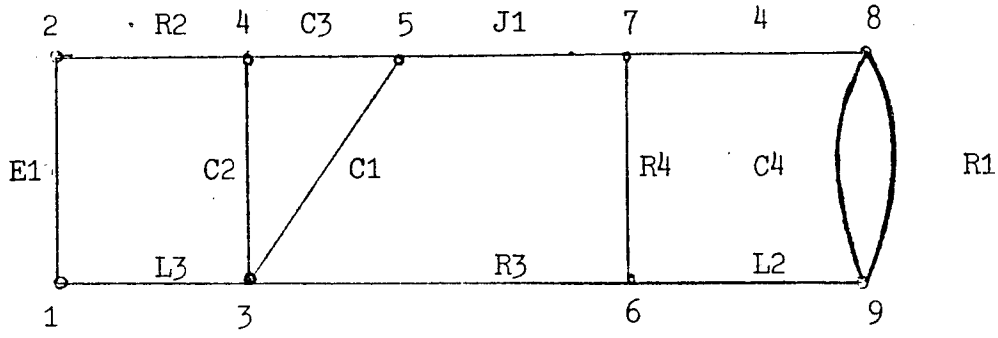
-Si Q = A, les V_{nj} sont des tensions nodales V_n. A matrice d'incidence.

La loi des cycles s'écrit dans IMAG 2 (et IMAG 3) :

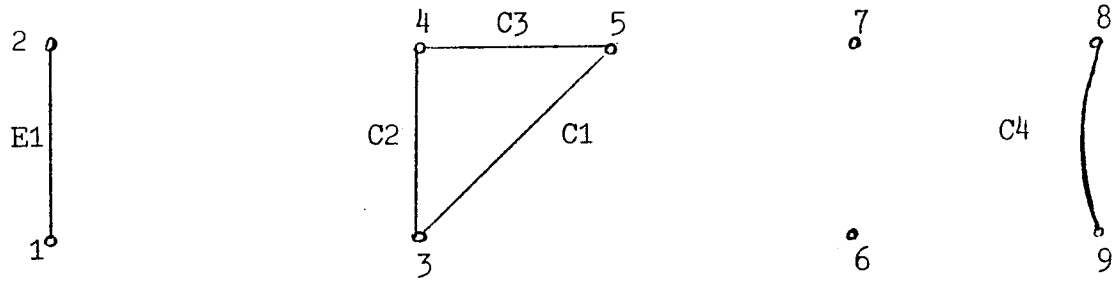
$$V_b = A^T V_n \qquad (II.9)$$

Soit G_c un sous-graphe de G ne contenant comme branches que les capacités et sources de tension, mais tous les noeuds de G.

exemple :



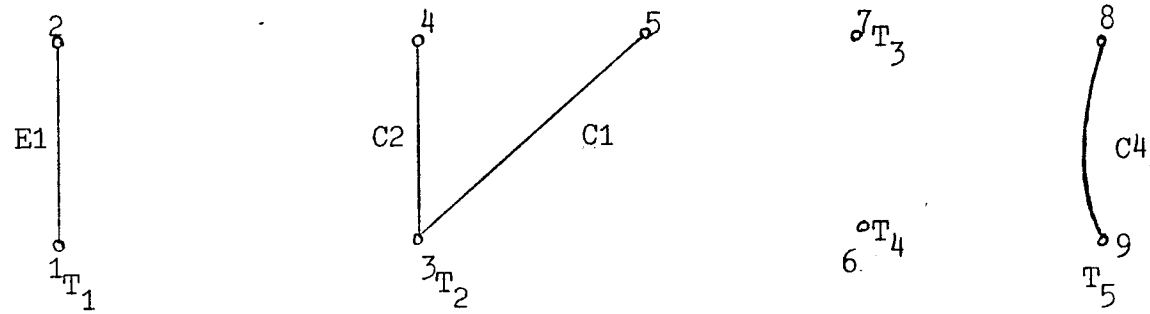
Graphe G



Graphe G_c.

- Soit T_c un arbre de G_c et T_i une composante de T_c .

exemple :



Arbre T_c.

- T_i a pour racine le noeud ρ_i et $V\rho_i$ en est sa tension nodale.

Soit v_n^i le vecteur des tensions des noeuds appartenant à l'arbre partiel T_i, v_n^i est obtenu par combinaison linéaire de $V_{\rho i}$ et des tensions de branches de T_i, V_b^i . T_i étant formé de capacités et sources de tensions, on a :

$$V_n^i = \mathcal{T}_i \begin{bmatrix} v_\delta^i \\ V_E^i \\ V_\rho^i \end{bmatrix} \quad \text{(II.10)}$$

Ceci étant vrai pour chaque arbre partiel T_i :

$$V_n = \mathcal{T} \begin{bmatrix} V_\delta \\ V_E \\ V_\rho \end{bmatrix} \quad \text{(II.11)}$$

Un noeud ρ_i est choisi comme masse du circuit et $v_{\rho i} = o$. Le vecteur colonne du second membre dans (II.11) a donc bien NN - 1 composantes.

La loi des mailles peut donc s'écrire :

$$V_b = (A^t \mathcal{T}) \begin{bmatrix} V_\delta \\ V_E \\ V_\rho \end{bmatrix} \quad \text{(II.12)}$$

ou sous la forme explicitée :

$$\begin{bmatrix} V_\alpha \\ V_\beta \\ V_\gamma \\ V_J \end{bmatrix} = \begin{bmatrix} F_{\alpha\delta} & F_{\alpha E} & o \\ F_{\beta\delta} & F_{\beta E} & F_{\beta\rho} \\ F_{\gamma\delta} & F_{\gamma E} & F_{\gamma\rho} \\ F_{J\delta} & F_{JE} & F_{J\rho} \end{bmatrix} \begin{bmatrix} V_\delta \\ V_E \\ V_\rho \end{bmatrix}$$

(II.13) (II.14) (II.15) (II.16)

$$
\begin{bmatrix} V_\delta \\ V_\varepsilon \\ V_\varphi \\ V_E \end{bmatrix} = \begin{bmatrix} 1_{\delta\delta} & o & o \\ F_{\varepsilon\delta} & F_{\varepsilon E} & F_{\varepsilon\rho} \\ F_{\varphi\delta} & F_{\varphi E} & F_{\varphi\rho} \\ o & 1_{EE} & o \end{bmatrix} \begin{bmatrix} V_\delta \\ V_E \\ V_\rho \end{bmatrix}
$$

(II.17) (II.18) (II.19) (II.20)

Les 1_{ii} sont des matrices unitaires.

Les matrices F_{xy} sont des matrices topologiques et ne contiennent donc comme termes que o, - 1 ou + 1.

On peut remarquer dès maintenant le rôle important tenu par les variables non dynamiquement indépendantes V_ρ, puisque la plupart des tensions de branches s'expriment en fonction d'elles.

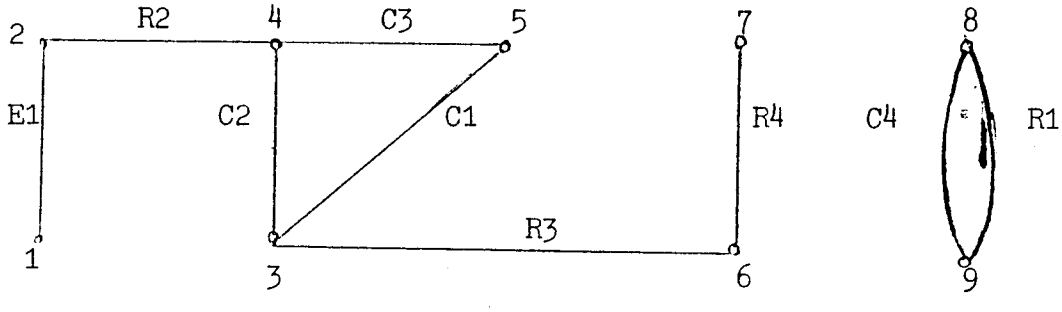
II. 2.2.1.4. Loi des noeuds

Ici encore, la formulation est différente de celle utilisée par Bryant (II.7). Elle utilise un ensemble de coupures particulier choisi par souci d'efficacité du traitement sur ordinateur.

Définitions :

G_{cR} : sous-graphe de G ne contenant que des capacités, des sources de tension et des résistances.

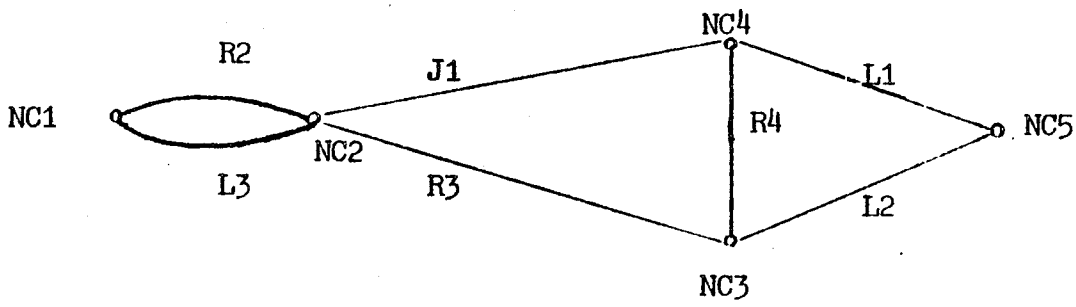
exemple : (pour le graphe G vu précédemment).



Graphe G_{cR}

G_c^* : Graphe G contracté, obtenu en court-circuitant sources de tension et capacités.

exemple :

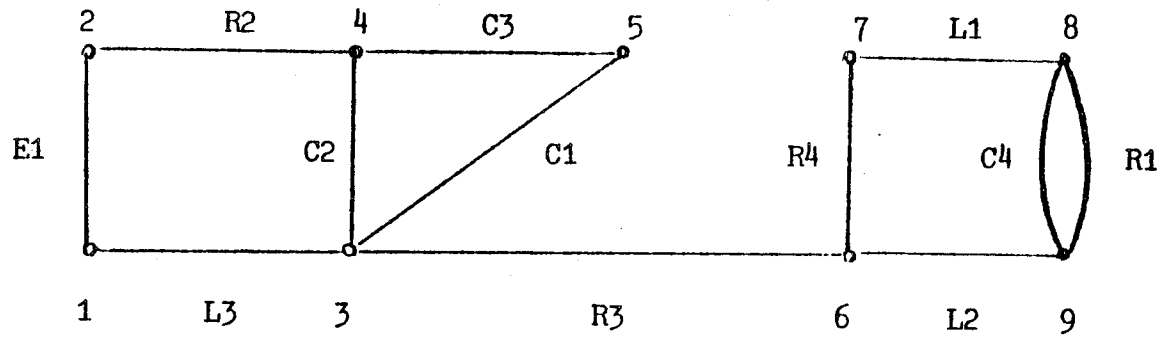


Graphe G_c^*

Les noeuds NCi correspondent aux composantes connexes du graphe G_c.

G_{cR1} : sous-graphe de G contenant les sources de tensions, les capacités, les résistances et les selfs.

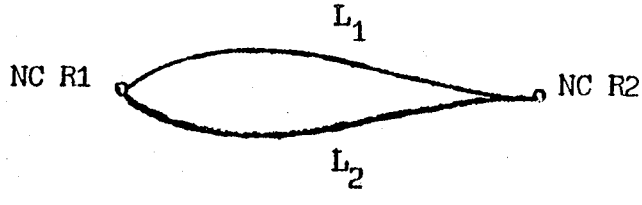
exemple :



Graphe G_{cR1}

G_{cR}^* : graphe G contracté obtenu en court-circuitant sources de tension, capacités et résistances.

exemple :

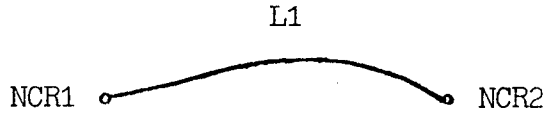


Graphe G_{cR}^*

Les noeuds NCR_i correspondent aux composantes connexes du graphe G_{cR}.

T_1^* : arbre sur G_{cR}^*

exemple :



Trois ensembles de coupures sont utilisés : celles-ci sont déterminées à l'aide des sous-graphes de G définis ci-dessus.

- coupures nodales sur G_{CR}^* : C^g.

Si G_{CR}^* a NCB noeuds, NCB - 1 coupures nodales seront indépendantes entre elles.

- coupures nodales sur G_C^* : C^ρ .

chaque noeud de G_{CR}^* contient NC_i noeuds de G_C^*. On prendra NC_i - 1 coupures nodales parmi les NC_i possibles pour chaque noeud de G_{CR}^*.

Si G_C^* a NCC noeuds, le nombre de coupures C^ρ est égal à :

$$\sum_{i=1}^{NCB} (NC_i - 1) = NCC - NCB$$ et elles sont indépendantes entre elles.

- coupures par rapport aux branches de T_C : C^f (une coupure pour chaque branche de T_C).

Le nombre de composantes connexes de T_C étant NCC, on a NN-NCC coupures C^f indépendantes entre elles car chacune isole une branche distincte de T_C.

Une coupure C^f ne peut être engendrée à l'aide de coupures C^ρ et C^g car elle contient une branche de T_C et est seule à contenir cette branche. Inversement, pour la même raison, une coupure C^ρ ou C^g ne pourra être engendrée

à l'aide de coupures C^f.

Il faut montrer que les coupures C^ρ et C^g sont indépendantes entre elles.

Si NCB = 1, il n'existe pas de coupures C^g. Soit donc NCB>1.

Soit i un noeud de G^*_{CR} autour duquel est formée une coupure C^g_i .

Ce noeud i "contient" NC_i noeuds de G^*_C donc $NC_i - 1$ coupures C^ρ : C^ρ_i .
Parmi les NC_i coupures nodales possibles, une n'a pas été formée. Soit C_j cette coupure et j le noeud de G^*_C associé.

Les C^ρ_i et C_j sont indépendantes entre elles car en nombre inférieur à NCC (puisque NCB>1).

Une combinaison des C^ρ_i ne pourra engendrer la coupure C^g_i car C^g_i isole un certain nombre de noeuds de G^*_C et parmi ces noeuds, le noeud j. Aucune des coupures C^ρ_i n'isole le noeud j. C^g_i et les C^ρ_i sont donc indépendantes entre elles.

Si l'on prend deux noeuds i et k de G^*_{CR}, les coupures C^ρ_i et C^g_j sont indépendantes entre elles car elles isolent deux noeuds de G^*_C différents.

On constitue bien ainsi un ensemble de coupures indépendantes

NCB - 1 coupures C^g
NCC - NCB coupures C^ρ
NN - NCC coupures C^f

NN - 1 coupures.

On forme donc une matrice Q à NN - 1 lignes et de rang NN - 1, ce qui permet d'écrire la loi des noeuds sous le forme :

$$
\begin{array}{ll}
\text{(II.21)} & \left[\begin{array}{cccccccc} Q_{\alpha\delta} & Q_{\beta\delta} & Q_{\gamma\delta} & Q_{J\delta} & 1_{\delta\delta} & Q_{\varepsilon\delta} & Q_{\phi\delta} & o \end{array}\right. \\
\text{(II.22)} & \left[\begin{array}{cccccccc} o & Q_{\beta\varepsilon} & Q_{\gamma\varepsilon} & Q_{J\varepsilon} & o & Q_{\varepsilon\varepsilon} & Q_{\phi\varepsilon} & o \end{array}\right. \\
\text{(II.23)} & \left[\begin{array}{cccccccc} o & o & Q_{\gamma\phi} & Q_{J\phi} & o & o & Q_{\phi\phi} & o \end{array}\right. \\
\text{(II.24)} & \left[\begin{array}{cccccccc} Q_{\alpha E} & Q_{\beta E} & Q_{\gamma E} & Q_{JE} & o & Q_{\varepsilon E} & Q_{\phi E} & 1_{EE} \end{array}\right.
\end{array}
\left[\begin{array}{c} i_\alpha \\ i_\beta \\ i_\gamma \\ i_J \\ i_\delta \\ i_\varepsilon \\ i_\phi \\ i_E \end{array}\right] = o
$$

(II.21) correspond aux coupures C^f par rapport aux capacités de l'arbre.

(II.24) correspond aux coupures C^f par rapport aux sources de tension.

(II.22) correspond aux coupures C^g.

(II.23) correspond aux coupures C^ρ.

L'équation (II.23) est remplacée par une équation permettant d'éliminer plus aisément i_ϕ.

On associe à l'arbre T_1^* un ensemble de coupures indépendantes C_f^g, équivalent à C_g, ce qui permet d'écrire l'équation :

$$\text{(II.23-bis)} : A_{\gamma\phi}\, i_\gamma + A_{J\phi}\, i_J + 1_{\phi\phi}\, .i_\phi = o$$

Les matrices Q_{ij} et A_{ij} sont des matrices topologiques de termes o, - 1 ou + 1.

Un tel ensemble de coupures peut paraître artificiel. Cependant, nous pouvons remarquer que sa construction utilise au maximum les renseignements tirés de l'arbre T_C construit pour exprimer la loi des cycles.

En effet :

- les coupures C^g utilisent les branches incidentes aux composantes connexes de G_{CR} dont l'obtention est une étape dans la construction de T_C.

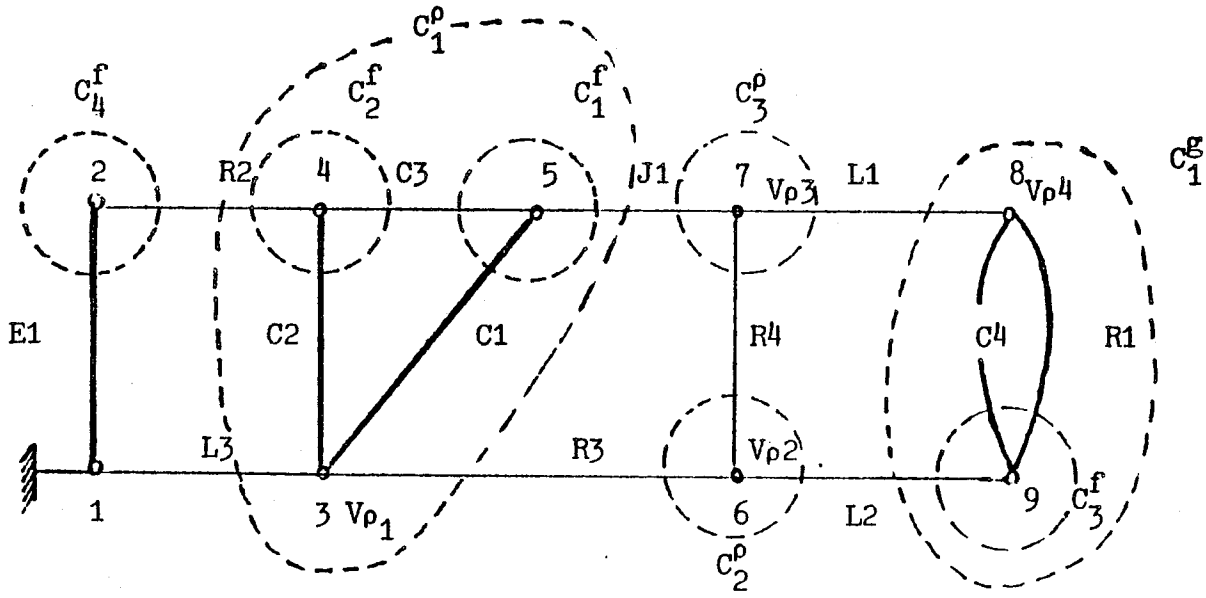
- les coupures C^{ρ} utilisent les branches incidentes aux composantes connexes de T_C, c'est-à-dire aux arbres partiels T_i.

- les coupures C^{f} sont construites par rapport aux branches de T_C.

Le travail nécessaire à l'obtention de cet ensemble de coupures est donc réduit au minimum dès que T_C est déterminé.

Dans l'exemple qui suit, nous indiquons les différentes coupures utilisées. Nous marquons en gras l'arbre T_C et indiquons la position des ρ_i.

exemple :



Ces coupures permettent d'écrire les équations :

$$C_1^{f} : i_{C1} + i_{C3} + i_{J1} = o$$

$$C_2^{f} : i_{C2} + i_{R2} + i_{C3} = o$$

$$C_3^{f} : i_{C4} + i_{R1} + i_{L2} = o$$

$$C_4^{f} : i_{E1} + i_{R2} = o$$

$$C_1^{\rho} : i_{R2} + i_{L3} + i_{R3} + i_{J1} = o$$

$$C_2^{\rho} : i_{R3} + i_{R4} + i_{L2} = o$$

$$C_3^{\rho} : i_{J1} + i_{R4} + i_{L1} = o$$

$$C_1^{g} : i_{L1} + i_{L2} = o$$

II. 2.2.2. Elimination des variables non dynamiquement indépendantes

Soit NB le nombre de branches du graphe.

La loi des cycles fournit NB - NN + 1 relations indépendantes, la loi des noeuds NN - 1, les lois des branches NB relations à 2 * NB inconnues.

Donc, au total 2 * NB relations à 2 * NB inconnues.

L'ensemble de ces relations caractérise entièrement le fonctionnement du circuit.

Cependant, pour obtenir un système d'équations plus aisé à résoudre, on cherche à exprimer ces relations en fonction des variables dynamiquement indépendantes V_δ (tension aux bornes des capacités de T_C) et i_l (courants dans les maillons selfiques) en éliminant les autres variables.

Ce processus d'élimination est exposé en détail dans [JAC 70] et nous n'y reviendrons pas ici.

On obtient ainsi les équations :

$$(II.25)\quad -(C_{\delta\delta} + Q_{\alpha\delta} \cdot C_{\alpha\alpha} F_{\alpha\delta}) \frac{dV}{dt} = (Q_{\beta\delta} \cdot R_{\beta\beta}^{-1} F_{\beta\delta} + Q_{\varepsilon\delta} \cdot R_{\varepsilon\varepsilon}^{-1} F_{\varepsilon\delta}) V_\delta$$

$$+ (Q_{\beta\delta} R_{\beta\beta}^{-1} F_{\beta E} + Q_{\varepsilon\delta} R_{\varepsilon\varepsilon}^{-1} F_{\varepsilon E}) \cdot E + (Q_{\beta\delta} R_{\beta\beta}^{-1} F_{\beta\rho} + Q_{\varepsilon\delta} R_{\varepsilon\varepsilon}^{-1} F_{\varepsilon\rho}) V\rho$$

$$+ (Q_{\gamma\delta} - Q_{\beta\delta} A_{\gamma\beta}) i_\gamma + Q_{J\delta} \cdot J$$

$$(II.26)\quad l_{\gamma\gamma} \frac{di}{dt} = F_{\gamma\delta} V_\delta + F_{\gamma E} \cdot E + F_{\gamma\rho} \cdot V\rho$$

$$(II.27)\quad (Q_{\beta\varepsilon} R_{\beta\beta}^{-1} F_{\beta\rho} + Q_{\varepsilon\varepsilon} R^{-1} F_{\varepsilon\rho}) \cdot V\rho = -(Q_{\beta\varepsilon} R_{\beta\beta}^{-1} F_{\beta} + Q_{\varepsilon\varepsilon} R_{\varepsilon\varepsilon}^{-1} F_{\varepsilon\delta}) V_\delta$$

$$- (Q_{\beta\varepsilon} R_{\beta\beta}^{-1} F_{\beta E} + Q_{\varepsilon\varepsilon} R_{\varepsilon\varepsilon}^{-1} F_{\varepsilon E}) E - (Q_{\gamma\varepsilon} - Q_{\beta\varepsilon} A_{\delta\beta}) i_\gamma - Q_{J\varepsilon} \cdot J$$

$$(II.28)\quad (Q^*_{\gamma\beta} l^{*-1}_{\gamma\gamma} F^*_{\gamma\rho} + Q_{\beta\beta} l^{-1}_{\beta\beta} F_{\beta\rho}) V_\rho = -(Q^*_{\gamma\beta} l^{*-1}_{\gamma\gamma} F^*_{\gamma\delta} + Q_{\beta\beta} l^{-1} F_{\beta\delta}) * V_\delta$$

$$- (Q^*_{\gamma\beta} l^{*-1}_{\gamma\gamma} F^*_{\gamma E} + Q_{\beta\beta} l^{-1} F_{\beta E}) E$$

ou Q^*, L^* et F^* sont obtenues à partir de Q, L, F, à l'aide de modifications exposées dans [JAC 70].

Les matrices R, C sont des matrices diagonales. L_{xx} et L_{yy} sont diagonales si aucune relation de mutuelle inductance n'existe. Dans le cas contraire, les coefficients de couplage font apparaître de façon symétrique des termes extra-diagonaux.
R, C et L proviennent de l'écriture des relations de branches.

L'expression particulière des lois des noeuds et des cycles a permis d'effectuer jusqu'ici les éliminations de manière formelle et finalement, les équations en sortie de l'analyseur topologique se présentent sous la forme :

(II.27.28.bis) $A_1 . VP = A_2 . Y + A_3 . E + A_4 . YL + A_5 . J$

(II. 25.bis) $A_6 \frac{dY}{dt} = A_8 . Y + A_9 . E + A_{10} . YL + A_{11} . J + A_7 . V\rho$

(II. 26.bis) $A_{15} \frac{dYL}{dt} = A_{16} . Y + A_{17} . E + A_{18} . V\rho$

Remarque :

Les équations (II.25) et (II.26) sont des équations différentielles non-linéaires tandis que (II.27) et (II.28) sont dans le cas général, des équations fonctionnelles non-linéaires.

Cette remarque est importante pour la validité des calculs que nous effectuerons sur ces équations par la suite.

A ce stade, nous n'avons pas obtenu la forme canonique (II.1) qui est utilisée dans IMAG 2. Cependant, comme elle n'est pas utilisée dans IMAG 3, nous renvoyons cette discussion au paragraphe suivant.

III. Différences entre les deux approches

Les équations (II.27.28.bis, II.25 et 26 bis) déterminent entièrement le fonctionnement du circuit. Notons les E_c.

Elles constituent l'invariant nécessaire et suffisant que nous noterons I_F.

Les approches adoptées dans IMAG 2 et IMAG 3 diffèrent sur l'utilisation qui est faite de cet invariant pour déterminer le fonctionnement du circuit.

On s'attache dans IMAG 3 à résoudre E_c en un nombre minimum d'opérations numériques de manière à minimiser les coûts de simulation.

Pour schématiser la méthode utilisée, considérons l'exemple très simple suivant :

- soit le problème P : calculer C = A * B où A, B, C, sont des matrices (n,n).

Un algorithme A résolvant ce problème effectuera n^2 multiplications.

Supposons que certains termes de A et B soient nuls et que nous les connaissions. Le problème P peut être transformé en P' :

- calculer C = A * B tel que $A_{i_1, j_1}, \ldots, A_{i_k, j_k}$ et $B_{l_1, m_1}, \ldots, B_{l_e, m_e}$ sont nuls.

Un algorithme A' adapté à P' n'effectuera pas n^2 multiplications mais un nombre inférieur qui sera fonction des termes non nuls.

Remarque : la transformation P → P' s'est effectuée en exprimant explicitement une donnée qui était contenue implicitement dans P.

Nous avons, dans IMAG 3, utilisé systématiquement l'approche ci-dessus et ceci à partir du problème P :

- déterminer le fonctionnement du circuit à partir de I_F.

Ainsi, nous avons transformé I_F en I'_F de manière à constituer le meilleur couple (I'_F, A) où A est un algorithme permettant de déterminer le fonctionnement du circuit à partir de I'_F .

Ce couple correspond à la machine abstraite M dont nous avons parlé au début du chapitre. Nous verrons dans le chapitre III qu'il a été choisi pour rendre t (T) + t (A) minimum. t (T) note le coût de la phase de traduction et t (A) celui de la phase d'analyse.

Dans IMAG 2, le problème était posé de manière différente dans la mesure où l'objectif était de formuler les équations de fonctionnement sous la forme canonique (II.1) afin d'obtenir un système différentiel de dimension minimum, ce qui permettait leur intégration par des algorithmes numériques classiques.

Nous indiquerons brièvement la démarche utilisée dans IMAG 2 en signalant les points où elle est insuffisante, puis nous développerons celle qui est utilisée dans IMAG 3.

Il est bien évident que la nature de ces démarches est très liée à la nature des algorithmes de résolution numérique utilisés dans la phase d'analyse. Cette étude étant reportée au chapitre II, il suffira de retenir ici que tous les algorithmes réellement utilisables demandent un grand nombre d'évaluations (>100) de la matrice des dérivées partielles des équations, J, et la résolution du système linéaire Jx = B.

III. 1. Démarche utilisée dans IMAG 2. Critiques

III. 1.1. Forme canonique

A partir des équations E_c, on peut obtenir la forme canonique (II.1) de la façon suivante :

supposons que l'on puisse écrire (II.27.28 bis) sous la forme :

$$V_\rho = A_1^{-1} \{A_2 \cdot Y + A_3 \cdot E + A_4 \cdot YL + A_5 \cdot J\} \quad (III,1).$$

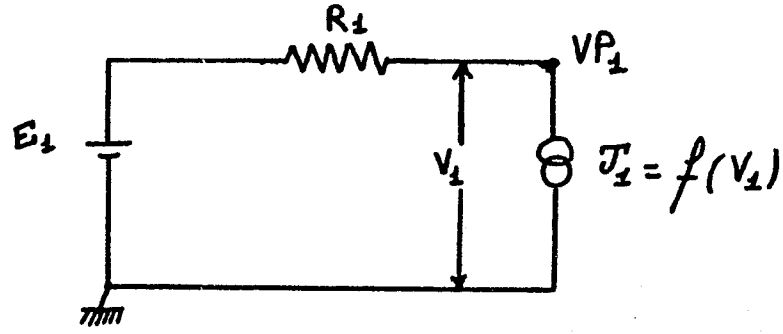
où le second membre ne contient pas V_ρ

Cette hypothèse peut se justifier dans la mesure où la transformation ci-dessus est possible pour un grand nombre de circuits.

Il existe cependant des exemples où elle est impossible.

Il est facile de voir sur les équations (II.25, 26, 27, 28) que si la valeur de certains composants R ou L ou de certaines sources E ou J s'exprime en fonction de $V\rho$, le second membre de (III.1) contiendra $V\rho$.

exemple : considérons le circuit suivant :



donc l'équation s'écrit :

$\frac{VP_1}{R_1} = - f(VP_1) + \frac{E_1}{R_1}$, donc le second membre contient VP_1. Ainsi, dans ce cas simple, la transformation ci-dessus ne peut s'appliquer.

En reportant (III,1) dans (II.25 et 26 bis) et en inversant les matrices A_6 et A_{15} du premier membre de ces équations, on obtient finalement le système que nous écrirons sous la forme condensée :

$$\frac{dy}{dt} = f(y, t) \quad \text{(III.2)}$$

ou $y = \{Y, YL\}$

Remarque : les restrictions ci-dessus permettent d'éliminer le problème difficile de l'unicité de la solution des équations fonctionnelles (II.27.28 bis). En effet, la fonction $V\rho(t)$ est ainsi exprimée comme solution (à chaque instant) d'un système linéaire.

(II.2) étant un système différentiel, l'unicité de la solution par rapport à des conditions initiales données peut être assurée si f vérifie les conditions d'unicité habituelle (voir paragraphe III.2.1).
Examinons de plus près la réalisation de cette transformation.
L'inversion des matrices A_1, A_6 et A_{15} ne peut pas, pratiquement, être réalisée formellement.
L'inversion formelle de matrices est un problème abordé par des systèmes spécialisés dans le calcul algébrique formel comme REDUCE [HEARN 73].
Cependant, les matrices considérées par ces systèmes sont de dimensions réduites. Malgré cela, le coût et l'occupation mémoire demandés par ces inversions demeurent prohibitifs.

L'inversion de A_6, A_1 et A_{15} est donc réalisée numériquement dans IMAG 2.

Ceci signifie que l'équation (III.2) n'est pas obtenue explicitement à la fin de la phase de traduction. Seule la valeur numérique de f sera accessible lors de la phase d'analyse.

(III.2) n'est pas un invariant du circuit au sens où nous avons défini ce terme.

La méthode de transformation que nous avons employée dans IMAG 3 ne pouvait donc s'appliquer à partir de l'équation (III.2).

Un autre inconvénient majeur de cette forme est le suivant :

- les équations (II.25, 26, 27, 28) montrent que les matrices A_i sont obtenues comme produits de matrices topologiques par des matrices généralement diagonales.

Si le circuit est volumineux, les matrices topologiques sont très creuses (contiennent un fort pourcentage de termes nuls) (1).

D'un point de vue algorithmique, cette propriété est très importante. Chaque équation du système E_C contiendra peu de variables, les expressions seront donc simples et peu coûteuses à évaluer.

Ceci n'est généralement pas vrai pour (III.2). En effet, l'inverse d'une matrice creuse ne l'est pas forcément, et f sera plus coûteuse à évaluer.

(1) Le graphe représentatif d'un circuit volumineux est généralement peu "dense". C'est-à-dire que le pourcentage des branches incidentes à un noeud donné est faible.

D'autre part, les non-linéarités éventuelles apparaissent de façon plus complexe dans f à cause des produits effectués.

Suivant les modes d'analyse demandés par l'utilisateur, l'équation (III.2) demandera une résolution particulière.

Ainsi, le calcul d'un état stable pour la valeur des sources d'excitations à $t = t_o$ demandera une valeur y telle que $\frac{dy}{dt} = o$.

On est amené à résoudre le système d'équations algébriques non-linéaires :

$$f\ (y,\ t_o) = o \qquad \text{(III.3)}$$

Le calcul d'une réponse en régime transitoire se ramène à l'intégration d'un système différentiel de conditions initiales du premier ordre :

$$\begin{cases} \frac{dy}{dt} = f\ (y,t) \\ \\ y\ (t_o) = y_o \end{cases} \qquad \text{(III.4)}$$

En régime alternatif, (III.2) est linéarisé pour aboutir à un système linéaire permettant de déterminer le vecteur complexe Δ_y en fonction de ΔE.

Chaque problème ci-dessus est un problème classique (ce qui ne veut pas dire que la solution en soit toujours simple) de l'analyse numérique.

Ceci justifie le choix de la forme canonique par les auteurs d'IMAG 2.

III. 1.2. Réalisation du calcul de la forme canonique dans IMAG 2

Un examen détaillé de cette réalisation exigerait des développements beaucoup trop longs. Aussi, nous nous contenterons de mettre en évidence ce qui en fait l'originalité par rapport à d'autres systèmes voisins comme ECAP2 [BRAN 71].

Les équations de fonctionnement du réseau sont obtenues sous la forme E_C et calculées sous la forme :

$$\left.\begin{array}{l} VP \leftarrow A_1^{-1} \cdot X \\ DY \leftarrow A_6^{-1} \cdot ID \\ DYL \leftarrow A_{15}^{-1} \cdot VG \end{array}\right\} \quad III.5$$

L'analyseur topologique fournit une chaîne codée permettant de calculer les termes non nuls du système E_C.

A chaque enregistrement de cette chaîne correspond un terme (ou tout une ligne) d'une de ces matrices.

Le traitement des termes en partie gauche et en partie droite de E_C est différent :

- partie droite :

Les matrices étant creuses, on ne les conserve pas sous la forme de tableaux, mais chaque terme non nul est repéré par son adresse dans une pile V de valeurs numériques et on génère une expression symbolique permettant de calculer le terme et le produit correspondant.

Les opérandes de ces expressions - en notation post-fixée - sont des adresses dans la pile V.

La suite d'expressions ainsi obtenues permet de calculer X, ID, et VG.

- partie gauche :

A_1 est traité comme un tableau dont les termes sont repérés par des adresses dans VG. On génère des expressions en notation post-fixée, permettant :

. l'appel d'un sous-programme de remise à zéro de ses termes dans V.

. le calcul de ses termes non nuls.

. l'appel d'un sous-programme permettant d'effectuer le produit A_1^{-1} . X .

A_6 et A_{15} sont séparés en plusieurs blocs diagonaux par permutation de lignes et de colonnes. Puis on génère l'inversion de chacun de ces blocs ainsi que les produits correspondants en partie droite.

On obtient finalement une suite d'expressions symboliques traduisant (III.5).

Ces expressions sont ordonnées les unes par rapport aux autres de manière à pouvoir être évaluées séquentiellement de façon correcte.

Remarque : la présence dans cette suite d'expressions d'appels à des sous-programmes numériques est la traduction de l'impossibilité d'obtenir la forme canonique comme un invariant.

Les expressions sont enfin traduites en code exécutable par un générateur de code.
Ce générateur est bien entendu dépendant de la machine hôte sur laquelle est mis en oeuvre le système.

A notre connaissance, dans la plupart des systèmes de simulation de circuits (en particulier ECAP2), les équations de fonctionnement sont traduites en langage évolué (généralement FORTRAN).
La génération s'en trouve vraissemblablement facilitée. Cependant, dans ce cas, le compilateur FORTRAN doit être appelé dynamiquement en cours d'exécution, ce qui a pour conséquence d'augmenter de façon significative le coût du traitement (l'expérience d'ECAP2 est significative à cet égard).

On peut penser que cette dernière approche rend le système indépendant de la machine hôte. En réalité, il n'en est rien, car l'appel dynamique du compilateur dépend fortement de la machine hôte et même de sa configuration particulière.

III. 1.3. Calcul de la matrice de Jacobi de f dans IMAG 2

La nécessité d'effectuer un grand nombre d'évaluations $J = \{\frac{\partial f}{\partial y}\}$ au cours de la phase d'analyse a conduit les auteurs d'IMAG 2 à rechercher une méthode économique pour ce calcul.

A partir de la suite d'expressions post-fixées ci-dessus, un dérivateur (sous-programme ECLAT) construit leurs dérivées par rapport aux variables d'état y.

Ces dérivées sont également traduites en notation post-fixée.

Nous ne pouvons détailler ici cet algorithme. Le lecteur intéressé pourra se reporter à la notice de programmation d'IMAG 2 [JAC. 70. a].

Chaque expression est décomposée en expression simple (ne contenant qu'un seul opérateur). Chaque expression simple est ensuite dérivée, puis les dérivées des expressions originales sont reconstituées en combinant les expressions simples.

On obtient ainsi une suite d'expressions post-fixées qui permettent de calculer la différentielle totale de f =

$$df = \sum_{i=1}^{n} \frac{\partial f}{\partial y_i} dy_i \qquad \text{(III.6)}$$

où df est un vecteur colonne.

Comme les expressions (III.5), les expressions dérivées sont traduites en code exécutable.

Au cours de la phase d'analyse, le calcul de J s'effectuera de la manière suivante :

- J est évalué colonne par colonne. Pour le calcul de la colonne i, dy_i est pris égal à 1, tandis que dy_j pour $j \neq i$ est pris égal à 0. (III.6) est évaluée en activant le code exécutable correspondant et les valeurs numériques de $\frac{\partial f}{\partial y_i}$ sont récupérées dans le vecteur df.

Si la matrice de Jacobi est de dimension N, son calcul nécessitera N évaluations de (III.6).

Remarques : nous pouvons faire deux remarques au sujet de cette approche.

- la première, à notre connaissance, est l'originalité de son emploi dans les systèmes de simulation de circuits électroniques.

Il semble en effet que dans les systèmes existants, J soit évalué par dérivation numérique.

C'est-à-dire que $\left[\frac{\partial f}{\partial y}\right]_{y=y_o}$ est évalué de la manière suivante (1) :

- on calcule f (y_o) puis f ($y_o + \Delta y_o$). Enfin :

$$\left[\frac{\partial f}{\partial y}\right]_{y=y_o} = \frac{f(yo + \Delta yo) - f(yo)}{\Delta y_o}$$

Il est difficile par cette méthode de garantir la précision des résultats numériques obtenus. Le problème de la dérivation numérique reste l'un des plus délicats de l'analyse numérique et le choix de Δy_o est difficile.

D'autre part, elle est sensiblement plus coûteuse que celle qui est utilisée dans IMAG 2.

- la deuxième remarque est relative à la forme des expressions traduisant (III.6). On pourrait penser qu'en faisant une analyse formelle de ces expressions, on puisse en déduire un ensemble d'expressions dont chacune d'entre elles correspondrait à un terme J_{ij} de la matrice J. Une telle démarche conduirait à un coût d'évaluation de J, N fois moins élevé que la méthode utilisée.

(1) Bien entendu, il existe des formules plus sophistiquées que celle que nous indiquons ici qui est la plus simple. Cependant, notons que dans ces cas,

Cependant cette analyse est impossible à cause de la présence dans les expressions d'appels à des sous-programmes numériques d'inversion. L'information relative à l'expression formelle des dérivées des inverses de A_1, A_6 et A_{12} ne peut être obtenue.
C'est l'une des raisons qui nous a amené à abandonner la forme canonique (III.2) dans IMAG 3.

III.2. Démarche utilisée dans IMAG 3

Le caractère généralement non creux de la forme canonique des équations ainsi que l'impossibilité pratique d'en obtenir une expression formelle, ce qui interdit d'en déduire des propriétés pouvant être utilisées dans la phase d'analyse, nous a conduits à l'abandonner dans IMAG 3.

Aussi, avons-nous décidé de travailler (1) directement sur les équations E_C que nous écrirons :

$$\begin{cases} G\,(VP, y, t) = o & \text{(III.7.a)} \\ F\,(y, \frac{dy}{dt}, VP, t) = o & \text{(III.7.b)} \end{cases}$$

ou de façon plus détaillée :

$$\text{(III.8)} \quad \begin{cases} \text{(a)} & -A_1\,(VP, y, t)\,.VP + A_2\,(VP, y, t)\,.Y + A_3\,(VP, y, t)\,.E + A_4\,.YL + A_5\,J = o \\ \text{(b)} & -A_6\,(VP, y, t)\,.\frac{dy}{dt} + A_8\,(VP, y, t)\,.Y + A_9\,(VP, y, t)\,.E \\ & \quad + A_{10}\,.\,YL + A_{11}\,.\,J + A_7\,(VP, y, t)\,.\,VP = o \\ \text{(c)} & -A_{15}\,(VP, y, t)\,.\,\frac{dYL}{dt} + A_{16}\,.\,Y + A_{17}\,.\,E + A_{18}\,.\,VP = o \end{cases}$$

(1) Ceci a été rendu possible par l'apparition d'algorithmes numériques (méthodes d'intégrations) permettant de prendre en compte le système (III.7). (Voir chapitre II).

ou A_4, A_5, A_{10}, A_{11}, A_{16}, A_{17} et A_{18} sont des matrices constantes (termes égaux à $\pm$ 1).
Y la tension aux bornes des capacités, et YL les courants dans les selfs.
y = {Y, YL}.

Dans ce paragraphe, nous examinerons les conséquences de l'introduction de l'équation (III.8) à la fois sur les possibilités de modélisation des circuits et sur son adaptation à améliorer le coût de la phase d'analyse par la détermination formelle de la matrice des dérivées partielles de F et G, ainsi que de la séquence d'opérations permettant de résoudre :

$$\left[\frac{\partial (G, F)}{\partial (y, VP)}\right] \left[x\right] = \left[B\right]$$

III. 2.1. Etude des équations E_C :

L'équation (III.8) est plus générale que la forme canonique utilisée dans IMAG 2. En particulier on peut remarquer que la valeur de tout composant du circuit peut être une fonction des variables d'état y ou de la tension des racines d'arbres partiels : VP.

Les restrictions sur le circuit comme celle du paragraphe III.1.1. sont donc levées.

Supposons que l'équation fonctionnelle (III.7.a) se mette sous la forme :

$$G_1 (VP) - k (y, t) = 0 \quad (III.9)$$

Si n_{VP} est la dimension du vecteur VP, l'équation (III.9) est un système de n_{VP} équations.

Les valeurs de k aux temps t_i dépendent à la fois de la valeur des sources d'excitations pour $t = t_i$ et des valeurs des variables d'états à ces instants.

(III.9) doit avoir une solution unique pour toutes les valeurs possibles pour k.

L'ensemble de ces valeurs n'étant pas connu a priori, on doit considérer que cet ensemble est l'espace $R^{nV\rho}$ tout entier.

Le problème est donc un problème global. Il est équivalent à celui de trouver l'inverse sur $R^{nV\rho}$ de la fonction G_1, c'est-à-dire :

$$V\rho = G_1^{-1}(k) \qquad (III.10)$$

G_1^{-1} existe si et seulement si la transformation $G_1(V\rho)$ est une fonction bijective.

Un théorème dû à Palais [Pal. 59] donne les conditions nécessaires et suffisantes pour cette existence.

Théorème de Palais :

Soit $G_1 : R^{nV\rho} \longrightarrow R^{nV\rho}$ une fonction de $R^{nV\rho}$ dans $R^{nV\rho}$.

(A) Si $G_1(V\rho)$ a des dérivées partielles du 1er ordre continues pour tout $V\rho \in R^{nV\rho}$, les conditions nécessaires et suffisantes pour que tout G_1^{-1} existe sur $R^{nV\rho}$ et possède des dérivées partielles du premier ordre continues pour tout $V\rho \in R^{nV\rho}$ sont : 1)le Jacobien de G_1 n'est jamais nul.

$$2)\lim //G_1(V\rho)// = \infty$$
$$//V\rho// \to \infty$$
$$\text{ou } //V\rho// \equiv \left[\sum_{i=1}^{n} V\rho_i^2\right]^{1/2}$$

(B) Si $G_1(V\rho)$ est continue, alors les conditions nécessaires et suffisantes pour que G_1^{-1} existe et soit continue sont données par la condition (2) ci-dessus et par la condition que $G_1(V\rho)$ soit localement bijective pour tout $V\rho \in R^{nV\rho}$.

Remarque : les conditions (A) ne sont pas nécessaires à l'existence de G_1^{-1}, mais assurent que ses dérivées partielles sont continues.

Considérons l'exemple suivant :

$$\begin{cases} k_1 = V\rho 1 \\ k_2 = V\rho_2 + (V\rho_3^2 - V\rho_3) \\ k_3 = V\rho_3 - V\rho_4 \\ k_4 = V\rho_4 \end{cases}$$

$\underline{\det}\ (J\ (V\rho)\ ) = 3\ V\rho_3^2$

La condition (A.1) n'est pas vérifiée par tout $V\rho$ dont $V\rho_3 = o$. Cependant, G_1^{-1} existe et s'écrit :

$$\begin{cases} V\rho_1 = k_1 \\ V\rho_2 = -k_3 + (k_2 + k_3)^{1/3} \\ V\rho_3 = (k_2 + k_3)^{1/3} \\ V\rho_4 = k_4 \end{cases}$$

On peut montrer aisément que G_1^{-1} vérifie les conditions (B).

Remarque : la formulation adoptée dans IMAG 2 vérifie les conditions du théorème de Palais si le système linéaire $A.V\rho = B$ à une solution unique $\forall y$ et t.

La formulation plus générale (III.7) ne vérifie pas toujours les conditions de ce théorème. En effet, on ne peut toujours mettre (III.7.a) sous la forme (III.9) et même dans ce cas, (A) ou (B) peuvent ne pas être vérifiées. On aura donc un certain nombre de restrictions sur la valeur des composants non-linéaires (cependant moins fortes que dans IMAG 2) pour que (III.7.a) satisfasse au théorème de Palais.

Le système différentiel (III.7.b) doit posséder également une solution unique.

Les théorèmes d'existence ou d'unicité comme ceux de Picard et de Wintner supposent le système différentiel mis sous la forme canonique.

Remarquons que (III.7.b) sous la forme détaillée (III.8.b, III.8.c) peut être mise sous la forme canonique puisque nous supposons que les A_i ne contiennent pas $\frac{dy}{dt}$. L'étude de l'existence et de l'unicité peut donc être faite sur la forme :

$$(III.7.b)' \quad \left\{ \begin{bmatrix} \frac{dY}{dt} \\ \frac{dYL}{dt} \end{bmatrix} = \begin{bmatrix} -A_8^{-1}\left\{A_8 . Y + A_9 . E + A_{10} . YL + A_{11} . J + A_7 . VP\right\} \\ -A_{15}^{-1} . \left\{A_{16} . Y + A_{17} . E + A_{18} . VP\right\} \end{bmatrix} = g\,(t, y, V\rho\,(y))$$

Le théorème de Picard permet d'assurer l'existence et l'unicité du problème de conditions initiales $\begin{cases} \frac{dy}{dt} = g\,(y, t) \\ y\,(t_o) = y_o \end{cases}$ sur un intervalle $[t_o, t_1]$.

Soit D un domaine de R^n : $(y_1, \ldots, y_n)$ alors :

Théorème de Picard :

Si g(y, t) est Lipchitzienne [GAST.68] sur D, (uniformément en t), alors le problème A a une solution unique sur $[t_o, t_1]$.

Toutefois, ce théorème est un théorème local. Si l'on cherche une solution correcte pour tout t, il faut avoir recours au théorème de Wintner :

Théorème de Wintner : [WINT.45]

Soit g(t, y) continue sur R^{n+1} : $(t, y_1 \ldots, y_n)$. Si l'on peut trouver une fonction L(r) continue $(r = y_1^2 + \ldots + y_n^2)$ telle que $/g_i\,(t, y_1 \ldots y_n)/ < L(r)$ (i=1, n) pour toutes valeurs de $r \in [o, +\infty[$ et si

$$\int_o^\infty \frac{dr}{L(r)} = \infty$$

alors toutes les solutions de (A) sont définies pour tout $t \in]-\infty, +\infty[$.

Une corollaire de ce théorème plus utilisable dans la pratique est le suivant :

Corollaire :

S'il existe une constante $K < \infty$ telle que :

$$/g_i\ (t, y_1, \ldots y_n)\ / < K\ .\ \max\ (/y_1/, \ldots, /y_n/)$$

quand $\max\ (/y_1/, \ldots /y_n/) \rightarrow \infty$, alors, toutes les solutions de A sont définies pour tout $t \in\ ]-\infty, +\infty[$.

Notons que les conditions requises par ce corollaire sont extrêmement fortes. En particulier, il exclue tous les polynomes de degré supérieur à 1.

Les conditions de ces théorèmes sont des conditions suffisantes, c'est-à-dire que g peut ne pas satisfaire au théorème de Picard et A posséder une seule solution.

La vérification par programme de telles conditions est pratiquement impossible. Aussi, nous supposerons que les équations (III.7) ont une seule solution.

Toutefois, dans les cas où des solutions suspectes sont trouvées, ou en cas de difficultés dans les algorithmes numériques, une vérification pourra s'effectuer a postériori.

Il apparaît que les problèmes d'unicité de la solution du système différentiel sont les mêmes dans IMAG 2 et IMAG 3.

Dans le cas où l'équation fonctionnelle (III.7.a) satisfait aux restrictions imposées dans IMAG 2, elle a une solution unique. Dans les autres cas, seule une étude de cette équation par le théorème de Palais pourra nous renseigner.

III. 2.2. Calcul des dérivées partielles de $\mathcal{E}_C$.

Comme nous l'avons vu, le calcul des dérivées partielles - nécessaire à la phase d'analyse - de f exige N évaluations des expressions (III.6) dans IMAG 2. Notre objectif est de construire l'expression formelle des dérivées partielles de (III.7).

En effet, dans IMAG 2, chaque terme $\frac{\partial fi}{\partial yj}$ se trouve évalué N fois.

La connaissance formelle des dérivées partielles des équations de fonctionnement permet d'éliminer cette redondance et le coût d'évaluation s'en trouve amélioré d'un facteur N.

Il est relativement difficile de donner une estimation quantitative du gain apporté par cette méthode sur le coût global d'utilisation du système sinon sur des exemples particuliers car elle dépend de plusieurs facteurs.

- dimension du circuit (N)
- complexité des éléments non-linéaires (complexité des expressions précisant leurs valeurs).
- difficultés d'analyse (nombre d'évaluations des dérivées partielles exigé).

Statistiquement, on peut évaluer à 40 % le temps d'évaluation des dérivées partielles par rapport au temps total $T_2 = t(T) + t(A)$ dans le cas d'IMAG 2. Ce qui permet d'envisager un temps T3 :

$$T3=T2 - 0.4 * T2 + (\frac{0.4 * T2}{N}).$$

Ceci sans tenir compte du gain apporté par l'évaluation de (III.7) par rapport à la forme canonique (III.2) (Système d'équations plus creux et sans inversion de matrices).

III.2.2.1. Possibilité d'emploi de REDUCE ou FORMAC.

Nous avions, dans un premier temps, envisagé d'utiliser pour ce calcul, des systèmes généraux de calcul formel comme REDUCE ou FORMAC,

pensant que ceci nous conduirait à un système aux possibilités plus générales qu'un programme de calcul formel construit par nous-mêmes.

Cependant, nous avons abandonné cette idée pour les raisons suivantes :

- ces systèmes sont utilisables sur un nombre relativement restreint de machines, ce qui nous faisait perdre la transportabilité d'IMAG 3.

- les langages d'accès de ces systèmes sont des extensions de langages de hauts niveaux type PL/1 ou LISP. On se trouvait ainsi devant l'alternative suivante :

générer les équations (III.7) dans le langage correspondant, puis :

- soit interrompre le système IMAG 3 pour poursuivre le traitement.

- soit appeler dynamiquement REDUCE ou FORMAC.

Dans le premier cas, l'aspect conversationnel d'IMAG 3 était perdu. Dans le second, les difficultés de mise en oeuvre étaient considérables et la présence simultanée de deux systèmes en mémoire exigeait un espace important (environ 100 K octets pour IMAG 2 en utilisant une technique de recouvrement, plus 100 K octets pour le système de calcul formel).

D'autre part, on se trouvait confronté au problème de l'augmentation des coûts de traitement provoqué par l'appel dynamique d'un compilateur.

III. 2.2.2. Algorithme de calcul

Les équations (III.8) sont disponibles sous la forme d'une notation post-fixée. Pour éviter une redondance dans le calcul de certains termes y figurant plusieurs fois, la forme de ces expressions est la suivante :

- c'est une suite d'affectations contenant un terme en partie gauche et une expression en notation post-fixée en partie droite. Cette suite d'affectations effectuées dans l'ordre permet d'évaluer F et G de (III.7). Cette liste se compose comme suit :

$$
\left.\begin{array}{l}
E \longleftarrow E\,(Y,\ YL,\ VP,\ t,\ k) \\
J \longleftarrow J\,(Y,\ YL,\ VP,\ t,\ k) \\
f_1 \longleftarrow g_1\,(Y,\ YL,\ VP,\ t,\ k) \\
b_2 \longleftarrow g_2\,(Y,\ YL,\ VP,\ t,\ k,\ b_1) \\
b_\rho \longleftarrow g_\rho\,(Y,\ YL,\ VP,\ t,\ k,\ b_1,\ b_2,\ldots,\ b\rho-1) \\
G \longleftarrow -A_1\,(Y,YL,\ VP,\ t,\ k,\ b_1,\ b_2,\ldots b\rho).VP \\
\qquad +A_2\,(Y,\ YL,\ VP,\ t,\ k,\ b_1,\ b_2,\ldots b\rho).Y \\
\qquad +A_3\,(Y,\ YL,\ VP,\ t,\ k,\ b_1,\ b_2,\ldots b\rho).E \\
\qquad +A_4\ .\ YL + A_5\ .\ J \\
F_1 \longleftarrow -A_6\,(Y,\ YL,\ VP,\ t,\ k,\ b_1,\ b_2,\ldots,\ b\rho)\ \dfrac{dY}{dt} \\
\qquad +A_8\,(Y,\ YL,\ VP,\ t,\ k,\ b_1,\ b_2,\ldots b\rho)\ .\ Y \\
\qquad +A_9\,(Y,\ YL,\ VP,\ t,\ k,\ b_1,\ b_2,\ldots b\rho)\ .\ E \\
\qquad +A_{10}\ .\ YL + A_{11}\ .\ J \\
\qquad +A_7\,(Y,\ YL,\ VP,\ t,\ k,\ b_1,\ldots B\rho)\ .\ VP \\
F_2 \longleftarrow -A_{15}\,(Y,\ YL,\ VP,\ t,\ k,\ b_1,\ldots b\rho)\ .\ \dfrac{dYL}{dt} \\
\qquad +A_{16}\ .\ Y + A_{17}\ .\ E + A_{18}\ .VP
\end{array}\right\} \qquad \text{(III.11)}
$$

où les A_i sont les matrices figurant dans (III.8), k note un ensemble de constantes du circuit et t le temps. Les $b\rho$ sont des quantités intermédiaires pouvant apparaître plusieurs fois dans les A_i et permettant d'éviter une redondance dans les calculs. La plupart d'entre elles correspondent au calcul de la valeur de composants non-linéaires. Dans ce cas,

la redondance s'explique par l'utilisation de ces valeurs dans plusieurs équations de fonctionnement.

Le dérivateur formel déjà utilisé dans IMAG 2 (ECLAT) nous permet de déterminer la différentielle totale de F et G sous la forme :

(III.12)

$$
\begin{aligned}
db_1 \longleftarrow\ & \frac{\partial g1}{\partial Y}\,dY + \frac{\partial g1}{\partial YL}\,dYL + \frac{\partial g1}{\partial VP}\,dVP \\
db_2 \longleftarrow\ & \frac{\partial g2}{\partial Y}\,dY + \frac{\partial g2}{\partial YL}\,dYL + \frac{\partial g2}{\partial VP}\,dVP + \frac{\partial g2}{\partial b_1}\,db_1 \\
\vdots & \\
db \longleftarrow\ & \frac{\partial g\rho}{\partial Y}\,dY + \frac{\partial g\rho}{\partial YL}\,dYL + \frac{\partial g\rho}{\partial VP}\,dVP + \sum_{i=1}^{\rho-1} \frac{\partial g2}{\partial b_i}\,db_i \\
dG \longleftarrow\ & -\frac{\partial A1}{\partial Y}\,dY + \frac{\partial A1}{\partial YL}\,dYL + \frac{\partial A1}{\partial VP}\,dVP + \sum_{i=1}^{\rho} \frac{\partial A1}{\partial b_i}\,db_i + \ldots \\
& \cdots\cdots\cdots \\
& + \frac{\partial A3}{\partial Y}\,dY + \frac{\partial A3}{\partial YL}\,dYL + \frac{\partial A3}{\partial VP}\,dVP + \sum_{i=1}^{\rho} \frac{\partial A3}{\partial b_i}\,db_i \\
& + A_4 \cdot dYL + A_5 \cdot dJ \\
dF_1 \longleftarrow\ & -\frac{\partial A6}{\partial Y} \cdot \frac{dY}{dt} \cdot dY - \frac{\partial A6}{\partial YL}\,\frac{dYL}{dt}\,dYL - \frac{\partial A6}{\partial VP}\,\frac{dY}{dt}\,dVP - A_6 \cdot d\left(\frac{dY}{dt}\right) \\
& - \sum_{i=1}^{\rho} \frac{\partial A6}{\partial b_i}\,\frac{dY}{dt}\,db_i + \\
& \cdots\cdots\cdots \\
& + A_{10}\,dYL + A_{11} \cdot dJ + \\
& + \frac{\partial A7}{\partial Y}\,d1 + \frac{\partial A7}{\partial YL}\,dYL + \ldots + \sum_{i=1}^{k} \frac{\partial A7}{\partial b_i}\,db_i \\
dF_2 \longleftarrow\ & -\frac{\partial A15}{\partial Y} \cdot \frac{dYL}{dt}\,dY - \ldots - \frac{\partial A15}{\partial VP}\,\frac{dYL}{dt}\,dVP \\
& - \sum_{i=1}^{\rho} \frac{\partial A15}{\partial b_i}\,\frac{dYL}{dt}\,db_i \\
& + A_{16} \cdot dY + A_{17} \cdot dE + A_{18} \cdot dVP \,.
\end{aligned}
$$

On calcule également dE et dJ sous la forme :

$$dE = \frac{\partial E}{\partial Y} dY + \frac{\partial E}{\partial YL} dYL + \frac{\partial E}{\partial VP} dVP$$

(III.13)

$$dJ = \frac{\partial J}{\partial Y} dY + \frac{\partial J}{\partial YL} dYL + \frac{\partial E}{\partial VP} dVP$$

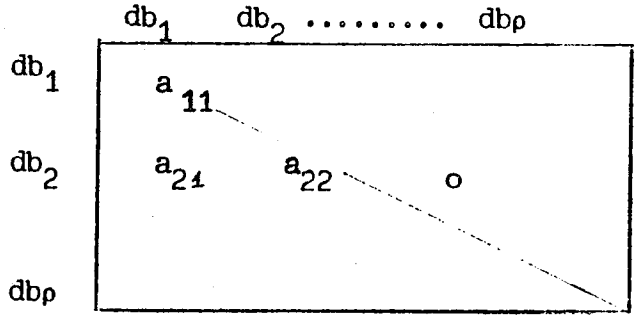
La première phase consiste à exprimer les parties droites de G, dF_1 et dF_2 en fonction de Y, YL, VP, k, dY, dYL, dVP, $\frac{dY}{dt}$, $\frac{dYL}{dt}$

Ceci s'effectue par substitution de dE, dJ et des db_i.

Pour minimiser l'espace mémoire nécessaire, la substitution s'effectue de bas en haut, c'est-à-dire à partir des db_i figurant dans dG, dF_1 ou dF_2. On remonte pour chaque db_i sa chaîne de dépendance jusqu'à l'obtention d'une expression ne contenant plus que Y, YL, VP, k, dY, dYL, dVP.

A l'intérieur de chaque expression, la substitution s'effectue de gauche à droite.

Pour optimiser cette phase du traitement, on utilise la matrice suivante :



ou a_{ij} = o si db_i ne dépend pas de db_j

= adresse de db_j sinon

Cette matrice est triangulaire inférieure et sa diagonale est nulle.

On pose :

$a_{ii} = o$ si db_i dépend de db_j , $j < i$

$= 1$ sinon.

L'utilisation de cette matrice nous permet de nous déplacer rapidement dans les chaînes de dépendance, elle est construite par une lecture séquentielle des db_i.

Chaque expression ainsi obtenue pour dG, dF_1 et dF_2 se présente sous la forme d'une somme de produit de 2 termes :

- une fonction M (Y, YL, VP, t, k, $\frac{dY}{dt}$, $\frac{dYL}{dt}$)

- un terme : dY, dYL, dVP, d ($\frac{dY}{dt}$), d ($\frac{dYL}{dt}$) que l'on note $d\phi$

L'expression est ensuite mise en facteur par rapport a $d\phi$, pour obtenir :

$$PD = \begin{Bmatrix} dG \\ dF_1 \\ dF_2 \end{Bmatrix} \leftarrow \sum_{i=1}^{n} \varphi_i \, d\phi_i$$

De cette manière $\varphi = \frac{\partial (F1, F2, G)}{\partial \phi}$

Enfin, $\frac{\partial (F1, F2)}{\partial (Y, YL)}$ et $\frac{\partial (F1, F2)}{\partial (\frac{dY}{dt}, \frac{dY}{dt})}$ sont regroupés pour chaque i sous la forme (1)

$$\frac{\partial F_1^i}{\partial Y_i} - \alpha \frac{\partial F_1^i}{\partial \left(\frac{dY_i}{dt}\right)} \quad \text{et} \quad \frac{\partial F_2^i}{\partial YL_i} - \alpha \frac{\partial F_2^i}{\partial \left(\frac{dYL_i}{dt}\right)}$$

(1) - La raison de ce regroupement tient aux algorithmes numériques utilisées dans la phase d'analyse.(Voir chapitre II).

On obtient finalement la matrice suivante :

$$J = \begin{bmatrix} \frac{\partial G}{\partial VP} & \frac{\partial G}{\partial Y} & \frac{\partial G}{\partial YL} \\ \frac{\partial F1}{\partial VP} & \frac{\partial F1}{\partial Y} - \alpha \frac{\partial F1}{\partial (\frac{dY}{dt})} & \frac{\partial F1}{\partial YL} \\ \frac{\partial F2}{\partial VP} & \frac{\partial F2}{\partial Y} & \frac{\partial F2}{\partial YL} - \alpha \frac{\partial F2}{\partial (\frac{dYL}{dt})} \end{bmatrix} \qquad \text{(III.14)}$$

où chaque terme non nul est donné par une expression post-fixée.

Cet ensemble d'expressions, ainsi que les expressions (III.11), sont traduites en code machine exécutable pour permettre une évaluation rapide au cours de la phase d'analyse. On utilise pour cela le générateur de code développé par les auteurs d'IMAG 2.

Signalons que R. MIGNONE a écrit un programme permettant de simplifier les expressions des termes de J évitant ainsi un certain nombre de redondances provenant des substitutions et qui a été incorporé dans le système IMAG 3.

La mise en oeuvre de l'algorithme est fortement liée à la forme particulière des équations (III.12). Il présente donc l'inconvénient de ne pouvoir être utilisé hors de son contexte, mais cette adaptation nous a permis de rendre minimum l'espace mémoire nécessaire - qui est souvent important dans les algorithmes généraux - et le coût de cette phase de traitement.

Une estimation quantitative de ce coût est difficile à établir. Il dépend bien entendu de la dimension du circuit, mais surtout de la complexité des expressions donnant la valeur des éléments non-linéaires et de leur position dans le circuit.

Il ne semble pas dépasser 3 à 4 s (360-67 OS/MVT) pour des exemples comportant 200 à 250 variables (Y, YL, VP) et de complexité "moyenne" (1), ce qui est très inférieur au coût d'un programme général.

L'utilisation des équations de fonctionnement (III.7) nous a permis d'exhiber un nouvel invariant du circuit que l'on notera I_J. Cet invariant se déduisant de l'invariant fondamental I_F, nous l'appellerons invariant auxilliaire.

On peut remarquer que I_J - comme I_F - est indépendant des modes d'analyse, ce qui le distingue de l'invariant que nous allons examiner dans la suite.

III.2.3. Résolution du système linéaire J.X = B.

Poursuivant notre démarche, nous recherchons un invariant I_1 permettant de résoudre à moindre coût le système linéaire :

$$J.X = B \qquad \text{(III.15)}$$

au cours de la phase d'analyse.

La résolution de (III.15) par des méthodes classiques est extrêmement coûteuse. Dans les méthodes directes, ce coût est proportionnel au cube de la dimension du système.

Les méthodes itératives (Gauss - Seidel, etc...) se révèlent - sur les tests que nous avons effectués - encore plus coûteuses que les précédentes, même en utilisant des techniques d'accélération de la convergence.

L'emploi de ces méthodes classiques interdit pratiquement d'aborder de façon efficace des circuits comportant plus d'une vingtaine de transistors, ce qui reste très loin des circuits intégrés que l'on veut maintenant réaliser.

(1) Un circuit de complexité "moyenne" est un circuit comprenant des transistors bipolaires décrits par le modèle d'Ebers et Moll,

Le système d'équations (III.7) étant creux, le jacobien J l'est également. Cette propriété va nous permettre de réduire le coût de résolution de (III.15) de façon considérable.

III.2.3.1. Propriétés de (III.15). Approche de résolution

Le taux d'éléments non nuls de la matrice J est une fonction du circuit. D'une facon générale, plus le circuit est volumineux, plus ce taux est faible, (pour un même type de circuit).

Si l'on exclue le cas des petits circuits - c'est-à-dire jusqu'à une dizaine d'équations (2 à 3 transistors) - pour lesquels les problèmes de coût ne se posent pas, on peut considérer que le taux d'éléments non nuls de J varie entre 5 et 25 %.

Une connaissance approximative de ce taux est importante, car comme nous le verrons, elle nous guidera dans le choix d'un mode de stockage en machine des termes de J.

Parmi ces termes non nuls, certains sont constants au cours de la phase d'analyse, d'autres dépendent du temps seul, d'autres enfin sont fonction des variables : y, VP, $\frac{dy}{dt}$.

Comme nous le verrons en détail dans le chapitre II, les modes d'analyse continu, transitoire et alternatif exigent un grand nombre de résolutions de (III.15). En effet, les algorithmes utilisés étant itératifs, et chaque itération utilisant une linéarisation de (III.7), le système linéaire (III.15) devra être résolu à chacune d'elle.

De l'ordre de 20 fois pour le mode continu et alternatif, 100 à 200 fois pour le mode transitoire.

Les valeurs ci-dessus sont des valeurs moyennes, mais peuvent augmenter considérablement dans certains cas particuliers difficiles.

La résolution n diffère des résolutions n+1 et n-1 par deux facteurs :

- le second membre B peut être différent.
- les valeurs prises par certains termes non nuls de J sont différentes.

Par contre, la position des termes nuls et non nuls reste identique.

$J_{ij} = 0$ correspond à l'absence de la variable y_j dans l'équation n°:i et est donc globalement nul dans tout le domaine de variation des variables. Bien entendu, certains termes non nuls peuvent être nuls pour des points particuliers de ce domaine.

Quel que soit l'algorithme utilisé pour résoudre (III.15), il se décomposera finalement en une suite d'opérations I_{l_o} portant sur les éléments J_{ij} de J, X_i de X et B_i de B.

Certaines de ces opérations - compte-tenu de la nature de la matrice J - sont triviales (multiplication par un terme nul, addition d'un terme nul, etc...), donc peuvent être supprimées dans la suite I_{l_o} pour former une suite I_l.

Cette suite I_l dépendra de deux facteurs :

(a) - position des termes non nuls de J.

(b) - ordre des opérations déterminé par l'algorithme.

Pour un circuit donné, (a) est un invariant. Dans les méthodes directes basées sur l'élimination de Gauss (1), l'ordre des opérations dépend de (a), ainsi que de contraintes numériques - fonction de la valeur des termes de J - permettant d'assurer que la solution obtenue pour (III.15) est numériquement correcte.

Soit D_1 un domaine de variation de (y, VP) tel que dans ce domaine, les contraintes numériques ci-dessus conduisent à la même séquence I_1. Notons $I_{1_{D1}}$ cette séquence.

Alors $I_{1_{D1}}$ est un invariant du circuit relativement au domaine D_1.

Une approche intéressante pour résoudre (III.15) peut donc être la suivante :

(1) Déterminer la séquence $I_{1_{D_o}}$ où D_o est un domaine contenant les valeurs initiales (y_o, VP_o). Cette séquence peut être générée sous la forme d'une suite d'expressions symboliques.

(2) Pour chaque résolution de (III.15) dans D_o, effectuer cette séquence d'opérations.

(3) Pour chaque changement de domaine D_i, déterminer la séquence $I_{1_{D_i}}$ correspondante.

Cette méthode présente les avantages suivants :

- la détermination de la séquence $I_{1_{Di}}$ ne s'effectue qu'une seule fois pour chaque D_i d'où un gain de temps appréciable.

(1) Klyuyev et Kokovkin-Shcherbak (Klyu 1965) ont montré qu'un système linéaire quelconque ne peut être résolu en un nombre d'opérations inférieur à celui requis par l'élimination de Gauss.

- Pour chaque résolution de (III.15), on effectue un nombre minimum d'opérations.

Bien entendu, il est nécessaire que les changements de domaines D_i ne soient pas trop fréquents et d'autre part, que l'on soit capable de les détecter car ils ne sont pas connus a priori.

Le premier point ne peut qu'être constaté expérimentalement. Dans la plupart des cas, le seul domaine D_o permet de couvrir toute une phase d'analyse. Dans certains cas, on peut être amené à effectuer 2 à 3 déterminations, mais rarement plus. Une explication de ce fait sera donnée dans le chapitre II.

Nous examinerons le problème des changements de domaine D_i dans le paragraphe suivant.

La séquence $I_{1_{D_i}}$ dépendant également de la position des termes non nuls de J, on peut envisager, pour déterminer la séquence optimale (contenant le minimum d'opérations) de transformer le système (III.15) en un système équivalent (par exemple par permutation de lignes et de colonnes). Ce problème sera envisagé dans le paragraphe suivant où nous exposons l'algorithme utilisé pour déterminer les séquences $I_{1_{D_i}}$.

Remarque : la méthode ci-dessus ne peut s'appliquer dans le cas d'IMAG 2. Les inversions de matrices nécessaires pour déterminer la forme canonique empêchent de connaître la position des termes nuls de J, les substitutions VP(y) dans f s'effectuant numériquement.

III.2.3.2. Algorithme de détermination de $I_{1_{D_i}}$.

Un algorithme permettant de déterminer $I_{1_{D_i}}$ doit posséder les deux caractéristiques essentielles suivantes :

(a) - minimiser le nombre d'opérations de $I_{1_{Di}}$ pour un système (III.15) donné.

(b) - conduire à une solution précise même dans le cas où le système est mal conditionné (ce qui arrive fréquemment dans les problèmes issus des circuits électroniques).

D'autre part, dans certains cas, les valeurs numériques des termes de J ne sont pas modifiées entre plusieurs résolutions successives. Il est donc intéressant de disposer d'un algorithme permettant de générer une séquence dont une partie soit indépendante du second membre B.

Ces considérations nous ont conduit à choisir un algorithme basé sur l'élimination de Gauss et utilisant la décomposition de J en 2 matrices L et U, triangulaires inférieure et supérieure.

Considérons la matrice :

$$[J\ B] = \begin{bmatrix} J_{11} & \cdots & J_{1n} & b \\ \vdots & & \vdots & \vdots \\ J_{n1} & \cdots & J_{nn} & b_n \end{bmatrix} \qquad \text{(III.16)}$$

Le premier pas consiste à normer la première ligne par J_{11} :

$$J^{(1)}_{ij} = (1/J_{11}) * J_{1j}$$

$$b^{(1)}_{1} = (1/J_{11}) * b_1$$

Dans le deuxième pas, on élimine J_{21} par combinaison linéaire des lignes 1 et 2, puis on norme la deuxième ligne par J_{22} :

$$J^{(1)}_{2j} = J_{2j} - J_{21} * J^{(1)}_{1j} \qquad j = 2,n$$

$$b^{(1)}_{2} = b_2 - J_{21}\, b^{(1)}_{1}$$

$$J_{2j}^{(2)} = (1/J_{22}) * J_{2j}^{(1)} \qquad j = 3,n$$

$$b_2^{(2)} = (1/J_{22}^{(1)}) * b_2^{(1)}$$

Au troisième pas, on élimine les éléments de la troisième ligne à gauche de la diagonale, puis on norme les éléments restants par $J_{33}^{(2)}$:

$$J_{3j}^{(1)} = J_{3j} - J_{31} * J_{1j}^{(1)} \qquad j = 2,n$$

$$b_3^{(1)} = b_3 - J_{31}\, b_1^{(1)}$$

$$J_{3j}^{(2)} = J_{3j}^{(1)} - J_{32}^{(1)} * J_{2j}^{(2)} \qquad j = 3,n$$

$$b_3^{(2)} = b_3^{(1)} - J_{32}^{(1)}\, b_2^{(2)}$$

$$J_{3j}^{(2)} = (1/J_{33}^{(2)}) * J_{3j}^{(2)} \qquad j = 4,n$$

$$b_3^{(3)} = (1/J_{33}^{(2)}) * b_3^{(2)}$$

De cette manière, après n pas, on obtient :

$$\begin{pmatrix} 1 & J_{12}^{(1)} & \cdots & J_{1n}^{(1)} & b_1^{(1)} \\ & 1 & & J_{2n}^{(2)} & b_2^{(2)} \\ & & & & \\ & & & 1 & b_n^{(n)} \end{pmatrix}$$

La solution peut alors être obtenue par substitution :

$$x_n = b_n^{(n)}$$

$$x_{n-1} = b_{n-1}^{(n-1)} - J_{n-1,n}^{(n-1)} * x_n$$

$$x_i = b_i^{(i)} - \sum_{f=i+1}^{n} J_{ij}^{(i)} * x_j$$

Remarque : cette élimination est équivalente à celle de Gauss. Toutefois, dans cette dernière, elle s'effectue généralement par colonnes.

Pour rendre la triangulation indépendante du second membre, il faut enregistrer les opérations effectuées sur les b_i.

Pour reconstituer ces opérations, il suffit - sur le troisième pas - de stocker J_{31}, J_{32} et $1/J_{33}$. On voit aisément que ceci se généralise pour un pas i quelconque.

On adopte les deux règles suivantes :

- les termes $1/J_{ii}$ (i-1) sont stockés sur la diagonale

- les termes $a_{ij}^{(j-1)}$, i>j sont stockés en position (i,j) : matrice triangulaire inférieure.

On obtient finalement :

$$\begin{pmatrix} d_{11} & u_{12} & \cdots & u_{1n} \\ l_{21} & d_{22} & & u_{2n} \\ l_{31} & & & \vdots \\ \vdots & & & \\ l_{1} & \cdots & & d_{nn} \end{pmatrix}$$

La matrice ci-dessus n'est pas une matrice au sens de J, mais plutôt un enregistrement d'opérations.

Le calcul de X s'effectuera en deux étapes à partir de la matrice ci-dessus :

(1) - Calcul de $b_1^{(1)}$ ----- $b_n^{(n)}$ par substitution sur le système.

$$\begin{pmatrix} d_{11} & & \\ l_{21} & \ddots & \\ l_{n1} & \cdots & d_{nn} \end{pmatrix} \begin{pmatrix} b_1^{(1)} \\ \\ b_n^{(n)} \end{pmatrix} = B$$

(2) - Calcul de X par substitution sur

$$\begin{bmatrix} 1 & U_{12} & \cdots & U_{1n} \\ & \ddots & & \vdots \\ & & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ \vdots \\ b_n^{(n)} \end{bmatrix}$$

Cet algorithme a été employé sous cette forme par Tinney et Walker [TIN 67] dans des problèmes de réseaux de transport. On peut remarquer que les problèmes (a) et (b) n'ont été que partiellement pris en compte. Nous nous y attachons maintenant. Les problèmes de précision et d'erreurs d'arrondies rencontrés dans la résolution de systèmes linéaires par des méthodes directes sont des problèmes difficiles. Ils ont été largement explorés par Wilkinson [Wilk 65].

Rappelons seulement ici qu'on peut envisager deux stratégies pour limiter ces erreurs :

- le pivotage partiel
- le pivotage total.

Dans l'algorithme ci-dessus, chaque terme à droite de la diagonale dans une ligne i est normé par $J_{ij}^{(i-1)}$. Ce terme est appelé pivot relatif à la ligne i. On compte au total n pivots (un par ligne). Le choix de ces pivots est important pour la précision de la solution X obtenue. D'une manière générale, on a intérêt à choisir comme pivot un terme ayant une valeur absolue maximum, ceci à chaque pas de la décomposition.

Le pivotage partiel consiste à choisir au pas i, le terme de plus grande valeur absolue dans la ligne i.

Le pivotage total consiste à choisir ce terme dans la sous-matrice $J^{(i)}$ de J correspondant aux lignes et colonnes non encore traitées. (On doit alors effectuer une permutation de lignes et de colonnes sur $J^{(i)}$.
Bien que cette méthode conduise aux meilleurs résultats, elle est relativement coûteuse et ne permet pas de satisfaire facilement au critère (a) puisque l'ordre des éliminations se trouve entièrement déterminé.

D'autres raisons que nous verrons dans le chapitre II nous ont conduit à adopter un pivotage partiel. Au cours de la décomposition de la matrice J, il est bien évident que certains termes nuls dans J peuvent cesser de l'être dans les matrices L et U.

Exemple : considérons l'opération :

$$J_{3j}^{(1)} = J_{3j} - J_{31} * J_{1j}^{(1)}$$

supposons $J_{3j} = 0$ mais $\begin{cases} J_{31} \neq 0 \\ J_{1j}^{(1)} \neq 0 \end{cases}$

dans ce cas, $J_{3j}^{(1)} \neq 0$

Dans la littérature, l'ensemble des termes ainsi générés s'appelle le "fill-in".

La génération de ces termes présente deux inconvénients :

- elle exige de l'espace mémoire pour leur stockage
- elle augmente le nombre d'opérations nécessaires à la résolution du système.

Le "fill-in" dépend de la position des termes non nuls dans J, ainsi que de l'ordre dans lequel sont conduites les éliminations, c'est-à-dire finalement des pivots choisis à chaque étape de la factorisation.

Ainsi, le problème peut se formuler de la manière suivante :

- choisir un ensemble de pivots p permettant :
  - . une précision numérique suffisante
  - . minimisant le nombre de termes générés.

Ces deux exigences sont souvent contradictoires et les algorithmes permettant de minimiser le "fill-in" qu'on peut trouver dans G.B. Dantzig et al [Dantz. 69] ne tiennent pas compte des critères numériques.

Aussi, nous avons constaté qu'ils devaient être rejetés.

Hachtel, Brayton et Gustavson Hacht. 71] proposent un algorithme intéressant permettant de tenir compte des deux objectifs. Ils y introduisent la notion de variabilité des termes de la matrice.

Ces termes sont répartis en trois classes :

- termes constants (variabilité 1)
- termes fonctions de t seul (variabilité 2)
- termes fonctions des variables (variabilité 3).

Les opérations de la factorisation L/U ne faisant intervenir que des éléments du type 1, pourront être effectuées une seule fois pour chaque analyse, celles qui font intervenir les éléments du type 2 au plus, à chaque variation de t, etc...

A chaque multiplication ou division de la factorisation, on associe sa variabilité :

$$VTM = \text{Max}\{VT(i, j), VT(k, l)\}$$

où $VT(i, j)$ désigne la variabilité de J_{ij}.

VTM indique quand l'opération doit être effectuée.

Au pas k de la factorisation, tous les termes non nuls de $J^{(k)}$ sont candidats pour devenir le $k^{ième}$ pivot.

Pour chaque $J_{i,j}^{(k)}$, on calcule un indice de performance :

$$IP\ (i,j) = \mu . M\ (i,j) + \gamma . R\ (i,j).$$

M est une fonction du nombre d'opérations requises, ainsi que de leur variabilité, R un facteur de précision numérique, γ et μ des paramètres que l'on peut ajuster.

$$R\ (i,j) = \frac{1}{r_i} \sum_{l} \frac{/J_{i,l}^{(k)}/}{/J_{ij}^{(k)}/}$$

où r_i est le nombre d'éléments non nuls sur la ligne i et l parcourt les termes non nuls de cette ligne.

$$M\ (i,j) = \sum_{ope} \lambda_{VIM} \cdot VIM_{ope}$$

où ope parcourt l'ensemble des multiplications et divisions nécessaires au pas k si $J_{i,j}^{(k)}$ est choisi comme pivot.

λ_{VIM} est un poids associé à chaque classe.

La valeur des paramètres λ, μ et γ est extrêmement critique et très lié à la nature du problème.

Nous avons utilisé cette méthode dans IMAG 3. Cependant, elle se révèle extrêmement coûteuse pour le choix des pivots, ce qui exclue pratiquement un nouveau choix au cours de la phase d'analyse. D'autre part, elle ne semble utilisable que pour des matrices très creuses (<5 %) et avec très peu d'éléments variables.

Sinon, le nombre d'opérations de variabilité maximum croît rapidement, ce qui fait perdre tout les avantages de cet algorithme. D'autre part, il semble que l'on se heurte à des problèmes de précision numérique.

Aussi, nous avons abandonné cette méthode au profit d'une autre beaucoup plus simple et peu coûteuse.

Remarquons qu'une ligne intervient d'autant plus dans la factorisation L/U qu'elle a un indice faible.

D'où l'idée de ranger les lignes par nombre d'éléments non nuls croissant. On effectue ensuite un pivotage partiel sur chaque ligne, ce qui conduit à des permutations de colonnes.

Au cours du chapitre III, nous verrons la différence de coût des deux méthodes précédentes. Celle de Hachtel et al étant utilisée dans le système ASTAP [Weeks 73].

Le problème de changement de domaine D_i se ramène à déterminer si un choix de pivots effectués pour des valeurs $(y_i, \frac{dyi}{dt}, VP_i, t_i)$ reste ou non correct.

Cette notion dépend évidemment du circuit et de ses équations de fonctionnement, mais cette liaison semble extrêmement difficile à faire et exigerait des connaissances a priori sur le fonctionnement du circuit que l'on ne possède pas en général.

Il faut donc se contenter de vérifier que dans la décomposition,

$J + E = LU$ où LU est la décomposition numériquement exacte de $J + E$ obtenue à partir de J, E est voisine de la matrice nulle d'une manière fixée a priori.

Cette vérification devant être extrêmement rapide car elle est effectuée à chaque factorisation.

Nous utilisons le critère $\left|\frac{J_{i,j}}{J_{ii}}\right| < k$

avec $k > 0$ à fixer pour chaque cas.

Jusqu'ici, ce critère semble donner satisfaction.

III.2.3.3. Mise en oeuvre programmée

Les deux problèmes essentiels de cette mise en oeuvre sont les suivants :

(a)- choix d'une structure de donnée permettant de stocker J et sa décomposition.

(b)- traitement de la suite d'opérations formelles permettant de résoudre le système linéaire et générée par l'algorithme du paragraphe précédent.

Dans les problèmes de matrices creuses, on s'attache à minimiser l'espace mémoire occupé en ne stockant que les termes non nuls.
Ce gain est souvent réalisé au détriment de la rapidité de traitement.
En effet, les structures de données mises en jeu sont des listes et les éléments ne peuvent être retrouvés que grâce à des balayages.

Des auteurs se sont employés à proposer des structures de listes efficaces et adaptées au traitement dela factorisation L/U. On peut trouver un panorama de ces recherches dans Tewarson [Tew. 73].
Nous avons tenté d'utiliser ces schémas, en particulier le chaînage orthogonal tel qu'on peut le voir dans Knuth [Knuth 68].

Cependant, ces schémas sont intéressants pour des matrices extrêmement creuses (< 5 %). Pour des taux de remplissage supérieurs, leur emploi devient discutable à la fois au point de vue de l'espace mémoire sauvegardé et surtout par l'augmentation des coûts qu'ils introduisent.

Aussi, nous sommes revenus pour cette réalisation, à un stockage de J sous la forme de tableau carré, axant ainsi notre démarche sur la rapidité de traitement.

Pour le point (b), nous nous trouvions devant l'alternative suivante :

- générer la suite d'opérations en code machine exécutable en utilisant le générateur de code développé par les auteurs d'IMAG 2.

- construire un interpréteur écrit en FORTRAN et permettant d'interpréter la suite d'opérations de résolution.

Bien entendu, la première méthode est la plus efficace. Dans la seconde, subsistent en effet des opérations de décodage de la chaîne des expressions.

Le rapport d'efficacité entre les deux approches dépend de la machine utilisée.

Sur le calculateur IBM 360-67 utilisé à Grenoble, nous avons constaté un rapport de l'ordre de 4.

Cependant, dans la première solution, on doit utiliser des macros systèmes pour récupérer les interruptions (débordement, division par 0, etc...) et contrôler le déroulement numérique de la résolution.
Ceci entraîne une dépendance importante vis-à-vis de la machine hôte qui n'était pas souhaitable dans le cas d'IMAG 3.

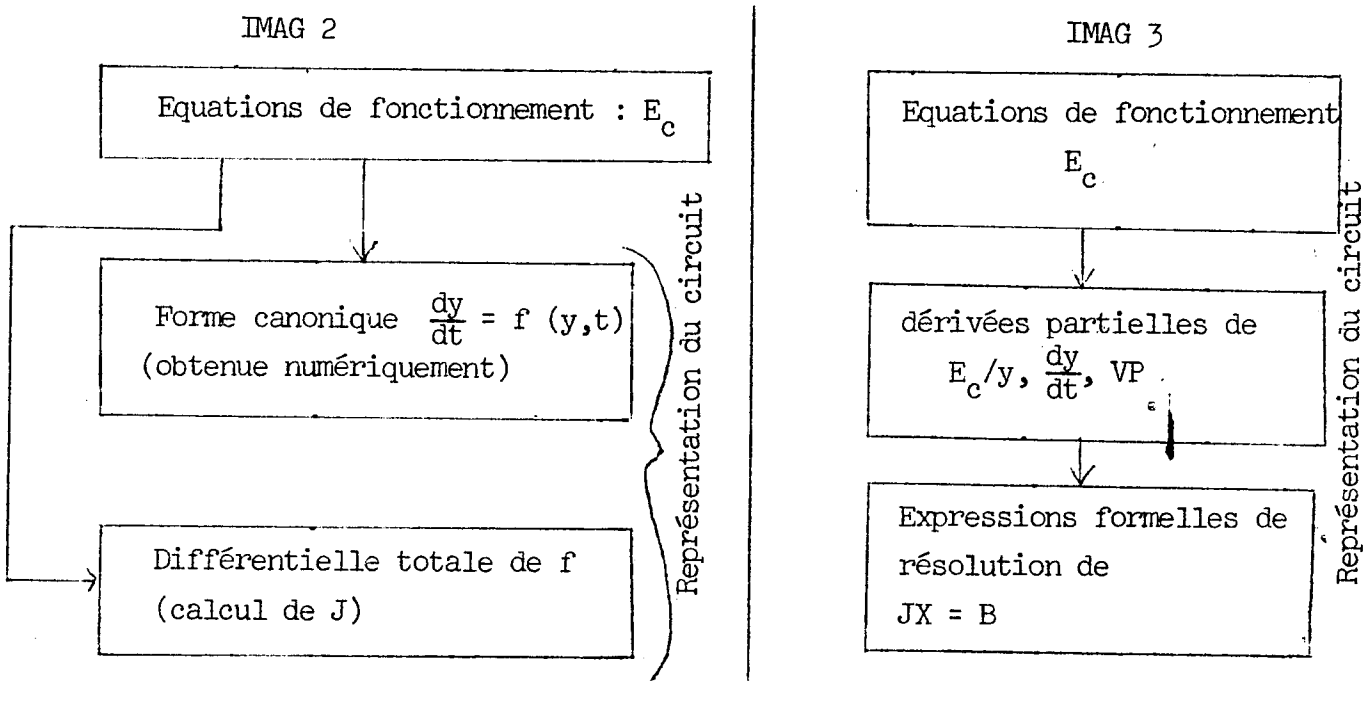
Aussi, nous avons construit un interpréteur permettant d'évaluer cette suite d'expressions.

Le principe en est simple et nous n'y insisterons pas ici.

Finalement, grâce à la méthode décrite ci-dessus, nous sommes parvenus à réduire le coût de la résolution de (III.15) de manière importante. Ce coût est maintenant proportionnel à la dimension du système linéaire, alors que dans IMAG 2, il était une fonction du cube de cette dimension. Ceci nous permet d'envisager le traitement de circuits beaucoup plus volumineux.

IV Conclusion

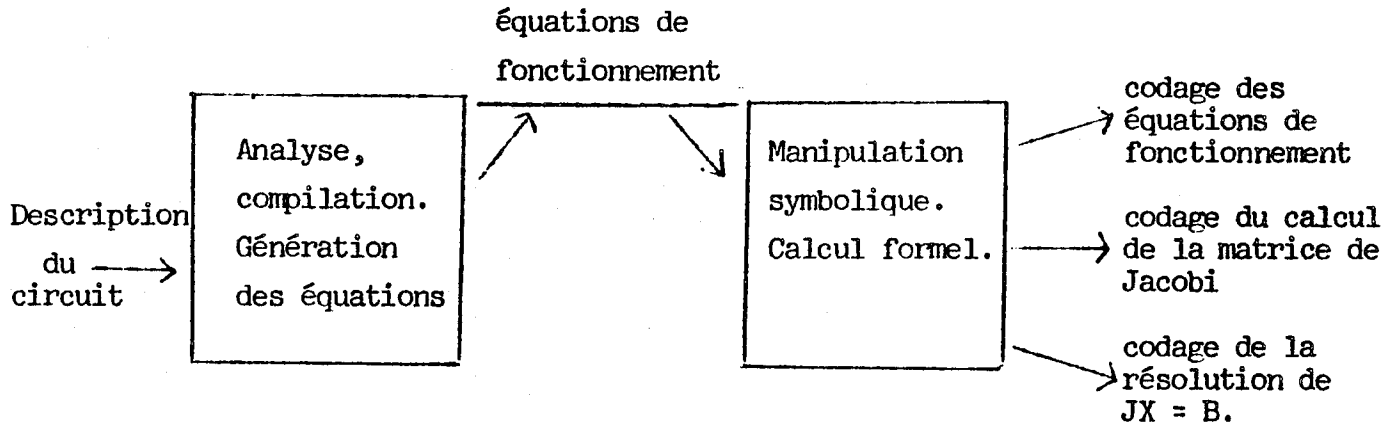
Nous avons dans ce chapitre exposé les points communs entre les approches adoptées dans IMAG 2 et IMAG 3 ainsi que leurs divergences dans la phase de traduction. Celles-ci peuvent se schématiser de la manière suivante :



On peut voir sur ce schéma que la différence essentielle réside dans les invariants du circuit que l'on a pu obtenir grâce à l'utilisation directe des équations E_c dans la phase d'analyse. Il est bien évident que cette transformation a été possible parce que nous pouvions envisager d'utiliser dans la phase d'analyse des algorithmes adéquats.

D'un point de vue pratique, la réalisation de cette phase de traduction peut se décomposer en deux étapes :

- analyse et compilation du langage de description, génération des équations de fonctionnement.
- manipulation symbolique et génération des invariants.



La liaison entre ces deux étapes s'effectue par une structure de donnée constituée par un codage des équations de fonctionnement en écriture post-fixée. Ces deux étapes sont donc relativement indépendantes l'une de l'autre. Cette indépendance permet d'envisager l'utilisation de la deuxième étape (ainsi que les algorithmes d'analyse) à l'étude de problèmes autres que les circuits électroniques, mais donc, les équations de fonctionnement peuvent se ramener à des schémas algébro-différentiels.

Finalement, les avantages essentiels de l'approche adoptée dans IMAG 3 sont les suivants :

- formalisme plus général permettant l'étude de circuits plus généraux.
- les équations de fonctionnement adoptées sont plus creuses que

la forme canonique, d'où une simplicité de calcul plus grande et une matrice de Jacobi plus creuse.

- calcul formel des termes de la matrice de Jacobi (impossible sur IMAG 2), d'où calcul plus rapide de cette matrice (rapport N environ si N est la dimension de la matrice) et possibilité d'adopter un algorithme efficace de résolution de JX = B.
- résolution de JX = B en un nombre d'opérations minimum et par interprétation d'un code symbolisant ces opérations, d'où la suppression des tests de recherche de l'ordre des éliminations, ce qui conduit à un coût de résolution proportionnel à N au lieu de N^3.

Les algorithmes d'analyse utilisés dans IMAG 3 sont différents de ceux d'IMAG 2 et choisis de manière à utiliser les résultats de cette approche. Dans le chapitre suivant, nous abordons l'étude de ces algorithmes.

Les gains effectifs apportés par cette approche ne pourront être précisés qu'après cette étude, puisque, comme nous l'avons déjà indiqué, les phases de traduction et d'analyse sont liées dans le sens où la première prépare des données pour la seconde.

CHAPITRE II

Nous avons vu dans le chapitre précédent comment sont obtenues les équations décrivant le circuit ainsi que certaines de leurs propriétés qui nous ont permis de constituer un ensemble d'invariants destiné à faciliter la phase d'analyse.

Le choix de ces invariants a été guidé par les algorithmes que nous savions pouvoir utiliser dans l'analyse, aussi, ce choix et le développement des algorithmes ont-ils été conduits simultanément. La simulation proprement dite, c'est-à-dire l'interprétation par la machine abstraite M des commandes de simulation, consiste à calculer la solution des équations de fonctionnement E_c pour leur forme particulière mise en jeu pour chacun des trois régimes d'analyse possibles.

La résolution d'un ensemble d'équations peut s'envisager de deux manières distinctes :

- par des méthodes exactes qui permettent d'obtenir des solutions formelles.
- par des méthodes numériques dans lesquelles on recherche des valeurs numériques approchées des solutions.

Dans la pratique du calcul par ordinateur, seules les méthodes numériques sont actuellement utilisables dans le cas de systèmes d'équations complexes.

G. FORSYTHE définit ainsi les compétences nécessaires à la résolution d'un problème réel :
"The successful solution of a realistic problem in applied mathematics requires the fusion of four distinct ingredients : (1) knowledge of the subject area of the problem ; (2) knowledge of the relevant mathematics ; (3) knowledge of the relevant computer science ; (4) a talent for selecting just what part of all this knowledge will actually solve the problem, and ignoring the rest".

La réunion de ces compétences est suffisamment rare pour expliquer que l'emploi des méthodes numériques dans l'étude des circuits électroniques (ainsi que dans d'autres domaines) reste un problème majeur.

Toutefois, un certain nombre de progrès ont été réalisés et nous ont permis d'aboutir au système IMAG3. Les contraintes essentielles sur les algorithmes d'analyse sont :

- le coût
- la fiabilité.

Pour des raisons évidentes d'utilisation, les coûts doivent être minimaux pour permettre d'aborder raisonnablement des circuits réels qui - avec l'apparition des circuits intégrés - sont de plus en plus complexes. D'autre part, l'utilisateur attend une solution correcte pour son problème. Aussi, les algorithmes numériques doivent-ils être fiables. Ce problème de la fiabilité des méthodes numériques utilisées est complexe. Il tient essentiellement au fait que les conditions requises pour que les algorithmes disponibles soient utilisables ne sont pas toujours satisfaites par les problèmes réels. D'autre part, on se trouve tributaire des contraintes inhérentes au calcul sur ordinateur (représentation finie des nombres, etc...).

En simulation, trois types de méthodes numériques sont utilisées :

- méthodes de résolution de systèmes d'équations algébriques non linéaires $F(X) = 0$ (calcul des états stables).

- méthodes d'intégration de systèmes d'équations algébriques et différentielles couplées (calcul des réponses transitoires).

- méthodes de résolution de systèmes linéaires (réponse en fréquences, systèmes linéarisés obtenus dans les deux types de méthodes précédentes).

Nous aborderons successivement les deux premiers points. La résolution des systèmes linéaires a été examinée dans le chapitre I. En effet, dans IMAG3, l'algorithme correspondant est utilisé dans la phase de traduction pour générer une séquence codée qui est une donnée pour les algorithmes d'analyse au même titre que les équations du circuit. Aussi, nous indiquerons seulement l'influence de cette approche sur les deux premiers types de méthodes.

La troisième partie sera consacrée aux problèmes liés à l'optimisation des circuits électroniques. Ce domaine s'inscrit dans la tendance s'efforçant d'aboutir à une coopération plus étroite entre l'homme et la machine dans le processus de conception.

Le schéma d'une procédure de conception utilisant l'optimisation peut se décrire ainsi :

(1) Formuler une fonction objectif qui reflète les critères de conception et choisir un ensemble de paramètres du circuit (paramètres de conception) pour lesquels on recherche une valeur optimum.

(2) Choisir une valeur initiale pour ces paramètres.

(3) Simuler le circuit et ajuster la valeur des paramètres de manière à satisfaire les critères de conception (minimisation de la fonction objectif).

Le troisième pas s'effectue de manière itérative, aussi, l'optimisation utilise la simulation de manière intensive puisque chaque itération utilise une (ou plusieurs) simulation du circuit.

Ainsi, pour que l'optimisation ne conduise pas à des coûts prohibitifs, il est nécessaire que la simulation soit extrêmement rapide. Les performances obtenues en simulation par le système IMAG3 sont une première tentative pour atteindre cet objectif.

Avant d'aborder le calcul des états stables, faisons quelques remarques valables pour tous les calculs que nous allons examiner. L'utilisateur électronicien se contente généralement d'une précision modeste sur les résultats. A ceci deux raisons :

- les méthodes de fabrication des composants ne permettent pas d'assurer l'exactitude des valeurs nominales de leurs paramètres physiques (on parle de résistances à 10 %, 5 %, etc...).

- les schémas équivalents ne sont que des approximations des composants physiques complexes, et ces approximations sont en réalité encore assez mal connues.

Dans la plupart des cas, une précision de l'ordre de 10 % sera suffisante. D'autre part - ce qui peut sembler contradictoire avec la remarque précédente - tous les calculs sont effectués en double précision (notation FORTRAN IV),

c'est-à-dire que chaque nombre est codé sur 64 positions binaires, au lieu de 32 habituellement. Ceci a été rendu nécessaire, non pas pour atteindre à une grande précision sur les résultats, mais pour permettre aux algorithmes de s'affranchir dans une certaine mesure des contraintes de calcul sur ordinateur dont nous parlions plus haut, et d'obtenir une plus grande fiabilité.

Précisons enfin que nous n'aborderons pas le problème de l'analyse en régime alternatif car nous n'avons rien ajouté à ce qu'elle était dans IMAG2 [JAC 70].

Signalons que Messieurs Le Faou et Matvitchouk [LE FAOU 76] ont développé une méthode permettant d'aborder de façon nouvelle ce type d'analyse. Cette méthode a été mise en oeuvre dans IMAG3.

I - Calcul des états stables du circuit

L'un des problèmes essentiels que l'on rencontre à propos de l'emploi de méthodes numériques dans un système tel qu'IMAG3 tient à sa généralité, c'est-à-dire à sa capacité de simuler (ou d'optimiser) des circuits de nature et de fonctionnement très différents.

Cette propriété est bien entendu une qualité du système, mais elle rend difficile la détermination des caractéristiques des équations de fonctionnement qui pourraient être intéressantes pour leur résolution.

Rappelons que l'état stable d'un circuit caractérise sa réponse en régime stabilisé, tous les courants et les tensions du circuit gardent une valeur constante.

Dans ce cas, les équations de fonctionnement E_c s'écrivent :

$$(I,1)\quad \begin{cases} G\,(VP, y, t_o, E\,(y, VP, t_o), J\,(y, VP, t_o)) = o \\ F\,(y, VP, t_o, E\,(y, VP, t_o), J\,(y, VP, t_o)) = o \end{cases}$$

(I, 1) est un système d'équations algébriques non linéaires.

Bien que la résolution numérique d'équations algébriques non linéaires soit un problème classique, il n'en présente pas moins des difficultés importantes qui sont loin d'être résolues aussi bien sur le plan théorique que sur le plan pratique.

A cette difficulté inhérente aux méthodes numériques disponibles s'en rajoute une autre, qui tient à la formulation des modèles mathématiques de composants complexes (MOS, etc...) et qui, jusqu'ici - à notre connaissance - n'a été résolue par aucun des systèmes existants.

Elle tient au fait que le modèle d'un tel composant n'est pas un modèle mathématique particulier, mais en réalité plusieurs modèles mathématiques. Chacun correspondant à une zone de fonctionnement physique du composant.

Après avoir indiqué les principales caractéristiques des méthodes utilisables et des méthodes effectivement utilisées dans IMAG3 pour résoudre le système (I.1), nous montrerons que cette approche est insuffisante. En fonction des modèles mathématiques des composants complexes, nous donnerons une autre approche permettant de résoudre le problème du calcul des états stables de manière plus rigoureuse.

I.1 - Les méthodes

Nous nous intéressons ici au problème tel qu'il est appréhendé dans IMAG2 et IMAG3, c'est-à-dire résoudre le système algébrique non linéaire.

$$f(x) = 0 \qquad (I.2)$$
$$x = \{y, VP\} \in R^n$$

f et $\frac{\partial f}{\partial x}$ pouvant être discontinues en certains points de R^n.

I.1-1 - Les méthodes utilisables

On peut distinguer essentiellement deux types de méthodes pour résoudre de manière approchée un système $f(x) = 0$.

- les méthodes itératives de points fixes,
- les méthodes de partitionnement.

I.1-1.1 - Méthodes de points fixes

On résoud $f(x) = 0$ en recherchant le point fixe d'une fonction associée g , c'est-à-dire un point x^* tel que :

$$x^* = g(x^*) \qquad (1.3)$$

Si l'on considère la séquence :

$$x_{k+1} = g(x_k) \qquad (I.4)$$

alors, sous certaines conditions, cette séquence peut converger vers un point fixe unique et l'on a :

$$x^* = \lim_{k \to \infty} x_k$$

Un critère de convergence est le suivant :

Si g est une application contractante de R^n dans R^n, c'est-à-dire s'il existe une constante $K < 1$ telle que :

$$//g(x) - g(y)// \leq K // x-y // \qquad x, y \in R^n$$

alors, g a un point fixe unique et la séquence $x_{k+1} = g(x_k)$ converge vers ce point fixe.

Remarquons que si K est proche de 1, la convergence peut être très lente. Aussi, l'un des problèmes majeurs dans l'élaboration effective de ces méthodes est de construire une application g qui entraîne une convergence rapide.

Définissons :

$$g(x) = x - K(x)\, f(x) \qquad (I.5)$$

où K(x) est une matrice (n x n) non singulière pour un point fixe x^* de g(x).
Toutes solutions x^* de f(x) = 0 est un point fixe de g(x) comme on peut le constater en faisant $x = x^*$ dans (I.5).
Inversement, si x^* est un point fixe pour g(x), alors :

$g(x^*) = x^* = x^* - K(x^*)\, f(x^*)$, ce qui entraîne $f(x^*) = 0$ puisque $K^{-1}(x)$ existe.

Donc :

x^* est une solution de f(x) = 0 si et seulement si x^* est un point fixe de (I.5).

Le problème est ainsi ramené à choisir K(x) telle que l'itération (I.4) converge vers un point fixe.

Un choix intéressant est $K(x) = J^{-1}(x)$ où J(x) est la matrice Jacobi de f. On obtient l'itération classique de Newton-Raphson :

$$x_{k+1} = x_k - J^{-1}(x_k)\, f(x_k) \qquad (I.6)$$

Nous avons supposé jusqu'ici que l'application g était contractante sur R^n. Cependant, étant donné un problème de point fixe dans R^n, il est généralement impossible de dire pratiquement si elle est ou non contractante sur tout R^n ou même sur une partie de R^n.

En effet, il existe des théorèmes donnant des conditions sur f et/ou g pour que l'application g soit contractante (par exemple le théorème d'Ostrowski-Kantorovitch pour la méthode de Newton-Raphson). Mais ces théorèmes sont :

- soit des théorèmes d'existence
- soit ne permettent pas de construire effectivement dans la pratique les domaines où g serait contractante.

Ceci signifie en particulier que le choix de la valeur initiale x_o de l'itération (I.4) reste arbitraire.

D'autre part, les théorèmes supposent que les applications utilisées sont continues et possèdent des dérivées continues. Les méthodes de points fixes ne sont donc pas, dans le cas général, adaptées au problème posé suivant la formulation du début de ce paragraphe en raison de l'impossibilité pratique de construire le domaine de convergence et de la présence éventuelles de discontinuités.

I.1.1.2 - Méthodes de partitionnement

Elles reposent sur des principes différents des méthodes de points fixes et sont du type suivant (cas d'une seule équation) :
supposons déterminé un intervalle $[a, b]$ de R tel que :

(1) f(x) est continu sur $[a, b]$
(2) f(a) et f(b) sont de signes contraires.

Ces conditions entraînent qu'il existe au moins une racine entre a et b. Par des dichotomies successives, on parvient à encadrer une racine par deux nombres a_k et b_k distants de $\frac{b-a}{2^k}$. On arrête le calcul lorsque $\frac{b-a}{2^k}$ est suffisamment petit.

La simple dichotomie peut être remplacée par des techniques plus sophistiquées (recherche de Fibonacci, etc...).

L'avantage de ces méthodes par rapport à celles de points fixes - dans le cas où f est supposée continue - est que ne se pose pas le problème du domaine de convergence. En effet, dans ce cas, si la condition (2) est satisfaite - ce qui peut se vérifier pratiquement - alors, il existe au moins une solution et l'algorithme converge. Elles semblent donc intéressantes, si toutefois on parvient à les étendre facilement à un système de dimension quelconque.

Notons cependant que leur convergence est généralement plus lente que dans le cas de méthodes comme celle de Newton-Raphson (si l'on se trouve dans son domaine de convergence). Aussi, elles doivent plutôt être envisagées comme des méthodes de localisation rapide de la racine et permettant d'initialiser une méthode de point fixe au voisinage de celle-ci.

I.1.2 - Les méthodes utilisées dans IMAG3

Pour le problème tel qu'il est formulé au début de ce paragraphe, aucune des méthodes utilisables ne convient véritablement à la fois :

- par la présence éventuelle de discontinuités
- et même dans le cas où la continuité est assurée par l'impossibilité pratique de construire un domaine de convergence pour les méthodes de points fixes.

Aussi, il en résulte nécessairement un certain arbitraire, tant au niveau du choix des méthodes qu'au niveau des résultats que l'on pourra obtenir.

Les méthodes utilisées dans IMAG3 étaient déjà - à quelques modifications de détails près - utilisées dans IMAG2. Le lecteur intéressé pourra se reporter à JAC [70], LE FAOU [72], REYNAUD [74], HENN [74].

Historiquement, la première méthode utilisée a été la méthode classique de Newton-Raphson déjà citée :

$$x_{k+1} = x_k - J^{-1}(x_k)\, f(x_k) \qquad (I.7)$$

Elle réalise un compromis intéressant entre une convergence rapide (convergence quadratique) et sa facilité de mise en oeuvre. Cependant, sa convergence n'est pas globale dans R^n comme l'on peut s'y attendre, et l'on constate des échecs. Expérimentalement, il semble que le nombre d'échecs devient important pour des systèmes de dimension $n > 10$ et pour des circuits comportant des éléments non classiques.

Aussi, une autre méthode a été mise en oeuvre.

Elle est dérivée de celle de Newton-Raphson. On introduit un paramètre dans l'itération (I.7) :

$$x_{k+1} = x_k - r_k\, J^{-1}(x_k)\, f(x_k) \qquad (I.8)$$

où r_k est un réel positif choisi de façon à rendre minimum

$$E(r) = \frac{//f(x_{k+1})//}{//f(x_k)//} .$$

L'idée initiale de cette méthode est due à Broyden [Broy 65]. Elle apparaît comme une méthode hybride entre celle de Newton-Raphson et une méthode de descente pour le critère //f(x)// minimum.

Le choix de r_k est exposé dans LE FAOU [72].
Notons que cette méthode assure seulement une décroissance de //f(x)// ce qui ne garantit pas la convergence vers la solution. Aussi, elle n'a pas de justification théorique. En effet, si la valeur initiale x_o est choisie dans le domaine de convergence de la méthode de Newton, le paramètre r est inutile car il sera choisi voisin de 1 et sa détermination conduira à des calculs auxiliaires inutiles.

Si x n'est pas dans le domaine de convergence et r est choisi "loin" de 1, la direction calculée perd sa signification et les itérés successifs x_i sont choisis dans des régions arbitraires. Cependant, d'un point de vue pratique, cette méthode présente un gain considérable par rapport à celle de Newton-Raphson.

Pour des valeurs "moyennes" de n ($10 < n < 50$), elle donne de bons résultats (statistiquement 1 échec pour 10 exemples).

Pour des valeurs de n supérieures, on constate que les échecs sont plus nombreux sans que l'on puisse expliquer clairement ce phénomène.

Une des causes d'échecs importante est l'existence possible d'un "cul de sac", c'est-à-dire d'un minimum relatif de //f(x)// strictement positif. Dans ce cas, seul le choix d'une nouvelle valeur initiale x peut permettre d'aboutir à la solution.

Ces deux méthodes - actuellement opérationnelles dans IMAG3 - même si elles permettent souvent d'obtenir de bons résultats, ne peuvent être satisfaisantes dans tous les cas. En particulier, leur comportement en présence de discontinuités reste aléatoire.

D'autres méthodes ont été envisagées. En particulier, l'intégration d'un système différentiel associé [Branin 72] =

$$\frac{df}{ds} + f(x) = 0 \qquad (I.9)$$

f ne dépendant pas explicitement de s, on a :

$$\frac{df}{ds} = \left(\frac{\partial f}{\partial x}\right) \frac{dx}{ds} = J \frac{dx}{ds}$$

Ce qui permet d'écrire (I.9) sous la forme :

$$\frac{dx}{ds} = - J^{-1} f(x) \quad \text{en supposant } \det(J) \neq 0$$

La solution de (I.9) s'écrit :

$$f\{x(s)\} = f(x_o)\, e^{-s}$$

Donc, x(s) est bien solution de f(x) = 0 quand $s \to \infty$.
Si J est singulier sur une hypersurface S^{n-1} de S^n, la méthode cesse de fonctionner.
Le vecteur unitaire tangent à la courbe x(s) en un point où det [J] ≠ 0 est, d'après les formules de Cramer,

$$T = - \text{Adj}\,(J) * f(x) * \text{Sgn}\,(\det\,(J)) \;/\; // \text{Adj}\,(J)\,.f(x) //$$

De part et d'autre de S^{n-1}, les vecteurs T sont de sens opposés, ce qui conduit à permettre un changement de signe dans (I.9) de manière à poursuivre dans le même sens sur la trajectoire :

$$\frac{df}{ds} \pm f(x) = 0 \qquad (I.10)$$

soit :

$$\frac{dx}{ds} = \frac{\text{adj}\,(J) * f(x)}{/\det\,(J)/} \qquad (I.11)$$

La méthode suivante a également été envisagée :
écrivons (I.2) sous la forme plus complète :

$$f(x, u) = 0 \qquad (I.12)$$

où u est un vecteur d'entrée d'excitations constantes (sources de courants et de tensions).

L'idée consiste à résoudre (I.12) en faisant varier progressivement les excitations de 0 jusqu'à leurs valeurs nominales u^*. Ainsi, on discrétise u en :

$$u_o = 0, u_1, \ldots, u_k = u^*.$$

et l'on se ramène à résoudre successivement (k+1) systèmes algébriques

$$f(x, u_i) = 0$$

Pratiquement, si l'on suppose que u et x sont fonctions d'un paramètre s, on peut transformer (I.12) en un système différentiel

$$\frac{\partial f}{\partial x} \cdot \frac{\partial x}{\partial s} + \frac{\partial f}{\partial u} \frac{\partial u}{\partial s} = 0 \qquad (I.13)$$

que l'on peut intégrer par un schéma d'intégration numérique implicite tel qu'on pourra en trouver dans le paragraphe suivant.

L'idée de transformer un système d'équations algébrique en un système différentiel associé que l'on résoud numériquement, a été introduite indépendemment par Lahaye [Lah 48], Davidenko [Dav 53], Goldberg et Richard [Gold 63].

Bien que ce type de méthodes soit théoriquement plus élaborées que la méthode de Newton-Raphson, le problème des discontinuités n'a pas pu y être résolu de manière satisfaisante, ce qui finalement conduit à un pourcentage d'échecs supérieur à la méthode de Newton-Raphson.

Actuellement, dans IMAG3 - ainsi d'ailleurs que dans d'autres systèmes analogues - le problème du calcul des états stables demeure l'un des plus critiques malgré sa simplicité apparente. Aussi, un effort important doit-il être accompli dans ce domaine.

En effet, ce type de calcul est un instrument indispensable à l'étude des circuits logiques. D'autre part, l'étude des réponses en régime transitoire s'effectue généralement en excitant un circuit supposé être dans un état stable préalablement calculé.

Remarquons enfin que les méthodes précédentes utilisent à chaque itération la solution d'un système linéaire :

$$\left[\frac{\partial f}{\partial x}\right] \cdot \Delta x = B$$

L'intérêt de l'invariant de résolution de ce système introduit au chapitre I apparaît ici clairement.

I.2 - Position du problème du calcul des états stables en fonction des modèles.

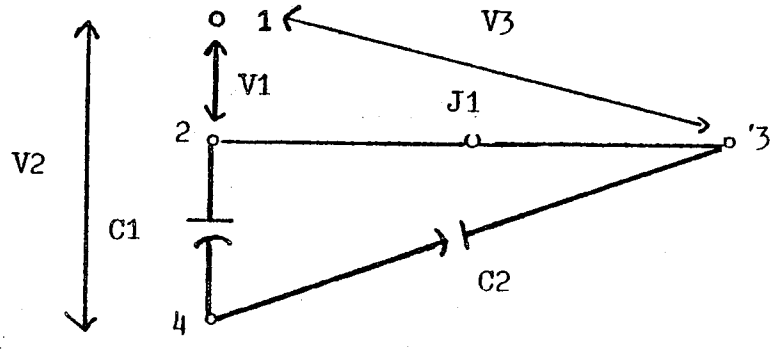
Les difficultés rencontrées dans le calcul des états stables, tant dans IMAG2 que dans IMAG3, nous ont conduit à nous interroger sur la nature des modèles mathématiques complexes et à y rechercher la source d'un grand nombre de ces difficultés.

Si l'on regarde la littérature consacrée au problème du calcul des états stables, on s'aperçoit que l'effort de recherche porte sur l'élaboration d'un algorithme permettant de résoudre $f(x) = 0$, $x \in R^n$ possédant un domaine de convergence plus important ou ayant une vitesse de convergence plus grande que les algorithmes actuellement connus. Ce type de recherche est bien entendu intéressant, cependant, tous ces algorithmes sont élaborés en supposant des hypothèses sur f (comme la continuité) généralement non vérifiées dans la pratique.

Il en résulte que les progrès introduits par ces algorithmes restent relativement faibles.

Aussi, plutôt que d'envisager le problème sous cet angle, nous considérons qu'un outil C.A.O. est bien entendu constitué d'un ensemble d'algorithmes, les plus performants possible, mais également d'un langage de communication homme-machine qui doit permettre à l'utilisateur de communiquer au système le maximum de renseignements sur l'objet étudié dans le but souhaité de rendre le système aussi efficace que possible.

Considérons le schéma suivant représentant un type de transistor MOS :



dont les relations s'écrivent :

$$K = \begin{cases} 0 & \text{si } V1^2 \geq V3^2 \\ 1 & \text{sinon} \end{cases}$$

$$VB = K * V1 + (1-K) * V3$$

$$D = (VB + a_1)^{a_2}$$

$$VT = a_3 * D$$

$$VGT = V2 - VB - VT$$

$$VGTE = VGT / (a_4 * VGT)$$

$$VD = (2 * K-1) . (V3 - V1)$$

$$VDE = a_5 * VD * (1 - a_6 * VGT)$$

$$JO = \begin{cases} 0 \text{ si } VGT < 0 \\ a_6 \text{ sinon} \end{cases}$$

$$JOHM = VDE * (VGTE * a_7 - VDE)$$

$$JSAT = VGTE * (VGTE - a_8 . VDE)$$

$$J = \begin{cases} JO * JOHM & \text{si } VDE < VGTE \\ JO * JSAT & \text{sinon} \end{cases}$$

$$J1 = (2 * K-1) * J.$$

où les a_i sont des constantes.

On remarque que l'expression de la source J1 en fonction des tensions V1, V2 et V3 du circuit, est complexe. En particulier, cette expression change de forme suivant le domaine de R^3 : {V1, V2, V3} dans lequel on se trouve.

Au niveau où nous nous plaçons de l'étude des composants électroniques, ceux-ci peuvent être considérés comme ayant un fonctionnement continu. Cependant, la modélisation mathématique ne traduit qu'une approximation de ce fonctionnement. En particulier, on ne sait généralement pas construire, pour un composant complexe, un modèle analytique représentant correctement toutes ses possibilités de fonctionnement physique. Aussi, on est conduit à construire le modèle à l'aide de représentations mathématiques différentes pour chaque plage de fonctionnement.

Notons $\{M_i^j\}$ l'ensemble des représentations mathématiques utilisées pour décrire le composant j dans ses différentes plages de fonctionnement.

Considérons un circuit de k composants interconnectés.
L'ensemble des modèles mathématiques {M} permettant de représenter le circuit dans R^n sera :

$$\{M\} = X_k \{M_i^j\} \quad \text{produit cartésien des k ensemble } \{M_i^j\}$$

Le problème que nous nous posons est le suivant :

(1) quel est parmi cet ensemble de modèles, celui qui représente le circuit supposé dans un état stable.

D'un point de vue algorithmique, la question précédente se traduit par :

(2) quel système algébrique f(x) = 0 - f continue - faut-il résoudre pour trouver un état stable.

Et d'un point de vue physique :

(3) quelle est la zone de fonctionnement de chacun des composants du circuit lorsque celui-ci est dans un état stable.

La réponse à l'une de ces trois questions permet de répondre aux deux autres car elles sont équivalentes. Il est bien évident que l'on ne peut répondre directement à la question (2). La raison en est que la fonction f(x) n'a de signification vis-à-vis du circuit que pour les points x^* tel que $f(x^*) = 0$.

Actuellement, dans IMAG2 et IMAG3, on ne sait pas quel est le système f(x) = 0 - f continue - qu'il faut résoudre pour trouver un état stable.
Ce qui conduit au niveau algorithmique à la situation suivante :
considérons un processus itératif comme la méthode de Newton-Raphson :

$$x_{k+1} = x_k - J^{-1} \cdot f(x_k)$$

Soit x_o un point initial dans l'espace d'état du circuit. A ce point correspond une fonction fx_o et l'on tente de résoudre $fx_o(x) = 0$.
Supposons, ce qui est fréquent, qu'à un itéré x_i, corresponde une fonction $fx_i(x)$. On tente alors de résoudre $fx_i(x) = 0$, etc...

Ces choix successifs traduisent seulement le fonctionnement d'un algorithme et n'ont aucun lien avec le fonctionnement réel du circuit. D'autre part, le choix de la fonction initiale fx_o est arbitraire.

Schématiquement, on demande à un algorithme capable en principe de résoudre un type de problème de choisir le problème particulier à résoudre parmi un ensemble de problèmes. Une telle situation comporte une grande part d'arbitraire et explique une grande partie des difficultés rencontrées.

Lever l'arbitraire sur le choix du problème implique de répondre à la question (3). Un algorithme permettant de répondre à cette question à partir des renseignements dont dispose le système sur le circuit, c'est-à-dire :

- sa topologie
- la valeur de ses composants.

serait un algorithme heuristique construit à partir de règles traduisant un fonctionnement qualitatif. Cette approche pourrait être considérée comme une application de l'intelligence artificielle à l'analyse des circuits. Cependant, dans l'état actuel des travaux dans ce domaine [Suss 75], nous pensons qu'il est préférable de faire effectuer cette heuristique par le concepteur du circuit.

En effet, un électronicien n'élabore pas un circuit au hasard, mais poursuivant un objectif précis, il utilise son intuition et son expérience à l'atteindre. Aussi, il a une idée de son fonctionnement physique qu'il convient d'utiliser.

En particulier, il sera généralement capable - au moins partiellement - de répondre à la question (3).

Ce que nous proposons de faire, est d'enrichir la sémantique de la description du circuit de manière à permettre à l'utilisateur de communiquer au système sa réponse à la question (3), d'en déduire une réponse à la question (2), d'utiliser les renseignements ainsi obtenus dans le calcul des états stables.

I.3 - Extension du langage de description

Le but recherché est de définir un langage permettant à l'utilisateur de communiquer au système des renseignements qualitatifs sur le fonctionnement du circuit de manière à déterminer quel système d'équations $f(x) = 0$ - f continue - traduit un fonctionnement en régime stabilisé, et d'autre part de préciser dans quelle région de l'espace d'état se trouve un état stable.

On peut distinguer schématiquement deux catégories d'utilisateurs :

- les personnes travaillant à l'élaboration de schémas équivalents de composants complexes,
- celles qui définissent des circuits et utilisent ces schémas équivalents.

Il faut fournir aux premiers les moyens de préciser les caractéristiques de leurs schémas et aux seconds les moyens d'utiliser ces caractéristiques dans le contexte d'un circuit particulier.

Comme nous l'avons vu dans le chapitre I, le langage de description contient deux notions essentielles :

- la description de schémas équivalents par des types abstraits,
- l'utilisation dans des circuits particuliers par génération à partir des types abstraits déjà définis.

Cette distinction correspond naturellement aux deux catégories d'utilisateurs et il convient de l'exploiter au mieux possible dans l'extension que nous envisageons.

I.3.1 - Extension du langage au niveau des types

Un 'TYPE' définit un schéma équivalent paramétré. Ces paramètres ne prenant des valeurs effectives que lors de la génération.
Poursuivant dans cette voie, nous allons paramétrer des renseignements qualitatifs sur le 'TYPE'.

Les différents modèles mathématiques décrivant un composant sont introduits par des expressions SI. A chaque ensemble de choix possibles d'occurrence des expressions SI correspond un modèle mathématique continu.

Permettre de préciser un modèle mathématique revient donc à donner une syntaxe permettant d'indiquer quel ensemble sera considéré.

De la même manière que dans une déclaration de 'TYPE' figurent des paramètres quantitatifs, peuvent figurer des paramètres qualitatifs, chacun correspondant à un mode de fonctionnement :

TYPE : < nom de type > (< liste de connecteurs >) < liste de paramètres quantitatifs > ; < liste de paramètres de fonctionnement > $

Exemple : TYPE : T (E, B, C) ,,,,,,, $ BLOQ, SAT $

BLOQ insique (par exemple) un fonctionnement en mode bloqué et SAT en mode saturé.

Ainsi, lors de la génération d'un composant effectif, on écrira :

COMPOSANT : 'T1' (T) ——— ; BLOQ $
: 'T2' (T) ——— ; SAT $
: 'T3' (T) ——— ; BLOQ, SAT $

T1 fonctionnera en mode bloqué, T2 en mode saturé.
T3 pourra fonctionner dans les deux modes.

A la différence des paramètres quantitatifs qui correspondent à des affectations de valeurs numériques, les paramètres de fonctionnement correspondent à des fonctions qui doivent être interprétées par le corps du TYPE.

Ainsi, ces fonctions peuvent déterminer des occurrences particulières d'expressions SI. Dans ce cas, on a :

SI (< expression logique >) < paramètre de fonctionnement > : (< expression >)
< paramètre de fonctionnement > : (< expression >) $

Exemple : A = SI (V2.GT.V1) FONC1 : (B * * 2)
FONC2 : (C * * 2-1) $

Lors du calcul du régime continu, le calcul de A s'effectuera de la manière suivante :

- dans un fonctionnement caractérisé par le paramètre FONC1, A ← B * * 2

- si le paramètre choisi est FONC2, A ← C * * 2 - 1

- dans un fonctionnement possible suivant les deux modes, le choix s'effectue sur la valeur de l'expression logique comme auparavant.

Remarque : l'expression logique intervient ici comme une contrainte déterminant dans l'espace (V1, V2) les zones de fonctionnement FONC1 et FONC2.

Ces paramètres de fonctionnement peuvent également correspondre à des contraintes sur les variables du schéma équivalent. Ces contraintes spécifiant des zones de fonctionnement du composant dans son espace d'état.

Exemple : TYPE : M (A, B, C, D) ,,,,,; FONC1, FONC2, FONC3 $

description du type

spécifications

FONC1 :

liste de contraintes

$

FONC2 :

$

FONC3 :

$ $

Le mot-clé spécifications indique la description des paramètres de fonctionnement sous la forme de contraintes.

Ces contraintes sont de la forme :

< expression arithmétique > .GT. / .LT. < expression arithmétique >

Exemple : (V1 + V2).LT. (V3 * * 2)

(A * (B - 1)).GT. C

Ainsi, l'utilisateur qui construit des circuits à partir de composants n'a pas à connaître la réalisation effective dans le corps du 'TYPE' des paramètres de fonctionnement, mais seulement leur signification extérieure.

Ceci se rapproche de manière évidente de la programmation modulaire.

I.3.2 - Utilisation effective dans un circuit

Si l'on considère un composant particulier C_i, on peut y distinguer 2 niveaux sémantiques.

Le premier niveau correspond au composant lui-même, c'est-à-dire à sa nature physique, ses possibilités de fonctionnement. Ce niveau est exprimé dans le 'TYPE' et dans la valeur de ses paramètres physiques.

Le second niveau correspond à son utilisation effective dans un circuit particulier. Ce second niveau sémantique est donc lié au contexte du circuit. Il est traduit de manière implicite par l'application des lois de Kirchoff aux bornes de sorties du composant. Une autre manière de le traduire est de spécifier les paramètres de fonctionnement :

Exemple :

T1 (1, 2, 3) / ,,,,,; BLOQ /

Le transistor T1 implanté dans le circuit entre les noeuds 1, 2 et 3 fonctionnera suivant un mode spécifié par le paramètre BLOQ.

L'extension dont nous avons esquissé les grandes lignes ci-dessus - et qui ne peut être considéré comme définitive - consiste à exprimer a priori des propriétés qui, bien que contenues dans le modèle mathématique, ne peuvent être perçues qu'après une simulation - résolution des équations -.

Il est important de voir que dans une étude quantitative sur ordinateur, un modèle mathématique ne prend de sens qu'à travers l'algorithme utilisé pour l'interpréter - par exemple l'algorithme de résolution du système $f(x) = 0$ - Ceci correspond à la notion d'un objet représenté d'un point de vue informatique par une machine abstraite constitué ici de :

$$M_a = \{\text{équations de fonctionnement}\} + \{\text{algorithmes numériques}\}$$

Le modèle informatique M_a est fortement lié aux algorithmes numériques dont la caractéristique essentielle est qu'ils sont figés. Cette extension peut être considérée comme un premier pas vers le paramétrage des algorithmes par des caractéristiques de l'objet modélisé.

I.4 - Exploitation des renseignements qualitatifs

A chaque paramètre de fonctionnement est lié :

- un choix particulier des occurrences des expressions SI
- et/ou un ensemble de contraintes sur les variables du schéma équivalent.

Dans le cas d'une expression SI, la partie expression logique constitue une contrainte marquant si l'on se trouve dans la zone du paramètre relatif à la première occurrence ou à la seconde.

On peut distinguer deux notions :

(a) - la notion de contraintes associées à un paramètre de fonctionnement et définissant dans l'espace d'état une région associée au paramètre de fonctionnement.

(b) - le choix des occurrences des expressions SI associées à un paramètre de fonctionnement et permettant de définir une fonction f(x) continue pour le calcul d'un état stable.

(b) permet de choisir f(x) continue telle que la solution de f(x) = 0 représente un état stable du circuit.

(a) indique une région de l'espace d'état dans laquelle se trouve un point X^* correspondant à un état stable.

Schématiquement, l'algorithme envisagé est le suivant :

(1) choix des occurrences des expressions SI associées aux paramètres de fonctionnement de chaque composant (choix de f(x))

(2) choix d'un point initial X_0 tel que toutes les contraintes associées aux différents paramètres de fonctionnement soient satisfaites. (Positionnement dans une région de l'espace d'état où se trouve l'état stable).

(3) résolution de f(x) = 0 par un algorithme numérique.

Ces différents points (1), (2) et (3) sont en cours d'élaboration.

L'approche précédente doit permettre d'aborder de manière plus rigoureuse et plus efficace le calcul des états stables de circuits comportant des composants complexes (MOS, etc...) et plus généralement des objets dont le modèle mathéma-

tique est discontinu.

Nous pensons qu'elle devrait être étendue dans le sens de permettre à l'utilisateur de fournir le maximum de renseignements qualitatifs sur le fonctionnement du circuit.

II - Calcul de la réponse en régime transitoire

Le calcul de la réponse en régime transitoire d'un circuit électronique consiste à calculer les valeurs des variables du circuit en fonction du temps pour des sources d'excitation données et à partir de conditions initiales qui sont : - soit précisées par l'utilisateur

- soit correspondent à un état stable calculé précédemment.

Généralement, on s'intéresse à un intervalle de temps $[t_o, t_f]$ déterminé et durant lequel le fonctionnement du circuit présente un intérêt. Cet intervalle doit être choisi avec soin. Le choix d'un intervalle trop petit ne permet pas de se faire une bonne idée du fonctionnement, tandis qu'un intervalle trop grand peut conduire à des calculs inutiles et à des difficultés numériques comme nous le verrons plus loin.

Ce calcul s'effectue en intégrant les équations de fonctionnement E_c sur l'intervalle $[t_o, t_f]$.

Dans IMAG2, ces équations se présentaient comme un système différentiel du premier ordre sous la forme canonique :

$$\frac{dy}{dt} = f(y, t) \qquad (II.1)$$

Dans IMAG3, on considère le système :

$$F(y, \frac{dy}{dt}, u, t) = 0 \qquad (II.2)$$

(II.2) est un système d'équations algébriques et d'équations différentielles couplées.

Les méthodes utilisées dans les deux cas seront donc sensiblement différentes.

Nous supposons que ces équations ont une solution unique relativement à des conditions initiales données.

L'une des caractéristiques essentielles de ces systèmes est qu'ils sont "stiffs", c'est-à-dire que les constantes de temps mises en jeu sont très différentes. Nous examinerons d'abord cette propriété et nous verrons qu'elle conduit à utiliser des méthodes particulières que nous exposerons dans une deuxième partie.

II.1 - Notion de système "stiff" et de stabilité

Nous nous contenterons de donner ici une idée intuitive de la notion de système différentiel "stiff".

Ces systèmes se rencontrent dans l'étude de phénomènes présentant des constantes de temps très différentes (circuits électroniques, analyse de phénomènes radio-actifs, contrôle de processus, etc...).
Le terme constante de temps est utilisé par les physiciens et les ingénieurs pour parler de la rapidité d'évolution d'un phénomène. Par exemple, l'équation $y' = \lambda y$ a pour solution $y = c\, e^{\lambda t}$. Si λ est négatif, y décroît d'un facteur $\frac{1}{e}$ dans un temps $t_d = \frac{-1}{\lambda}$.
t_d est la constante de temps.

Dans un système, les différentes composantes peuvent varier plus ou moins rapidement et dans un circuit électronique, des rapports de l'ordre de 10^6 à 10^9 entre les constantes de temps extrêmes sont relativement fréquents.

L'intégration de tels systèmes par des méthodes numériques présente des difficultés dont on n'a pris conscience que récemment à l'occasion d'applications particulières (circuits électroniques, cinétique chimique). Ainsi, l'ouvrage classique de Henrici [Hen 62] ne les mentionne pas.

Pour comprendre ces difficultés, il est nécessaire d'introduire quelques propriétés des schémas numériques utilisés.

II.1.1 - Caractéristiques générales des méthodes numériques d'intégration

Parmi les diverses méthodes connues permettant d'obtenir une solution approchée, celles basées sur le principe de la discrétisation semblent les plus adaptées pour un emploi sur ordinateur.

Considérons le système différentiel :

$$(II.3)\begin{cases} y' = f(y, t) \quad t \in [a, b] \subset R \\ y(a) = y_a \in R \end{cases}$$

Ces algorithmes visent à obtenir une approximation numérique y_i de la valeur théorique de la solution inconnue $y(t)$ qu'aux seuls points t_i d'une partie finie de $[a, b]$.

Pour fixer les idées, on peut supposer que les t_i sont définis par :

$$(II.4)\begin{cases} t_i = a + i\,h \quad i = 0, 1, \ldots, k \\ h = \dfrac{(b - a)}{k} \end{cases}$$

Le pas de discrétisation h étant supposé constant sur tout l'intervalle $[a, b]$. On distingue traditionnellement les méthodes à pas séparés des méthodes à pas liés.

Dans les premières, connaissant y_i (approximation de $y(t_i)$), on détermine y_{i+1} par :

$$y_{i+1} = y_i + h\ \Psi(y_i, t_i, h)$$

où Ψ est une fonction de y_i, t_i et h seuls.

Dans les méthodes à pas liés, Ψ est également une fonction de $y_{i+1}, \ldots, y_{i-q+1}$ (méthodes à q pas liés).

Introduisons les notions d'erreurs de troncatures et de stabilité en nous appuyant sur la méthode la plus simple : la méthode d'Euler.

Soit le système (II.3). Nous pouvons approcher $y(h)$ en utilisant les deux premiers termes d'un développement en série de Taylor de $y(t)$.

$$y(h) \simeq y_h = y(a) + h\, f(y(a), a)$$

D'une manière générale, on définit la prochaine valeur en fonction des valeurs courantes par :

$$(II.5)\begin{cases} y_{i+1} = y_i + h\, f(y_i, t_i) \\ \text{où } t_i = i.h \end{cases}$$

Remarque : pour simplifier, h est supposé constant. Notons que la plupart des résultats concernant les méthodes d'intégration démontrés pour h constant, n'ont pas été établis dans le cas où h peut varier au cours du processus.

De nombreux auteurs ont étudié le problème des erreurs commises dans les schémas d'intégration numérique.
Par erreur, il faut entendre la différence :

$$y_t - y(t) = \text{valeur approchée} - \text{valeur exacte.}$$

Les deux types principaux d'erreurs sont :

- l'erreur de troncature, qui provient de l'utilisation d'une formule d'intégration approchée.

- l'erreur d'arrondi, provenant de calculs effectués avec un nombre fini de chiffres significatifs.

Considérons, par exemple, la méthode la plus simple qui est celle d'Euler :

$$y_{i+1} = y(t_i) + h\, f(y(t_i), t_i)$$

L'erreur de troncature sur un pas est : (série de Taylor)

$$y_{i+1} - y_i(t_{i+1}) = -\frac{h^2}{2}\, y''(\theta) \quad \text{avec } \theta \in [t_i, t_{i+1}]$$

L'erreur totale sur un pas est la somme de :

- l'erreur de troncature sur un pas,
- l'erreur d'arrondi sur un pas.

Si l'on considère l'erreur non plus sur un pas, mais sur n pas successifs, celle-ci est une combinaison des erreurs commises pour chacun des pas.

Le lecteur intéressé par ces problèmes, en particulier par les phénomènes de propagation des erreurs et par des formules donnant des majorants, pourra se reporter à Henrici [Hen 62] et Ceschino-Kuntzmann [Cesch 63].

Introduisons maintenant la notion de stabilité.
Il faut bien distinguer :

- la stabilité du système différentiel lui-même (indépendamment de toute méthode numérique).

Définition : un système différentiel est dit stable

si $y(t)$ et $z(t)$ désignant deux solutions, alors

$(y(t) - z(t)) \to 0$ quand $t \to + \infty$

- la stabilité de la méthode numérique utilisée.

Nous nous intéresserons exclusivement à celle-ci par la suite.

On peut en donner la définition intuitive suivante :

définition : une méthode M appliquée à un système différentiel E est dite stable pour $h_o > o$, si une perturbation finie des valeurs initiales produit une perturbation bornée des valeurs numériques suivantes pour tout h tel que $o \leq h \leq h_o$.

Notons que la stabilité n'entraîne pas la convergence de la méthode. Par exemple, $y_n = y_{n-1}$ est stable mais sûrement pas convergente, sauf pour $y' = 0$.

Ce qui précède concerne une perturbation au niveau de la valeur initiale, mais des erreurs peuvent être introduites à chaque pas et un effet cumulatif peut affecter le résultat final. Si ceci n'apparaît pas dans le résultat désiré, il y a alors stabilité absolue.

Définition : une méthode M est a-stable pour un système E et un pas h, si une perturbation d'une valeur y_n n'augmente pas pour les valeurs suivantes y_m avec $m > n$.

Cependant, ce type de définition reste qualitative et se trouve liée à la nature du problème.

Aussi, l'idée d'introduire, pour l'étude de ces phénomènes, l'équation test $y' = \lambda y$ (λ constante réelle ou complexe) et d'y "expérimenter" les qualités d'une méthode, a été utilisée.

Cette idée a été émise pour la première fois - semble-t-il - dans un contexte différent (résolution numérique d'équations linéaires aux dérivées partielles) par Von Neumann.

Dahlquist donne la définition suivante de la A-stabilité.

Définition : (Dahlquist : 1959)

Une méthode est dite A-stable si l'approximation numérique y_n qu'elle donne de $y' = \lambda y$, y_n tend vers o quand $n \to \infty$ pour h fixé et positif et Re $(\lambda) < o$.

L'application à l'équation test de la méthode d'Euler permet de montrer que l'on atteint la a-stabilité si et seulement si :

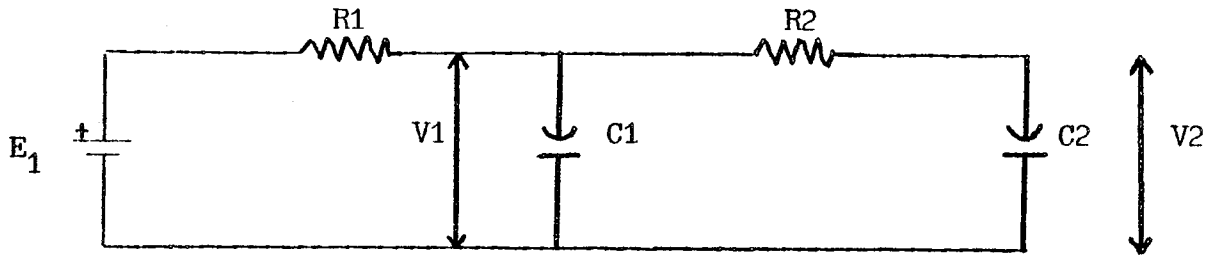
$$\begin{matrix} \lambda < o \\ h < \frac{2}{\bar{\lambda}} \end{matrix} \qquad \text{(III.7)}$$

Ceci montre que, dans le cas de la méthode d'Euler, le pas devra être limité si l'on veut préserver la stabilité.

Comme nous allons le voir dans le paragraphe suivant, cette constatation est d'une grande importance dans l'étude des circuits électroniques et dans une large mesure conditionnera l'utilisation de certaines méthodes.

II.1.2 - Difficultés d'intégration numérique des systèmes stiffs

Considérons le circuit linéaire suivant :



avec R1 = R2 = 1K, C1 = 1nF, C2 = 1pF et E1 = 10 V.

Les équations d'état s'écrivent :

$$\begin{bmatrix} \frac{dV_1}{dt} \\ \frac{dV_2}{dt} \end{bmatrix} = \begin{bmatrix} -2.10^6 & 10^6 \\ 10^9 & -10^9 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} + \begin{bmatrix} 10^{-2} \\ 0 \end{bmatrix} e$$

Les valeurs propres de la matrice de Jacobi sont -10^6 et -10^9, et les constantes de temps (inverse des valeurs propres), 1ns et 1000 ns.

Si l'on veut étudier le phénomène principal, c'est-à-dire intégrer le système sur un intervalle de l'ordre de 5000 ns, le pas d'intégration idéal serait vers 50 ns, de manière à avoir une courbe avec 100 points environ.

Cependant, si nous utilisons la méthode d'Euler, le pas devra nécessairement satisfaire (II.7) pour préserver la stabilité. C'est-à-dire

$$h < \frac{2}{\lambda max} = 2.10^{-9}$$

On sera conduit à effectuer 25 fois plus de pas qu'on le souhaiterait.

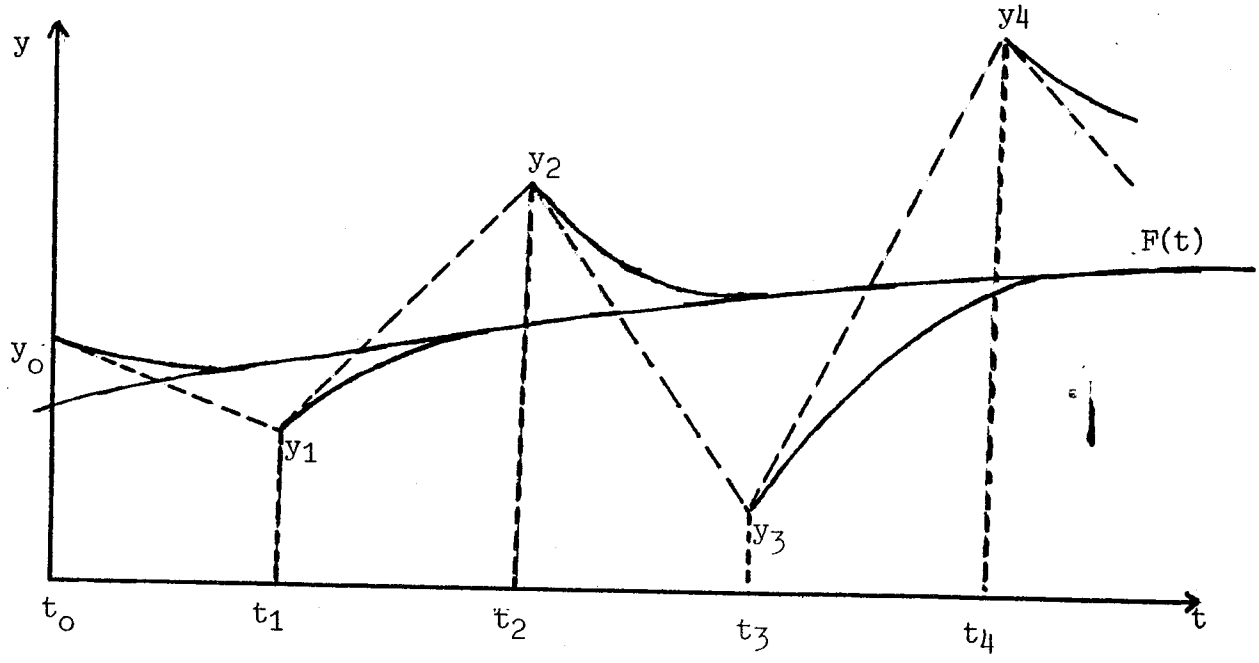
Pour un problème donné, la valeur maximum du pas d'intégration utilisable est appelée rayon de stabilité de la méthode. Il est généralement en relation directe avec la plus petite constante de temps (ou la plus grande valeur propre) du système.

On aura donc intérêt à utiliser dans les problèmes "stiffs" des méthodes ayant un rayon de stabilité maximum.

Considérons l'équation $y' = \lambda(y - F(t)) + F'(t)$
avec $\lambda \ll o$ et F(t) une fonction dont la variation est "lente". Sa solution est de la forme :

$$y = (y_o - F(o))\, e^{\lambda t} + F(t)$$

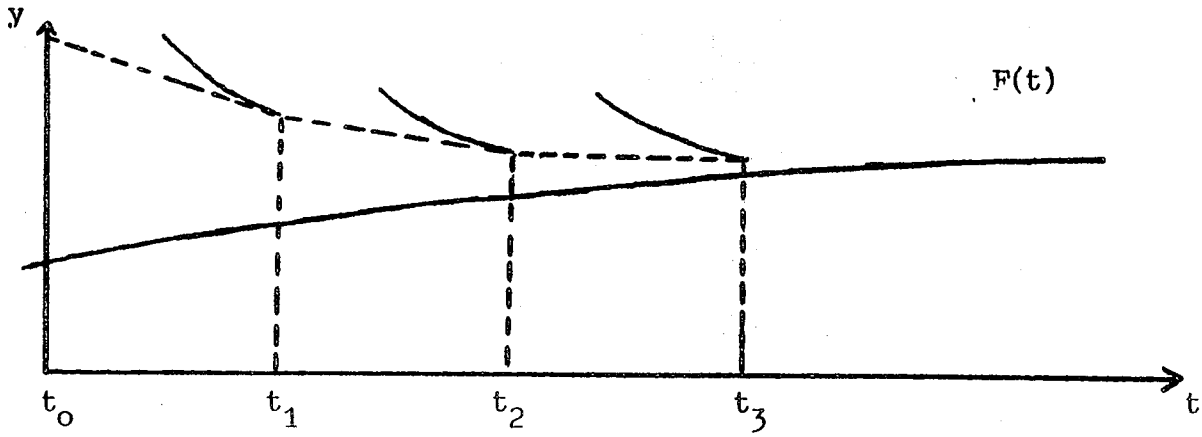
La méthode d'Euler appliquée à cette équation, peut se schématiser de la manière suivante :



Pour t assez grand, le terme en c $e^{\lambda t}$ est insignifiant devant F(t) qui est la composante intéressante de la solution.

Pourtant, on constate que l'on s'en éloigne de plus en plus.
Si $/1 + \lambda h/ > 1$, l'erreur est amplifiée à chaque pas.

Considérons le même problème résolu par la méthode d'Euler implicite $y_{n+1} = y_n + h\, f(y_{n+1}, t_{n+1})$. Remarquons que le second membre est maintenant fonction de y et t pour l'indice n+1. On peut montrer que cette méthode est stable. L'erreur est multipliée par $(1 - h\,\lambda)^{-1}$ à chaque pas.
Si $\lambda < o$, le facteur multiplicatif est toujours inférieur à 1 et l'on a dans ce cas le schéma suivant :

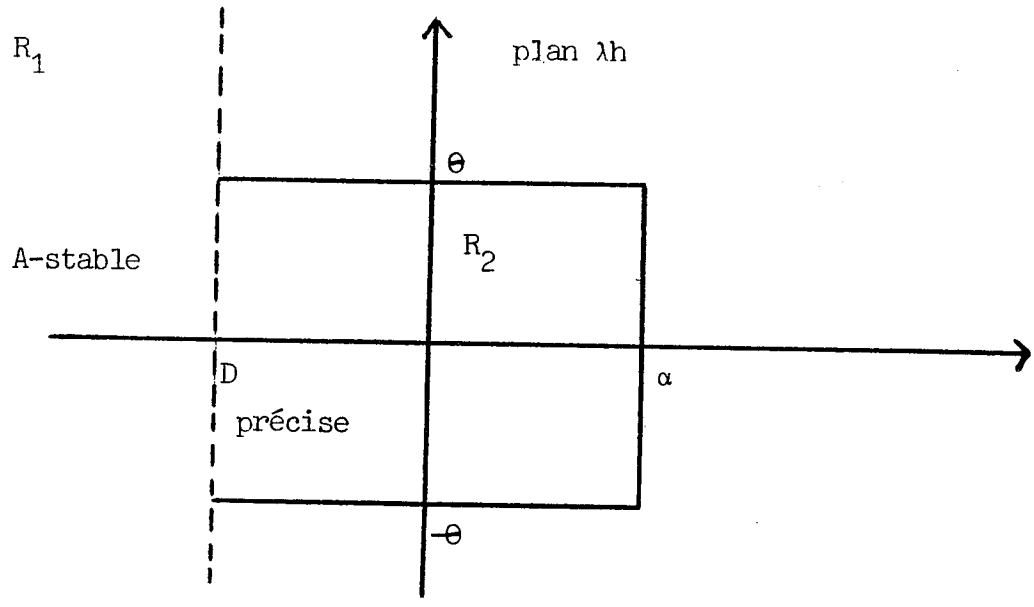


Dahlquist a montré que l'ordre maximum d'une méthode A-stable est 2 et que la méthode A-stable d'ordre 2 avec la plus petite erreur de troncature est la méthode trapézoïdale : $y_{n+1} = y_n + \frac{h}{2}[f(y_n, t_n) + f(y_{n+1}, t_{n+1})]$.

Afin d'élargir la classe des méthodes réellement utilisables, l'exigence de A-stablilté a dû être affaiblie.

Une des contributions les plus intéressantes a été apportée par Gear [GEAR 69]. Il définit la notion de **"quasi-stabilité"** (stiffly-stability).
Définition : une méthode est "quasi-stable" si, dans la région R1($Re(\lambda h) \leq D$) du plan complexe λh, elle est A-stable et si dans la région R2 ($D < Re(\lambda h) < \alpha$, $.Im(\lambda h)/ < 0$), elle est "précise"(*).

(*) - Le terme "précis" signifie ici que la méthode a une précision suffisante dans R2 pour suivre les variations rapides de la solution dans cette région.



Cette définition s'inspire des considérations suivantes : revenons au système test $y' = \lambda y$. $e^{\lambda h}$ est la variation d'une composante au cours d'un pas et due à la valeur propre λ. Soit $\lambda h = u + iv$.

Une valeur de λh proche de l'axe imaginaire et éloignée de l'axe réel correspond à une composante qui oscille et décroît très lentement. Aussi, son calcul précis demande une discrétisation suffisamment fine de l'ordre de 10 pas par période. La partie imaginaire de λh : v, ne peut donc dépasser $\frac{\pi}{5}$ en module et ainsi $/\theta/ < \frac{\pi}{5}$.

D'autre part, dans la région R1, on ne s'intéresse pas à une représentation précise des composantes. La seule exigence sera donc celle de A-stabilité.

Ainsi, il paraît suffisant d'utiliser, dans la partie gauche du plan complexe, des formules stables dans R1 et au voisinage de l'origine.

Gear a montré que certaines formules à pas liés de type :

$$\alpha_o y_n + \ldots + \alpha_k y_{n-k} + h \beta_o f(y_n, t_n) = o$$

considérées pendant longtemps comme peu intéressantes étaient "quasi-stables" pour $k \leq 6$ les rendant ainsi très intéressantes pour l'intégration des systèmes stiffs.

Cependant, dans le cas d'un pas variable, cette propriété n'a pas été démontrée. Comme nous allons le voir, ce sont des formules de ce type qui sont utilisées dans IMAG3.

II.2 - Méthodes utilisées dans IMAG3

L'analyse transitoire des circuits actuels est étroitement liée aux phénomènes d'instabilités rencontrés dans les méthodes d'intégration.

D'une part, ces circuits sont généralement volumineux, augmentant ainsi les chances d'y rencontrer des constantes de temps très différentes.

D'autre part, les constantes de temps de ces circuits (en particulier les circuits logiques : bascules, etc...) varient énormément au cours de la transition entre 2 états stables.

Par exemple, les constantes de temps d'un transistor peuvent se modifier d'un facteur 100 ou plus, entre les zones de blocage et de saturation.

Ceci signifie qu'une technique consistant à séparer le circuit en deux parties ("lente" et "rapide") ne peut être appliquée et que l'on se trouve bien en présence d'un véritable système "stiff". Les méthodes utilisées sont donc spécialement adaptées à ce type de problème.

II.2.1 - Rappels des méthodes utilisées dans IMAG2

Dans le domaine du calcul numérique de la solution d'un système différentiel à conditions initiales, il n'existe pas actuellement de méthodes universelles, mais un ensemble de méthodes dont chacune est adaptée à une classe de problèmes particuliers.

Ceci a conduit les auteurs d'IMAG2 à proposer à l'utilisateur du système, un ensemble de méthodes choisies en fonction des caractéristiques des équations à traiter et des contraintes introduites par les ordinateurs (coût, place limitée en mémoire).

Méthodes classiques

Le système différentiel étant sous forme canonique, ils ont successivement introduit :

- RK1 P1 : méthode d'Euler,
- RK4 P4 : méthode de Runge-Kutta, de rang 4 et d'ordre 4.

Puis a été introduit RK4 P1 (méthode de Runge-Kutta, d'ordre 1 et de rang 4), afin d'augmenter le rayon de stabilité.

Le lecteur intéressé par la formulation de ces méthodes, pourra consulter Ceschino et Kuntzmann p. 34 [Cesch 63].

Ces méthodes sont explicites et possèdent toutes un contrôle automatique du pas en fonction de l'erreur de troncature.
Elles ont l'avantage d'être faciles à mettre en oeuvre car elles exigent peu de calculs auxiliaires, contrairement aux méthodes implicites. D'autre part, elles sont généralement très précises, encore qu'ici, cette propriété n'est pas essentielle, l'électronicien se contentant d'une précision modeste.

Elles présentent cependant l'inconvénient de ne pas être A-stables et ne sont donc pas adaptées aux systèmes stiffs.

- Méthode de Fowler et Warten :

Pour augmenter le rayon de stabilité, une méthode dite "exponentielle" due à FOWLER et WARTEN [Fow 67] a été introduite. C'est une méthode non-linéaire bien adaptée à des non-linéarités de type exponentiel comme il en apparaît fréquemment dans les schémas équivalents de composants électroniques (diodes, transistors, etc...).

$$y(t + h) = y(t) + \int_t^{t+h} f(u, y(u))\, du$$

On suppose que $f(u, y(u))$ est de la forme $A\, e^{\lambda u} + B$ et on choisit A, B, λ de telle sorte qu'il y ait consistance de la fonction et de la dérivée au point t.

On est amené à prendre :

$$B = (y_n - y_{n-1}) / h_1 \quad \text{où } h_1 \text{ est la longueur du pas précédent.}$$

$$A = f(t_n, y_n) - B$$

$$\lambda = \{f(t_n + \delta, y_n + f(t_n, y_n)) - f(t_n, y_n)\} / \delta.A \text{ avec } \delta \leq \frac{h}{4}$$

et la recurrence s'exprime par :

$$y_{n+1} = y_n + h\, \{B + A.(e^{\lambda h} - 1) / \lambda h\}$$

Cette méthode est effectivement intéressante pour des non-linéarités de type exponentiel et est d'ordre 2.

Cependant, bien que plus élevé que dans le cas des méthodes précédentes, son rayon de stabilité n'est pas infini.

- Méthode trapézoïdale :

Enfin, la méthode trapézoïdale A-stable déjà citée, a été mise en oeuvre.

$$y_{n+1} = y_n + \frac{h}{2} \{f(y_n, t_n) + f(y_{n+1}, t_{n+1})\}$$

Comme pour toutes les méthodes implicites, on doit résoudre à chaque pas un système d'équations algébriques en y_{n+1}, généralement non-linéaire si le système différentiel est non-linéaire.

$$G_{n+1}(y_{n+1}) = y_n + \frac{h}{2} f(y_n, t_n) + \frac{h}{2} f(y_{n+1}, t_{n+1}) - y_{n+1} = 0$$

que l'on peut résoudre par la méthode de Newton-Raphson.
Le résultat obtenu par Dahlquist (A-stabilité, plus petite erreur de troncature), suggère que la méthode trapézoïdale est la mieux adaptée à la résolution des systèmes stiffs.

Cependant, appliquons cette méthode à l'équation test $y' = \lambda y$, λ constante complexe. On obtient :

$$y_n = \left\{\frac{1 + \lambda h/2}{1 - \lambda h/2}\right\}^n y_o.$$

Supposons $Re(\lambda h) \to -\infty$, alors $\left\{\frac{1 + \lambda h/2}{1 - \lambda h/2}\right\} \to -1$

Ceci signifie que pour des valeurs propres dont la partie réelle sera $\ll o$, les erreurs vont se traduire par des oscillations décroissant très lentement vers la solution nominale.

De ce point de vue, elle est inférieure à la méthode implicite d'Euler dans laquelle $y_n = (1 - \lambda h)^{-1} y_{n-1}$ et $(1 - \lambda h)^{-1} \to 0$ quand $Re(\lambda h) \to -\infty$

C'est la raison essentielle pour laquelle d'autres méthodes ont dû être recherchées.

II.2.2 - Méthodes utilisées

Comme nous l'avons vu précédemment, le caractère "stiff" des équations de fonctionnement du circuit conduit à utiliser des méthodes stables et donc implicites.

D'un point de vue pratique, une méthode est intéressante si, bien entendu, elle correspond à la classe des problèmes que l'on désire traiter, mais encore si elle présente une bonne souplesse d'utilisation.

En particulier, il est nécessaire de pouvoir contrôler aisément l'erreur de troncature et de faire varier le pas d'intégration pour l'adapter à l'allure locale de la solution.

Parmi l'éventail des méthodes disponibles, cet aspect pratique nous a conduit à utiliser des formules à pas liés dont Gear a montré qu'elles étaient "quasi-stables" et dont il a donné des stratégies permettant un contrôle de l'erreur de troncature et du pas d'intégration.

Il faut dire que les recherches théoriques progressent lentement dans ce domaine étant donné sa complexité (surtout dans le cas d'un pas variable) et que les contributions les plus intéressantes sont souvent l'oeuvre d'expérimentateurs connaissant bien les caractéristiques des problèmes à traiter.

D'autre part, ces formules s'adaptent bien à la résolution d'équations algébriques et différentielles couplées :

$$F(y, y', u, t) = 0$$

Déjà, dans IMAG2 [LE FAOU 72], une formule implicite à pas liés opérant sur les systèmes différentiels mis sous la forme canonique était utilisée.

Dans IMAG3, la même formule transformée pour opérer sur les systèmes algébro-différentiels implicites est utilisée. Cette formule est d'ordre 2.

D'autre part, une méthode à ordre variable a été introduite permettant d'obtenir éventuellement une précision plus importante et en principe une meilleure adaptation du pas.

La programmation de ces méthodes est délicate et demande de gros efforts si l'on veut réaliser une bonne adaptation aux problèmes à traiter. En particulier, de nombreux essais souvent coûteux sont nécessaires pour permettre la détermination des tests de contrôle et la valeur de certains des paramètres.

II.2.2.1 - Aspect théorique

Comme nous l'avons indiqué plus haut, les méthodes à k pas liés utilisent pour le calcul de y_{i+1} les résultats obtenus au cours des pas précédents. On distingue deux approches :

(a) - la fonction à intégrer f(y(t), t) est approchée par un polynôme qui prend aux points t_{i-j} (f = o, 1, ..., k-1) les valeurs $f_{i-j} = f(t_{i-j}, y(t_{i-j}))$ calculées précédemment.

Ce polynôme peut être déterminé par ces seules conditions, auquel cas le schéma récurrent obtenu est qualifié de formule de prédiction à k pas et définit explicitement y_{i+1}.

On peut aussi imposer au polynôme d'interpolation de prendre en outre au nouveau point t_{i+1} la valeur f_{i+1} encore inconnue puisqu'elle dépend de y_{i+1}. Dans ce cas, on a une formule de correction à k pas définissant y_{i+1} de manière implicite.

Les formules :

$$y_{i+1} = y_i + (1 + \nabla/2 + 5\,\nabla^2/12 + 3\,\nabla^3/8 + \ldots)\, hf_i$$

et

$$y_{i+1} = y_i + (1 - \nabla/2 - \nabla^2/12 - \nabla^3/24 - \ldots)\, hf_{i+1}$$

où ∇ désigne l'opérateur différence régressive, se déduisent facilement à partir d'identités classiques en calcul des différences finies.

Par troncature, elles fournissent des formules d'ordre aussi élevé qu'on veut.

(b) - Au lieu d'approcher f par un polynôme, on peut approcher la solution y(t), puis remplacer y' dans y' = f(y, t) par la dérivée du polynôme d'interpolation ainsi obtenu. Cette approche a été considérée longtemps comme peu intéressante (voir par exemple Henrici p. 206 [Henr 62]) à cause de la propagation des erreurs que l'on y rencontre.

Toutefois, les formules implicites obtenues sont "quasi-stables" pour $k \leq 6$ (Gear 71) et ce sont celles-ci qui sont utilisées dans IMAG3.

Par combinaison des deux approches (a) et (b), on obtient le schéma général des méthodes à k pas liés :

$$L_k\, y_{i+1} + L_{k-1}\, y_i + \ldots + L_o\, y_{i+1-k} =$$

$$h\,(\beta_k\, f_{i+1} + \beta_{k-1}\, f_i + \ldots + \beta_o\, f_{i+1-k}).$$

Lequel est explicite ou implicite (en y_{i+1}) selon que β_k est nul ou non (on suppose $L_k \neq o$ et $/L_o/ + /\beta_o/ \neq o$).

On dit que ce schéma est d'ordre p si, pour toute fonction y(t) suffisamment dérivable, les identifications $y_i = y(t_i)$ et $f_i = y'(t_i)$ conduisent pour la différence des deux membres à une expression en $0(h^{p+1})$.

Plutôt que d'envisager des formules de prédiction ou de correction prises isolément, on peut envisager de les coupler pour constituer un schéma prédicteur-correcteur à k pas liés.

(II.8)

$$(a) \quad y_n, (o) = \sum_{i=1}^{k} (L_i\, y_{n-i} + \beta_i\, h\, y'_{n-i})$$

$$(b) \quad y_n, (m+1) = \sum_{i=1}^{k} (L_i^*\, y_{n-i} + \beta_i^*\, h\, y'_{n-i}) + \beta_o^*\, h\, f(y_n, (m))$$

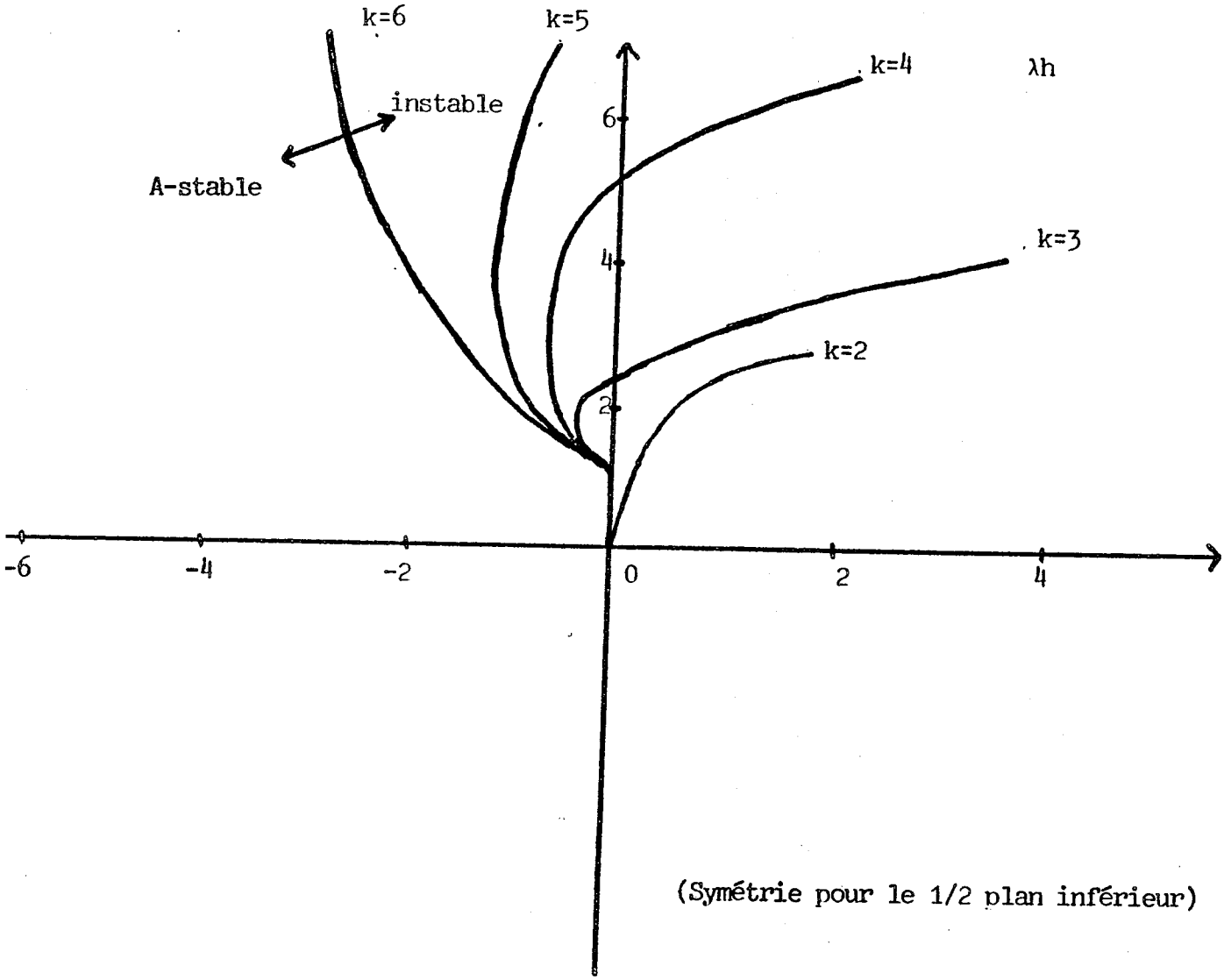
Le prédicteur (II.8.a) et le correcteur (II.8.b) ne sont pas nécessairement du même ordre. Si p est l'ordre du correcteur, on peut choisir un ordre q=p-1 pour le prédicteur de manière à avoir une bonne approximation de la solution dans y_n, (o) et minimiser le nombre d'itérations nécessaires à la convergence du correcteur.

L'itération (II.8.b), si elle donne souvent de bons résultats, ne converge généralement pas dans le cas de système stiffs non-linéaires. Dans ce cas, on doit obtenir la convergence en appliquant au correcteur la méthode de Newton-Raphson avec y_n, (o) comme valeurs initiales.

Comme nous l'avons indiqué plus haut, la "quasi-stabilité" dépend de trois paramètres D, θ et α, ainsi que de la précision requise qui dépend de l'ordre du schéma utilisé.

Ainsi, seules les méthodes d'ordre $k \leq 2$ sont "quasi-stables" pour D = o.

Dans la figure suivante, nous traçons les régions du plan complexe $h\lambda$ pour lesquelles les méthodes d'ordre $k = 2, 3, 4, 5$ et 6 sont A-stables.



(Symétrie pour le 1/2 plan inférieur)

Comme on peut le voir sur la figure, les méthodes d'ordre supérieur à 2 peuvent générer les solutions instables pour le système y' = Ay quand au moins une valeur propre de hA est située près de l'axe imaginaire et à gauche de celui-ci. Généralement, il est impossible de s'assurer que les valeurs propres de hA ne sont pas dans une région d'instabilité dans la partie gauche du plan complexe.

Cet argument milite en faveur de l'utilisation de méthodes d'ordre inférieur ou égal à 2.

Cependant, des problèmes éventuels de précision encouragent à des méthodes d'ordre supérieur. Aussi, dans IMAG3, utilisons-nous deux méthodes :

- une méthode d'ordre 2,
- une méthode à ordre variable (1 à 6).

L'ensemble de ces deux méthodes permet de faire face à la plupart des problèmes rencontrés en simulation de circuits électroniques.

II.2.2.2 - Aspects pratiques

L'utilisation des formules précédentes dans la résolution du système (II.2) :

$$F(y, y', u, t) = o$$

suppose leur adaptation à ce type de système ainsi que la résolution (ou le choix d'une solution) d'un certain nombre de problèmes auxiliaires, comme :

- initialisation des formules dont l'ordre est supérieur ou égal à 2,
- obtention de la convergence du correcteur,
- contrôle de l'erreur de troncature et du pas,
- choix de l'ordre des formules utilisées,
- choix du pas initial,
- traitement des cas où y(t) ou F sont discontinues,
- choix d'un pas minimum au-dessous duquel on ne peut poursuivre le calcul
- valeurs des constantes de tests,
- etc...

Dans la pratique, les solutions apportées à ces problèmes sont importantes et doivent être choisies en fonction de l'allure des systèmes que l'on doit effectivement résoudre.

Un mauvais choix de cet ensemble de solutions conduit généralement à un mauvais fonctionnement de la méthode qui se traduit par :

- une impossibilité de suivre la solution réelle,
- un volume de calculs prohibitifs.

Jusqu'ici, la méthode permettant de déterminer cet ensemble de solutions a été itérative. C'est-à-dire que l'on calcule effectivement un grand nombre d'exemples que l'on suppose représentatifs à l'aide d'un ensemble de solutions à priori S_o. Puis, à l'examen des résultats obtenus, on tente d'améliorer S_o pour obtenir S_1, etc...

Après n améliorations, on obtient un ensemble S_n supposé optimal pour la classe des problèmes que l'on veut résoudre.

Il est bien évident que cette approche est coûteuse étant donné le grand nombre d'essais qu'il faut effectuer pour parvenir à l'ensemble S_n.

D'autre part, cette démarche est peu rigoureuse et laisse une grande place à l'intuition et à "l'expérience".

Toutes les méthodes numériques utilisées dans IMAG3 ont été mises au point de cette manière. Pour l'intégration du système (II.2), l'utilisateur a le choix entre deux méthodes :

- une méthode prédicteur-correcteur d'ordre deux
- une méthode prédicteur-correcteur d'ordre variable (1 à 6).

Nous examinerons successivement la mise en oeuvre de ces deux méthodes et dans une dernière partie, nous exposerons quelques problèmes communs.

Nous commençons par l'adaptation des formules au système (II.2).

II.2.2.2.1 - Utilisation de formules à pas liés pour résoudre (II.2).

Considérons la formule à pas liés (correcteur) :

$$L_o\, y_n + \ldots + L_k\, y_{n-k} + h\,\beta_o\, y'_n = o \qquad \text{(II.9)}$$

On peut exprimer y'_n en fonction de y_n

$$y'_n = -\frac{L_o}{h\beta_o}\, y_n + \Sigma_n$$

où Σ_n est une combinaison linéaire des y_{n-i} supposés calculés au cours des pas précédents.

Ce qui permet d'écrire (II.2) pour $t = t_n$ sous la forme

(II.10)
$$F(y_n, y'_n, u_n, t_n) = F\left(y_n, -\frac{L_o}{h\beta_o}\, y_n + \Sigma_n, u_n, t_n\right) = o$$

On obtient ainsi à chaque pas un système d'équations algébriques en y_n et u_n.

La méthode la plus appropriée pour résoudre ce système algébrique semble être, dans le cas des systèmes stiffs, celle de Newton-Raphson.

L'itération de Newton-Raphson s'écrit :

$$(II.11)\qquad \begin{vmatrix} u_n^{m+1} \\ y_n^{m+1} \end{vmatrix} = \begin{vmatrix} u_n^{m} \\ y_n^{m} \end{vmatrix} - \left[\frac{\partial F}{\partial u}, \frac{\partial F}{\partial y} - \frac{L_o}{h\beta_o} \frac{\partial F}{\partial y'} \right]^{-1}_{m\ n} F(y_n^m, -\frac{L_o}{h\beta_o} y_n^m + \Sigma_n, u_n, t$$

où m est le compteur d'itérations.

On sait que la convergence de la méthode de Newton-Raphson exige une valeur initiale $y_n^{(o)}$ dans un voisinage de la solution y_n^+ .

Aussi, nous déterminons $y_n^{(o)}$ par une formule explicite à pas liés (prédicteur).

$$y_n^{(o)} = -\frac{1}{h} \sum_{i=1}^{k+1} \gamma_i\, y_{n-i} \qquad (II.12)$$

Remarques :

Les problèmes rencontrés lors du calcul des états stables existent également dans la résolution du système (II.10), mais cependant, de manière moins cruciale dans la mesure où l'initialisation (II.12) fournit une bonne approximation de la solution.

L'approche ci-dessus demande pour chaque itération de Newton-Raphson la résolution du système linéaire.

$$(II.13)\qquad \left| \frac{\partial F}{\partial u}, \frac{\partial F}{\partial y} - \frac{Lo}{h\beta_o} \frac{\partial F}{\partial y'} \right|_{m\ n} \left| \frac{\Delta u}{\Delta x} \right|_{m\ n} = F(y, -\frac{Lo}{h\beta_o} y + \Sigma_n, u, t)_{n\ n}$$

Comme nous l'avons vu dans le chapitre I, le code de résolution de (II.13) est obtenu comme un invariant du circuit, et on peut en voir ici tout l'intérêt.

Dans les algorithmes de calcul du régime continu, le système linéaire à résoudre pour chaque itération est de la forme (II.14).

$$(II.14)\qquad \left| \frac{\partial F}{\partial u}, \frac{\partial F}{\partial y} \right|_{m} \left| \frac{\Delta u}{\Delta x} \right| = F(y, o, u, t)$$

Il n'y a, a priori, aucune raison que la suite d'instructions permettant de résoudre (II.14) avec une précision suffisante le permette également pour (II.13).

En fait, les essais que nous avons effectués nous ont montré qu'avec les équations E_c que nous avons adoptées, ce n'était pas le cas.

Aussi, nous effectuons la détermination de cette séquence pour chaque régime d'analyse et - comme nous le verrons plus loin - pour chaque méthode d'intégration utilisée.

II.2.2.2.2 - Méthode d'ordre 2

C'est une adaptation au système (II.2) d'une méthode exposée par shichman [SHIC 70] et utilisée dans IMAG2 [LE FAOU 72] pour des systèmes différentiels mis sous la forme canonique.

Dans le cas d'un pas constant, le schéma prédicteur-correcteur prend la forme :

$$\text{(II.15)} \quad \begin{cases} y_n^{(o)} = y_{n-3} - 3 \times y_{n-2} + 3 \times y_{n-1} \\ y_n^{(o)} = -\frac{1}{3} y_{n-2} + \frac{4}{3} y_{n-1} + \frac{2}{3} h\, y'_n \end{cases} \quad \text{(II.15)}$$

Le schéma (II.15) est "quasi-stable".

Si le pas h est supposé variable au cours du processus, (II.15) devient :

- Posons $h_1 = t_n - t_{n-1}$

$h_2 = t_{n-1} - t_{n-2}$

$h_3 = t_{n-2} - t_{n-3}$

$$\text{(II.16)} \begin{cases} y_n^{(o)} = \dfrac{h_1 (h_1 + h_2)}{h_3 (h_2 + h_3)} y_{n-3} - \dfrac{h_1 (h_1 + h_2 + h_3)}{h_2 \cdot h_3} y_{n-2} \\ \qquad + \dfrac{(h_1 + h_2)(h_1 + h_2 + h_3)}{h_2 (h_2 + h_3)} y_{n-1} \\ y_n = -\dfrac{h_1^2}{h_2 (2h_1 + h_2)} y_{n-2} + \dfrac{(h_1 + h_2)^2}{h_2 (2h_1 + h_2)} y_{n-1} + \dfrac{h_1 (h_1 + h_2)}{2h_1 + h_2} y'_n \end{cases}$$

On remarque que les seconds membres de (II.16) sont des polynômes du second degré $g_1(t)$ et $g_2(t)$ respectivement. $g_1(t)$ satisfait :

$$g_1(t_{n-3}) = y_{n-3}$$

$$g_1(t_{n-2}) = y_{n-2}$$

$$g_1(t_{n-1}) = y_{n-1}$$

et $g_2(t)$ satisfait :

$$g_2(t_{n-2}) = y_{n-2}$$

$$g_2(t_{n-1}) = y_{n-1}$$

$$g_2'(t_n) = y_n'$$

Ainsi, (II.16) est obtenu en posant :

$$y_n^{(o)} = g_1(t_n) \quad \text{et} \quad y_n^{(m)} = g_2(t_n).$$

Reportant la deuxième équation de (II.16) (correcteur) résolue par rapport à y_n' dans (II.2), on obtient l'équation algébrique en y_n et u_n :

$$(II.17) \quad \left\{ F(y_n, y_n', u_n, t_n) = F\left\{y_n, \frac{2h_1 + h_2}{h_1 (h_1 + h_2)} \left(y_n + \frac{h_1^2}{h_2 (2h_1 + h_2)} y_{n-2}\right.\right.\right.$$

$$\left.\left.\left. - \frac{(h_1 + h_2)^2}{h_2 (2h_1 + h_2)} y_{n-1}\right), u_n, t_n\right\} = 0$$

Si l'on résoud (II.17) par la méthode de Newton-Raphson, on a :

$$(II.18)\quad \begin{cases} \dfrac{\partial F}{\partial (u,y)} = \left[\dfrac{\partial F}{\partial u} \,,\ \dfrac{\partial F}{\partial y} + \dfrac{2h_1 + h_2}{h_1 (h_1 + h_2)} \dfrac{\partial F}{\partial y'} \right] \\ \begin{pmatrix} u_n^{m+1} \\ y_n^{m+1} \end{pmatrix} = \begin{pmatrix} u_n^{m} \\ y_n^{m} \end{pmatrix} - \dfrac{\partial F}{\partial (u,\ y)_{m,n}} F_n^m \end{cases}$$

La valeur initiale $\begin{pmatrix} u_n^o \\ y_n^o \end{pmatrix}$ dans (II.18) est obtenue par le prédicteur de (II.16) pour les variables y et $u_n^o = u_{n-1}$, où u_{n-1} est la valeur de u calculé au cours du pas précédent.

Remarque : On pourrait calculer $u_n^{(o)}$ par une technique analogue à celle utilisée pour le calcul de $y_n^{(o)}$. Cependant, l'expérience montre que les variables u interviennent la plupart du temps linéairement dans (II.2). Ou, si ce n'est pas le cas, elles ne posent pas de problèmes de convergence au cours de l'itération (II.18).
Deux problèmes sont importants dans cette méthode :

(a) - la convergence de (II.18)

(b) - le contrôle de l'erreur de troncature locale et du pas.

La méthode est d'ordre 2 si l'on suppose que (II.18) est itérée jusqu'à sa convergence. Aussi, il importe de ne pas arrêter l'itération trop tôt, sinon, il y a risque d'accumulation d'erreurs dans les y_i successifs calculés sans que celles-ci puissent être détectées par le mécanisme de contrôle de l'erreur de troncature. D'un autre côté, permettre un trop grand nombre d'itérations peut conduire à des calculs inutiles.
Ecrivons (II.17) sous la forme :

$$(II.17.bis)\quad F(X_n) = o \qquad \text{ou } X_n = \begin{pmatrix} u_n \\ y_n \end{pmatrix}$$

Le nombre maximum d'itérations (II.18) pour résoudre (II.17.bis) est fixé à 3. Cette valeur a été déterminée empiriquement à partir des résultats de nombreux essais. On constate que la convergence s'obtient généralement en 1 ou 2 itérations. Si ce n'est pas le cas, on se trouve sûrement dans une région fortement non-linéaire auquel cas, le pas devra être réduit.

La convergence est détectée dans les deux situations suivantes :

$$//F// \leq 10^{-9}$$

$$\text{Max}_i \frac{/X_i^{m+1} - X_i^m/}{/X_i^m/ + k} \leq 5.10^{-4} \quad \text{et} \quad //F// \leq 10^{-5}$$

où k est une constante positive permettant de tenir compte d'éventuelles valeurs très faibles des X_i. k définit ainsi une amplitude ΔX_M pour laquelle une variation $/X_i/ < \Delta X_M$ ne serait pas significative. La valeur de k dépend de la nature de la variable X_i considérée (courant ou tension).

Si X_i est une tension, $k = 10^{-1}$ et si X_i est un courant $k = 10^{-4}$

En cas de croissance de //F//, on effectue une dichotomie sur les valeurs de X comme dans la méthode de Newton-Raphson, utilisée dans le calcul des états stables.

Si l'itération (II.18) est itérée jusqu'à sa convergence, l'erreur de troncature du correcteur (II.16) est de la forme :

$$E_n = \frac{h_1^2 (h_1 + h_2)^2}{6 (2h_1 + h_2)} y^{(3)} (t^*) \qquad \text{(II.19)}$$

où $t^* \in [t_{n-1}, t_n]$.

A chaque pas, l'erreur de troncature E_n doit satisfaire :

$/E_n/ < E_{max}$ où $E_{max} > 0$ est fixé a priori :

- soit par l'utilisateur du système,
- soit prend la valeur par défaut 0.1.

L'utilisateur peut spécifier E_{max} par une instruction du langage de commande : ERMAX = {valeur de E_{max}}.

La satisfaction de la contrainte précédente se réalise par une adaptation du pas d'intégration.

$y^{(3)}(t^*)$ est approchée par différences successives.

Si $/E_n/ > E_{max}$, les valeurs y_n et u_n obtenues sont refusées et le pas h courant est réduit d'un facteur 4.

Si $E_{min} \leq /E_n/ \leq E_{max}$, les valeurs y_n et u_n sont acceptées et le pas n'est pas modifié.

Si $/E_n/ < E_{min}$, les valeurs y_n et u_n sont acceptées et le pas est multiplié par 2. La constante E_{min} est choisie de telle manière que l'erreur de troncature reste inférieure à E_{max} lorsque le pas est doublé.

$$E_{min} = \frac{E_{max}}{10}$$

D'autre part, lorsque l'itération (II.18) ne converge pas, le pas est divisé par 4 et les valeurs y_n et u_n refusées.

Comme dans toutes les méthodes à plusieurs pas liés, se pose le problème du démarrage. Pour que le schéma (II.16) puisse fonctionner, il faut connaître au moins trois points.

$X = \begin{Bmatrix} u \\ y \end{Bmatrix}$: X_o, X_1, X_2 . X_o correspond aux conditions initiales, X_1 et X_2 sont calculés par la méthode d'Euler implicite d'ordre 1.

L'utilisateur peut fixer par une instruction du langage de commande le pas de sortie sur les courbes résultats (HS = {valeur du pas de sortie}).

D'autre part, pour minimiser le volume des calculs, le processus d'intégration devra progresser le plus souvent possible avec un pas HMAX.
La valeur initiale du pas est donc fixée à HMAX. On doit avoir HMAX $\leq$ HS.

Remarque : Par défaut, on sortira 100 points sur la courbe, c'est-à-dire

$$HS = \frac{t_{max} - t_o}{100} .$$

Les résultats obtenus par cette méthode sont bien supérieurs à ceux obtenus par la méthode trapézoïdale notamment en ce qui concerne l'amortissement des erreurs. Cependant, dans certains cas où la précision demandée est importante :

$\frac{E_n}{y} < 10^{-2}$, une méthode d'ordre 2 s'avère insuffisante.

C'est la raison pour laquelle nous avons introduit une méthode d'ordre variable (1 à 6), qui permet d'ajuster l'ordre en fonction de l'erreur de troncature permise.

II.2.2.2.3. Méthode d'ordre variable

L'avantage que l'on peut attendre d'une méthode d'ordre variable est le suivant :

- à chaque pas, on peut ajuster l'ordre de la méthode de telle manière qu'il permette d'adopter le pas maximum satisfaisant aux contraintes sur l'erreur de troncature.

Les mécanismes de contrôle du pas, de l'erreur de troncature et de l'ordre que nous utilisons, ont été développés par Gear [Gear 71].

La méthode utilise un prédicteur d'ordre p, et un correcteur d'ordre p.

$$(II.20) \quad \begin{cases} y_n^{(o)} = \tilde{L}_1 y_{n-1} + \ldots + \tilde{L}_p y_{n-p} + h \tilde{\beta}_1 y_{n-1} \\ y_n^{(m+1)} = L_1 y_{n-1} + \ldots + L_p y_{n-p} + h \beta_o y_n^{(m)} \end{cases}$$

On peut montrer que les équations (II.20) peuvent s'écrire sous forme matricielle :

$$(II.21) \quad \begin{cases} \tilde{w}_n = B.w_{n-1} \\ w_n = \tilde{w}_n + c.b \end{cases}$$

où :

$$w_n = [y_n,\ h\,y'_n,\ y_{n-1},\ \cdots,\ y_{n-p+1}]^T$$

$$\tilde{w}_n = [y_n^{(o)},\ h\,y'^{(o)}_n,\ y_{n-1},\ \cdots,\ y_{n-p+1}]^T$$

$$c = [\beta_o,\ 1,\ o,\ \ldots,\ o]^T$$

$$B = \begin{bmatrix} \tilde{L}_1 & \tilde{\beta}_1 & \tilde{L}_2 & \cdots & \tilde{L}_{p-1} & \tilde{L}_p \\ \gamma_1 & 1 & \gamma_2 & \cdots & \gamma_{p-1} & \gamma_p \\ 1 & 0 & & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 \cdots & 0 & 0 \\ \vdots & & & & & \\ 0 & & & \cdots & 1 & 0 \end{bmatrix}$$

$$b = [h\,y'_n - h\,y'^{(o)}_n]$$

Remarquons que les composantes de w_n représentent le polynôme $y_n(t)$ de degré p tel que $y_n(t_{n-i}) = y_{n-i}$ $\quad o \le i \le p$ et $y'_n(t_n) = y'_n$.

Considérons une transformation Q telle que :

$$a_n = Qw_n = \left[y_n,\ h\,y'_n,\ \frac{h^2}{2}\,y''_n,\ \ldots,\ \frac{h^p}{p!}\,y_n^{(p)}\right]^T$$, a_n représentant le même polynôme que w_n. On a :

$$\text{(II.22)} \quad \begin{cases} \tilde{a}_n = Q\tilde{w}_n = Q\,B\,Q^{-1}\,a_{n-1} = A.a_{n-1} \\ a_n = Qw_n = Q\tilde{w}_n + Qcb = \tilde{a}_n + 1b \end{cases}$$

où $A = Q\,B\,Q^{-1}$ est la matrice triangulaire de Pascal.

Remarque : Le vecteur a_n a été introduit dans ce contexte par Nordsieck. Il permet de sauvegarder l'information relative aux pas précédents en termes des dérivées successives du polynôme d'interpolation utilisé plutôt qu'en termes des valeurs de la solution. Ceci permet une plus grande souplesse de mise en oeuvre des mécanismes de contrôle (pas, ordre, erreur de troncature).

Dans (II.22), b peut être déterminé de la manière suivante :

$$\begin{aligned} o &= h\, f(y_n, t_n) - h\, y'_n \\ &= F(a_n, t_n) \\ &= F(\tilde{a}_n + 1.b, t_n) \end{aligned}$$

Cette équation peut être résolue par la méthode de Newton-Raphson à partir de la valeur initiale $b_{(o)} = o$.

$$b_{(m+1)} = b_{(m)} - [d\hat{F}(b_{(m)})/db]^{-1}\ \hat{F}(b_{(m)})$$

où :

$$\hat{F}(b) = F(\tilde{a}_n + lb, t_n)$$

Posons : $a_{n,(m)} = \tilde{a}_n + l\, b_{(m)}$

On obtient :

$$(II.23) \quad \begin{cases} a_{n,(o)} = A.a_{n-1} \\ a_{n,(m+1)} = a_{n,(m)} - l[\sum_{i=o}^{k} \frac{\partial F}{\partial a_i}\ l_i]^{-1}\ F(a_{n,(m)}, t_n) \end{cases}$$

L'équation $F(a,t) = K(y, y', t) = o$ est une équation différentielle si $\frac{\partial K}{\partial y'} \neq o$, sinon c'est une équation algébrique.

L'erreur de troncature sur un pas est de la forme :

$$C_{p+1}\ h^{p+1}\ y^{(p+1)}\ (t_n) + o\,(h^{(p+2)})$$

On prendra :

$$va_n(p) = h^{p+1} \; y_n^{(p+1)} \; / \; p!$$

Si ε est l'erreur requise, on exigera :

(II.24)
$$\varepsilon^2 \geq \sum_{i=1}^{n} \left[\frac{C_{p+1} \; h^{p+1} \; y_i^{(p+1-} \; (t_n)}{//y_i//_{t_n}} \right]^2$$

$$\simeq \sum^{n} \left[\frac{C_{p+1} \; . \; p! \;\; va_n^i \; (p)}{} \right]^2$$

où : $//y_i//_{t_n} = \text{Max} \{ \underset{t \leq t_n}{\text{Max}} \; /y_i/ , A \}$.

$A = 10^{-6}$ si y_i est un courant

$= 10^{-3}$ si y_i est une tension.

Pour choisir l'ordre et le pas, il faut estimer quel pas on pourrait utiliser à l'ordre p-1, à l'ordre p et à l'ordre p+1 (le dernier pas ayant été effectué à l'ordre p) tout en satisfaisant l'erreur demandée.

A l'ordre p, on pourra utiliser le pas h/pR2 ou

$$PR2 = 1.2 * \left\{ \sum_{i=1}^{n} \left[\frac{va_n^i(p)}{//y_i//_{t_n}} \right]^2 . \left[\frac{C_{p+1} \; . \; p \; !}{\varepsilon^2} \right]^2 \right\}^{1/2(p+1)}$$

Le facteur 1.2 force le pas à être plus petit que l'équation (II.24) le permettrait.

Si l'on utilise une méthode d'ordre p-1, on demandera :

$$\varepsilon^2 \geqslant \sum_{i=1}^{n} \left[\frac{C_p \; h^p \; y_i^{(p)} \; (t_n)}{//y_i//_{t_n}} \right]^2$$

et le facteur de modification du pas devient :

$$PR1 = 1.3 * \left\{ \sum_{i=1}^{n} \left[\frac{h^p \, y_i^{(p)} \, (t_n)}{//y_i//_{t_n}} \, \frac{C_p}{\varepsilon} \right]^2 \right\}^{1/2p}$$

Le coefficient 1.3 favorise le non-changement d'ordre.
De même, pour l'ordre (p+1),

$$\varepsilon^2 \geq \sum_{i=1}^{n} \left[\frac{h^{p+2} \, y^{(p+2)} \, (t_n)}{//y_i//_{t_n}} \, C_{p+2} \right]^2$$

$$PR3 = 1.4 * \left\{ \sum_{i=1}^{n} \left[\frac{C_{p+2} \, h^{p+2} \, y^{(p+2)}}{//y_i//_{t_n} \cdot \varepsilon} \right]^2 \right\}^{1/2(p+2)}$$

Le coefficient 1.4 favorise le choix de l'ordre p ou p-1.
Finalement, le pas h sera corrigé d'un facteur C_o ($h_N = \frac{h_{n-1}}{C_o}$) ou

$$C_o = \text{Min } [PR1, PR2, PR3].$$

et l'ordre adopté correspondra à celui qui permettra le pas le plus grand.

On constate expérimentalement que des changements trop fréquents de l'ordre et du pas, peuvent conduire à des phénomènes d'instabilité.

D'autre part, on a intérêt à choisir un ordre aussi peu élevé que possible (compatible avec l'erreur de troncature), pour augmentre la région de stabilité dans le plan des λh.

Les coefficients 1.2, 1.3 et 1.4 sont choisis pour favoriser un non-changement d'ordre et l'utilisation d'ordres peu élevés.
Des tentatives de changement de l'ordre et du pas sont effectués :

- soit quand l'erreur de troncature requise n'est pas satisfaite,
- soit quand l'erreur est satisfaite, tous les quatre pas.

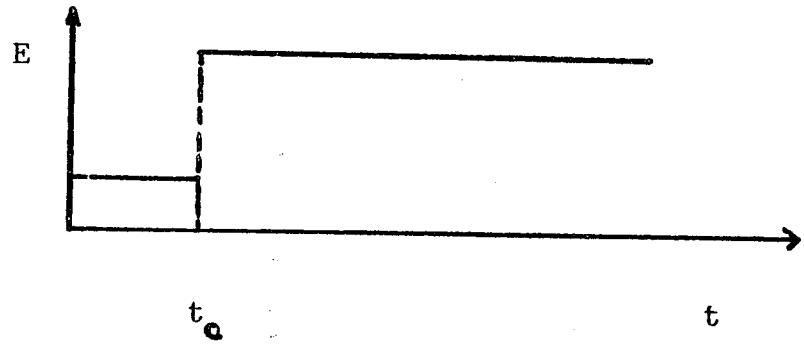
Comme dans la méthode précédente, le nombre maximum d'itérations de Newton-Raphson est fixé à 3. Dans le cas où la convergence n'est pas réalisée, le pas est réduit arbitrairement d'un facteur 4 (on ne peut utiliser le mécanisme de contrôle du pas).

II.2.2.3 - Problèmes particuliers

II.2.2.3.1 - Traitement des discontinuités

Dans certains cas, des discontinuités peuvent apparaître dans la fonction F. Un exemple simple est le suivant :

soit C un circuit dont une source d'excitation E a la forme d'un créneau comme ci-dessous :



F dépendant de E, sera discontinue pour $t = t_c$.
D'autres types de discontinuités plus complexes peuvent apparaître, notamment dans les circuits à MOS. Elles sont alors décrites par des expressions SI portant sur les variables du circuit.

Les algorithmes d'intégration supposant toutes les quantités continues, ne sont pas adaptés à ces cas particuliers. Ainsi, les algorithmes à pas liés utilisent pour calculer $y(t)$, $(t > t_c)$, des valeurs $y(t)$, $(t < t_c)$ qui n'ont plus de signification.

La méthode utilisée pour aborder ce type de problèmes est la suivante :

(1) - calcul de t_c

(2) - réinitialisation de l'algorithme à partir de $t = t_c$.

Le calcul de t_c s'effectue de deux manières distinctes :

- si la quantité qui introduit la discontinuité est donnée par une table dont l'abcisse est le temps, t_c s'obtient directement par lecture dans la table de l'abcisse des points critiques.

- dans les autres cas, il faut calculer t_c par une méthode approchée. Au voisinage de ces points, le comportement des algorithmes d'intégration est le suivant :

supposons que l'on se trouve en un point $t_i < t_c$ et tel que $t_i + h_i > t_c$.

L'erreur de troncature en $t_{i+1} = t_i + h_i$ sera grande à cause de la discontinuité en t_i et le pas sera refusé.
Un nouveau point sera recalculé en $t_{i_1} = t_i + h_{i_1} < t_c$.

L'erreur de troncature étant très faible, le pas sera augmenté et le point $t_{i_2} = t_{i_1} + h_{i_2} > t_c$ sera calculé. Ce point sera à nouveau refusé, etc...

Ce processus continuera jusqu'à ce que l'on arrive dans une zone de flou numérique au voisinage de t_c où le point accepté sera $t_{i_k} = t_c \pm \varepsilon$.

Si on laisse ce processus évoluer de lui-même, il peut être coûteux par le nombre de pas qu'il exigera. Aussi, on a intérêt à le diriger, par exemple, par une recherche dichotomique.

On se rapproche ainsi de t_c avec une succession de pas h_i, $\frac{h_i}{2}$, $\frac{h_i}{4}$, ... jusqu'à ce que l'on se trouve dans un voisinage adéquat du point t_c à partir duquel la méthode est réinitialisée.
Cette stratégie donne généralement de bons résultats.

Remarque : il est bien évident qu'une approche plus rigoureuse à ce problème consisterait à détecter les changements d'occurrences dans les expressions SI.

II.2.2.3.2 - Influence de la séquence codée $L_{L_{Di}}$ sur l'itération (II.11).

Ecrivons à nouveau l'itération (II.11) :

$$\begin{pmatrix} u_n^{m+1} \\ y_n^{m+1} \end{pmatrix} = \begin{pmatrix} u_n^m \\ y_n^m \end{pmatrix} - \left(\frac{\partial F}{\partial u} , \frac{\partial F}{\partial y} , - \frac{L_o}{h\beta_o} \frac{\partial F}{\partial y'} \right)^{-1}_{m \atop n} F(y_n^m, - \frac{L_o}{h\beta_o} y_n^m + \Sigma_n, u_n, t_n$$

application de la méthode de Newton-Raphson au système d'équations algébriques induit par le correcteur.

Comme nous l'avons vu dans le premier chapitre, une séquence codée $I_{L_{Di}}$ permet de résoudre au cours du processus d'intégration, le système linéaire (II.13).

$$\left(\frac{\partial F}{\partial u}, \frac{\partial F}{\partial y} - \frac{L_o}{h\beta_o} \frac{\partial F}{\partial y'} \right)_{\substack{m \\ n}} \begin{pmatrix} \Delta u \\ \Delta x \end{pmatrix} = F(y, -\frac{L_o}{h\beta_o} y + \Sigma_n, u, t)_{\substack{m \\ n}}$$

Les coefficients L_o et β_o dépendent de l'ordre p de la méthode.

La séquence $I_{L_{Di}}$ - comme on peut le constater sur l'algorithme qui permet sa détermination - va dépendre de u, y et y', ainsi que du pas h et de l'ordre p.

$$(II.25) \qquad I_{L_{Di}} = \phi\ (y, u, y', h, p, t)$$

En toute rigueur, à chaque variation de ϕ doit correspondre la détermination d'une nouvelle séquence $I_{L_{Di}}$ (changement de domaine de validité de la séquence).

Bien entendu, cette contrainte est inacceptable pour la rapidité de simulation et on espère qu'un nombre minimum de changements de séquence interviendra au cours de l'intégration.

La détermination de la séquence initiale $I_{L_{Di}}$ s'effectue à partir des conditions initiales du problème différentiel ainsi que du pas h_o et de l'ordre p_o de démarrage de la méthode.

L'expérience montre que cette séquence $I_{L_{Di}}$ couvre généralement tout le processus d'intégration.

La raison semble tenir à la conjonction de deux facteurs :

- si la matrice du système linéaire n'est pas trop mal conditionnée, l'ensemble des pivots choisis dans la factorisation L/U n'a pas une importance considérable sur le résultat final.

- dans la mesure où le prédicteur initialise les variables y et u dans le domaine de convergence de la méthode de Newton-Raphson, de légères erreurs effectuées dans la résolution de (II.13) peuvent éventuellement conduire à des itérations supplémentaires, mais rarement à une non-convergence.

La séquence $I_{L_{Di}}$ dépendant de h et de p, et dans la méthode à ordre variable, les variations du pas pouvant être plus importantes, on doit avoir un changement plus fréquent des séquences $I_{L_{Di}}$ dans cette dernière méthode. C'est effectivement ce que l'on constate expérimentalement.

Signalons pour terminer que le test introduit pour détecter les changements de domaines $\left(/\frac{J_{i,j}}{J_{ii}}/ < K\right)$ peut ou non être effectué au cours de la simulation.

Une instruction du langage de commande :

TOPIVO = {valeur de K}

permet de préciser la valeur de K. Dans le cas où cette valeur n'est pas précisée, le test précédent n'est pas effectué. La séquence $I_{L_{Do}}$ est alors utilisée tout au long du calcul, sauf si $J_{ii} = o$.

II.2.2.3.3 - Utilisation des différents paramètres des méthodes d'intégration, vue sous l'aspect utilisation.

Les paramètres qui peuvent être précisés par l'utilisateur, sont les suivants :

- HMAX : pas maximum de progression de la méthode,
- HS : pas de sortie des résultats,
- HMIN : pas minimum de progression,
- ERMAX : erreur de troncature,
- MAXDER : ordre maximum de la méthode.

Dans les cas difficiles, l'utilisateur peut fixer ces paramètres de manière à obtenir des résultats corrects.

Le paramètre HMIN permet, compte-tenu de la nature du circuit et de contraintes économiques, de préciser un nombre de pas maximum - c'est-à-dire un coût du calcul à ne pas dépasser -. Il semble - par expérience - qu'il ne doive pas être choisi inférieur à (TMAX-TO) $/10^7$, car les erreurs d'arrondis deviennent alors importantes.

- HMAX, dans le cas de circuits prenant en défaut la stabilité de la méthode, permet une limitation à priori du pas qui peut la maintenir dans son domaine de stabilité. D'autre part, un HMAX faible permet de limiter l'erreur de troncature et donc de donner dans certains cas une stabilité. En effet, si l'on travaille avec un pas tel que λh est peu supérieur au rayon de stabilité, l'amplification des erreurs est faible et si les erreurs sont elles-mêmes faibles, on parvient à des résultats corrects. (En ce sens, il joue un rôle analogue au paramètre ERMAX).

- Le paramètre ERMAX limite l'erreur de troncature. Cependant, comme l'électronicien ne demande pas une très grande précision, il est surtout utile dans les problèmes de stabilité comme nous l'avons vu ci-dessus. Dans cette utilisation, il est à la fois :

  - plus intéressant que le paramètre HMAX, car il ne limite pas comme lui le pas de la méthode, tout au long du calcul.

  - moins intéressant, car il intervient par l'intermédiaire du mécanisme de contrôle de l'erreur de troncature, et du pas qui peut lui-même être pris en défaut.

Aussi, dans les problèmes difficiles, l'approche que nous préconisons est la suivante :

(1) - limitation de l'erreur de troncature par diminution du paramètre ERMAX.

(2) - si après (1), les résultats ne deviennent pas satisfaisants, on diminue le pas maximum autorisé HMAX.

Dans le cas de la méthode à ordre variable, le paramètre MAXDER permet de spécifier un ordre compris entre 1 et 6 que la méthode ne pourra dépasser. Par défaut, ce paramètre est fixé à 6 par le système. Dans la plupart des cas, avec une erreur de troncature de l'ordre de 0.1 à 0.2, la méthode n'utilise pratiquement jamais des ordres supérieurs à 4.

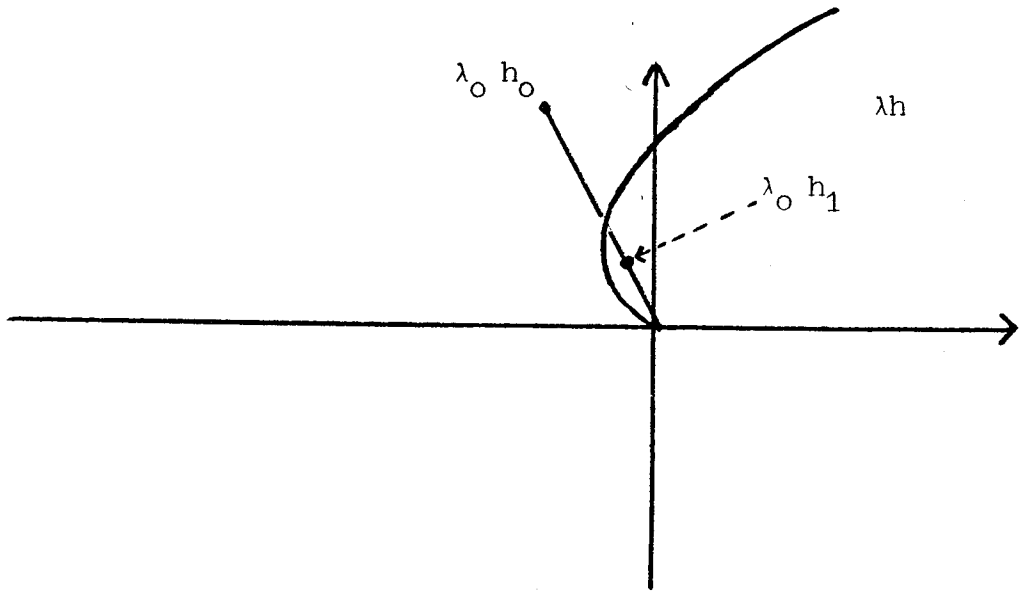
Cependant, il est quelquefois nécessaire de réduire cet ordre à 2 ou 3 de manière à augmenter la région de stabilité.

Remarque : Les modifications précédentes s'accompagnent généralement d'une modification du paramètre HMIN dans le sens d'une diminution du pas minimum acceptable compatible avec les nouvelles valeurs de HMAX ou de ERMAX.

Cependant, cette diminution s'accompagne de deux difficultés dont on doit tenir compte :

(a) - si l'on travaille avec un pas trop faible, les erreurs d'arrondi peuvent devenir importantes (supérieures à l'erreur de troncature) rendant illusoire l'utilisation d'un tel pas.

(b) - reprenons la courbe déterminant les régions de stabilité dans le plan des λh pour une méthode stiffly-stable et un ordre donné Q.



Supposons que l'on travaille au point $(\lambda_o h_o)$ du plan complexe où l'on se trouve dans une région de stabilité.

Réduire le pas à la valeur h_1 peut conduire - comme sur la figure - à pénétrer dans une région d'instabilité.

Une approche pour aborder cette difficulté nous semble être la suivante :

- fixer un pas HMIN tel que l'on ne pénètre pas dans une région d'instabilité.

Cependant, si la méthode tente d'utiliser un pas $h < HMIN$, ceci signifie que l'erreur de troncature pour $h = HMIN$ est trop importante. On est donc conduit à permettre des erreurs de troncature importantes pour maintenir la méthode dans une région de stabilité.

D'autre part, quand l'on progresse au pas h = HMIN, on doit abandonner les mécanismes de contrôle de l'erreur de troncature et du pas puisque celle-ci n'est pas satisfaite.

Ainsi, tant que l'erreur de troncature n'est pas satisfaite, on progresse avec le pas h = HMIN sans tenir compte de cette erreur. Puis, quand elle est à nouveau satisfaite, on s'autorise à augmenter le pas.

Cette approche nous a permis de traiter certains cas difficiles, comme des circuits comportant des diodes ZENER. Cependant, étant donné sa difficulté d'utilisation (détermination de HMIN), elle n'a pas été mise en oeuvre dans la version opérationnelle d'IMAG3.

En réalité, une détermination correcte de ces paramètres pour un cas particulier reste une affaire d'expérience qui s'exerce sur les résultats d'échecs précédents. Ceci est dû à des inconnues qui subsistent dans le problème (évolution des valeurs propres au cours de l'intégration, etc...), ainsi que l'influence de facteurs comme la modification du pas d'intégration, les erreurs d'arrondi$, comportement des méthodes en présence de solutions multiples pour les équations algébriques, qui demeurent mal connues.

Malgré tout, et bien que le problème semble à priori plus difficile que le calcul des états stables, nous possédons avec les deux méthodes mises en oeuvre dans IMAG3, des algorithmes permettant de faire face à la plupart des situations rencontrées en simulation de circuits électroniques (si toutefois les paramètres sont correctement déterminés).

Ces deux méthodes possèdent les deux propriétés essentielles suivantes :

- stabilité nécessaire au calcul des circuits électroniques réels,
- elles permettent d'exploiter le caractère creux de la matrice de Jacobi du système.

Il ne semble pas que depuis la réalisation d'IMAG3 soient apparues d'autres méthodes qui permettraient un progrès décisif par rapport à celles que nous utilisons. Toutefois, la conception modulaire d'IMAG3 permet d'intégrer aisément des méthodes qui se révèleraient intéressantes.

Nous verrons dans le chapitre III des exemples montrant les progrès réalisés grâce à l'utilisation conjointe de ces méthodes et des invariants introduits au chapitre I.

III - "Conception automatique" ou optimisation de circuits électroniques

Une des questions essentielles en C.A.O. est la suivante :

- jusqu'à quel point un ordinateur peut-il remplacer l'intuition humaine ?

Généralement, un concepteur de circuit désire construire un circuit répondant à certains critères appelés critères de conception.

exemple : la tension entre deux points donnés du circuit devra être de 3 volts quand celui-ci aura atteint un état stable.

Pour satisfaire à ces critères, le concepteur peut, avec l'aide de l'ordinateur (simulation), modifier :

(1) - la valeur des paramètres de conception (résistances, coefficients de couplages, paramètres physiques,...),

(2) - la structure du circuit elle-même.

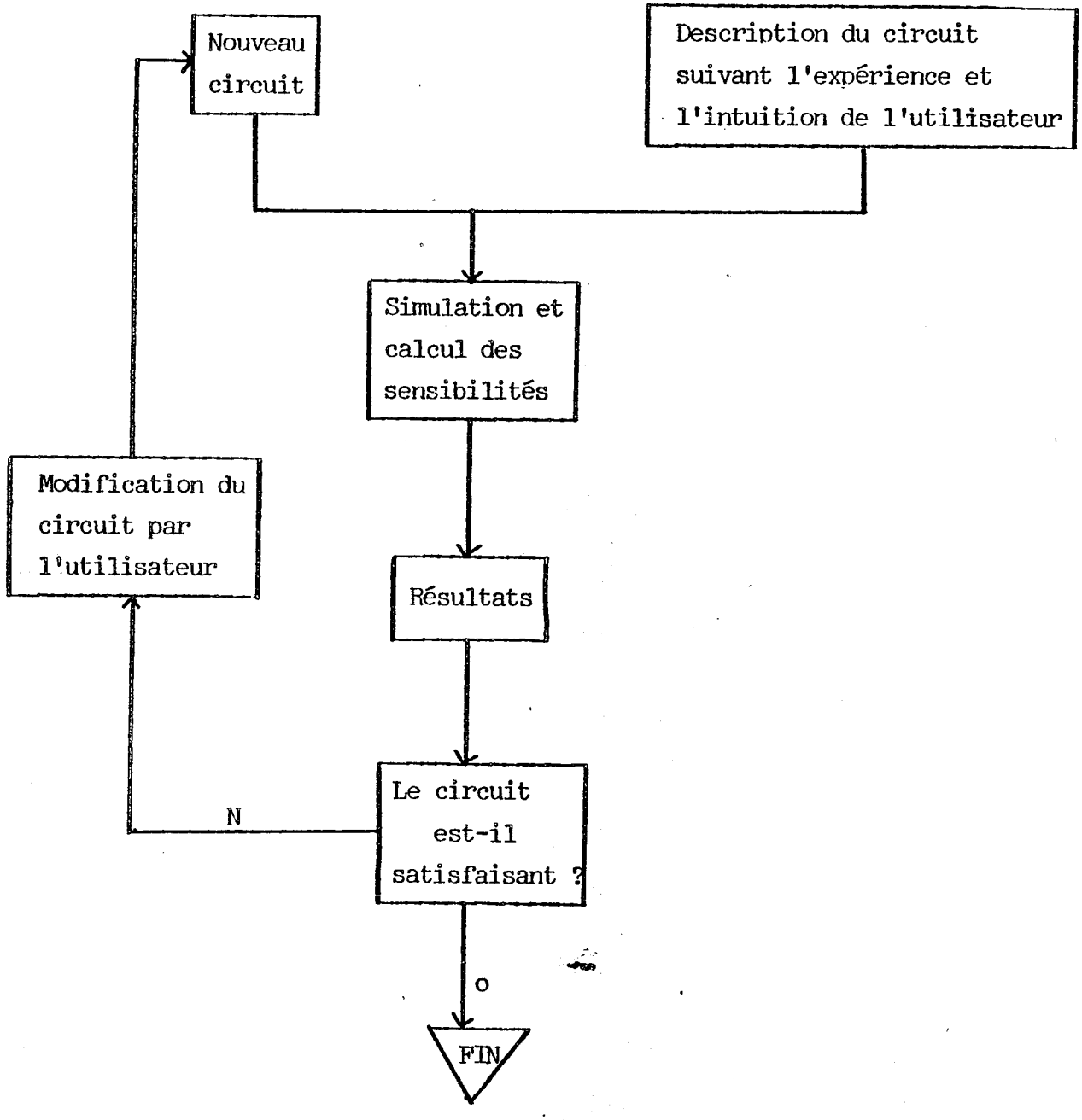
Ces deux corrections exigent évidemment un degré croissant "d'intelligence".

Si nous voulons développer une méthodologie tendant à automatiser ces corrections, c'est-à-dire compléter l'intuition par des algorithmes, il paraît raisonnable de se placer d'abord dans la situation n° 1.

Le premier pas effectué dans IMAG3, dans le but de guider le concepteur dans la recherche d'un circuit satisfaisant, a été le calcul des sensibilités (calcul de la dérivée partielle de n'importe quelle variable du circuit par rapport à n'importe quel paramètre constant dans la description).

La connaissance des sensibilités permet à l'utilisateur de déterminer l'influence de ces paramètres sur le comportement du circuit, et ainsi, lui sert de guide dans l'élaboration du circuit final.

Ce processus de conception peut se schématiser de la façon suivante :



Le but de la "conception automatique" est de remplacer la partie "modification du circuit par l'utilisateur" par un processus algorithmique permettant d'effectuer cette modification de manière automatique.

Dans IMAG2, un langage d'entrée, ainsi que des algorithmes permettant de faire de l'optimisation de circuits ont été mis en oeuvre [CAR 73].

Dans IMAG3, nous avons conservé le langage d'entrée ainsi que les algorithmes de minimisation. Nous avons repris et généralisé les algorithmes de calcul de gradient nécessaires à la minimisation de manière à les adapter à la formulation adoptée dans IMAG3.

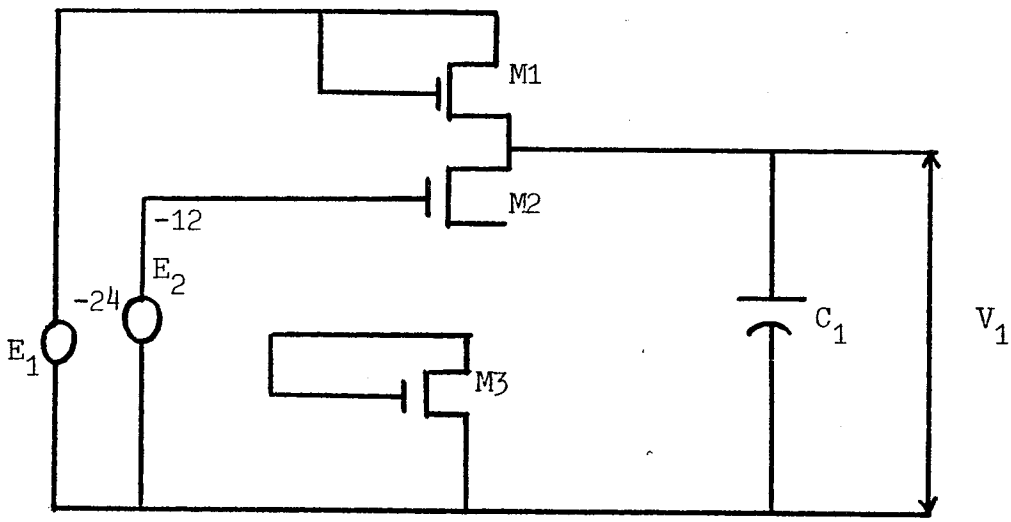
Dans cette partie, nous formulerons d'abord le problème de l'optimisation à la fois d'un point de vue mathématique et d'un point de vue pratique.

Ensuite, nous aborderons la définition du langage d'accès et enfin nous exposerons l'aspect interne du système d'optimisation.

III.1 - Position du problème

III.1.1 - Aspect pratique

Considérons le circuit suivant et intéressons-nous à son fonctionnement en régime stabilisé :



M1, M2 et M3 sont des transistors MOS.

Le concepteur désire que la tension de sortie V1 du circuit prenne comme valeur V1 = -12 V. Pour ce faire, il a la possibilité de modifier la largeur de diffusion de la grille du transistor M3 : P_1.

Une première approche pour résoudre ce problème consiste à donner une valeur initiale pour P_1 : P_{1_o}, de faire une simulation et un calcul de sensibilités : $\frac{\partial V1}{\partial P_{1_o}}$.

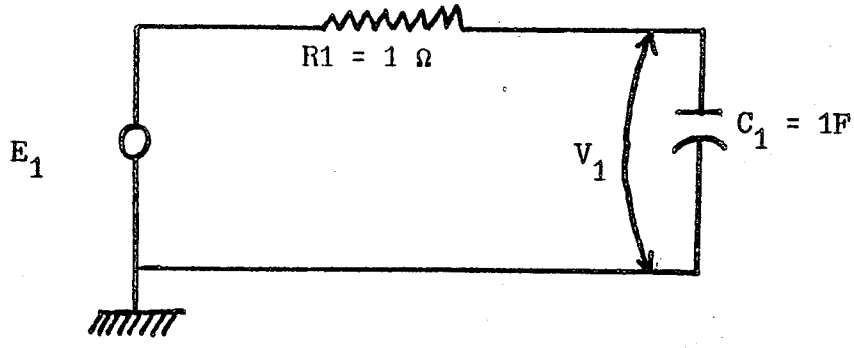
Au vu des résultats, le concepteur peut fixer une nouvelle valeur P_{1_1}, etc... Itérativement, il parviendra (si cela est possible), à une valeur P_{1_f} telle que : $V_1 = -12$ V.

Dans certains cas, cette approche peut donner de bons résultats, mais elle reste cependant fastidieuse et devient impraticable si plusieurs paramètres doivent être modifiés.

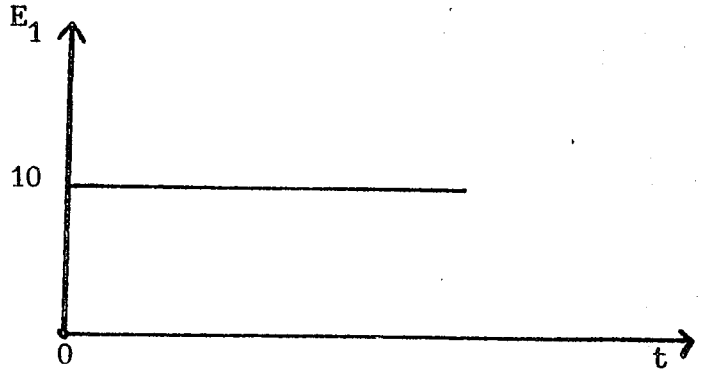
V_1 est supposé être une fonction de P_1 à travers le fonctionnement du circuit. Dans le cas où V_1 serait indépendant de P_1, on ne pourrait évidemment pas résoudre le problème posé. Une manière agréable pour le concepteur de circuit de formuler son problème à un système, peut être :

trouver P_1 tel que $V_1 + 12 = 0$.

Considérons un autre exemple simple se rapportant à une étude en régime transitoire.
Soit le circuit suivant :



où E_1 est de la forme :



On désire que V_1 atteigne 9V au bout de 5 s. Avec les valeurs des composants marqués sur le schéma, V1 atteint 9V au bout de 2.4 s.
Pour atteindre notre objectif, on se permet de modifier la valeur de la capacité C_1.
Le problème peut se formuler de la manière suivante :
calculer C_1 tel que, quand $V_1 = 9$, alors, $t = 5$.

On peut remarquer que ce problème est plus complexe que le précédent car il s'y introduit la variable indépendante temps.

Le but d'un système d'optimisation de circuits électroniques est de permettre à l'utilisateur de formuler de tels problèmes et de lui en fournir la solution.

Nous examinons maintenant comment ce type de problèmes peut s'exprimer sous forme mathématique dans le but d'élaborer des algorithmes permettant de les résoudre.

III.1.2 - Aspect mathématique

Soit p, un vecteur dont chaque composante représente un paramètre de conception, et soit ϕ une fonction de p que nous appellerons fonction objectif.

On définit :

$$\phi : R^p \rightarrow [0, +\infty[$$

Supposons que ϕ possède la propriété suivante :

- elle est minimum pour la valeur de p correspondant à la satisfaction des critères de conception.

Le problème de conception automatique formulé par l'utilisateur du système, peut ainsi l'être de la manière suivante :

- trouver p tel que ϕ (p) soit minimum.

On retrouve là un problème d'optimisation mathématique - recherche du minimum d'une fonction de plusieurs variables - qui peut s'aborder de manière algorithmique.

Exemples : dans le premier exemple du paragraphe précédent, la fonction ϕ peut prendre la forme :

$$\phi = (V1 + 12)^2$$

Plus généralement, on s'intéresse à des fonctions objectifs de la forme :

$$\phi(p) = \sum_{i=1}^{q} \{\varphi_i(p) - \tilde{\varphi}_i\}^{2k} \qquad (II.26)$$

où $\tilde{\varphi}_i$ est la valeur souhaitée pour $\varphi_i(p)$.

La formulation du deuxième exemple est plus complexe.

L'état instantané d'un circuit est caractérisé par la valeur des variables {y, u} obéissant aux équations de fonctionnement $F(y, y', u, p, t) = o$. où p désigne le vecteur des paramètres de conception.

On supposera que les composantes de p gardent des valeurs constantes au cours de la réponse transitoire du circuit.

C'est-à-dire : $\frac{dp}{dt} = o \qquad (II.27)$

Considérons un état initial (y_o, u_o) d'un circuit tel que $y'_o = o$ et un état atteint par ce circuit après un temps T à partir de (y_o, u_o).

Ce temps T est supposé défini par une relation

(II.28) $\qquad S(T(p), y(p,T), u(p,T), p) = o$

La fonction S sera dite <u>fonction de seuil</u>.

Cette dénomination représente bien l'utilisation qui est faite de la fonction S dans le système IMAG3. En effet, elle est utilisée pour déterminer le franchissement d'un seuil pour le circuit. Négative initialement, elle s'annule au passage du seuil, puis devient positive. Ceci peut correspondre à un circuit logique atteignant son état 1 par exemple.

Dans la littérature, une telle fonction S est généralement rencontrée sous le nom de <u>fonction cible</u>.

Dans le deuxième exemple du paragraphe précédent, la relation (II.28) prend la forme :

$$V1 - 9 = o$$

tandis que la fonction objectif peut s'écrire :

$$\phi = (T-5)^2$$

Plus généralement, on s'intéresse aux fonctions objectifs définies par une intégrale :

$$\phi = \int_o^{T(p)} \{\varphi(p,t) - \tilde{\varphi}(t)\}^{2k} \, dt \qquad \text{(II.29)}$$

Avec ce formalisme, on peut exprimer la fonction objectif précédente, par :

$$\phi = \left[\int_o^T dt - 5\right]$$

Le formalisme (II.29) permet à l'utilisateur de faire face à la plupart des situations pratiques. En particulier, il permet d'ajuster une réponse sur une réponse donnée.

Remarques :

(a) - Dans IMAG2, on ne considère pas le formalisme général (II.29), mais le cas particulier :

$$\phi = \{\varphi(p,T) - \tilde{\varphi}(T)\}^{2k}$$

(b) - Le problème ainsi posé :

minimiser (II.29) avec la contrainte (II.28) à la limite, y et u satisfaisant le système algébro-différentiel

$$F(y, y', u, p, t) = o \quad \text{et tel que} \quad \frac{dp}{dt} = o$$

est un cas particulier d'un problème connu en calcul des variations [BLISS 46] sous le nom de problème de LAGRANGE. Dans ce problème, on ne suppose pas :

$$\frac{dp}{dt} = o.$$

Généralement, il sera nécessaire d'introduire un certain nombre de conditions supplémentaires. Par exemple, limiter le domaine des valeurs possibles pour les paramètres de conception p ou certaines quantités physiques du circuit (puissance dissipée, tension, etc...).

Ces conditions seront exprimées sous la forme de contraintes :

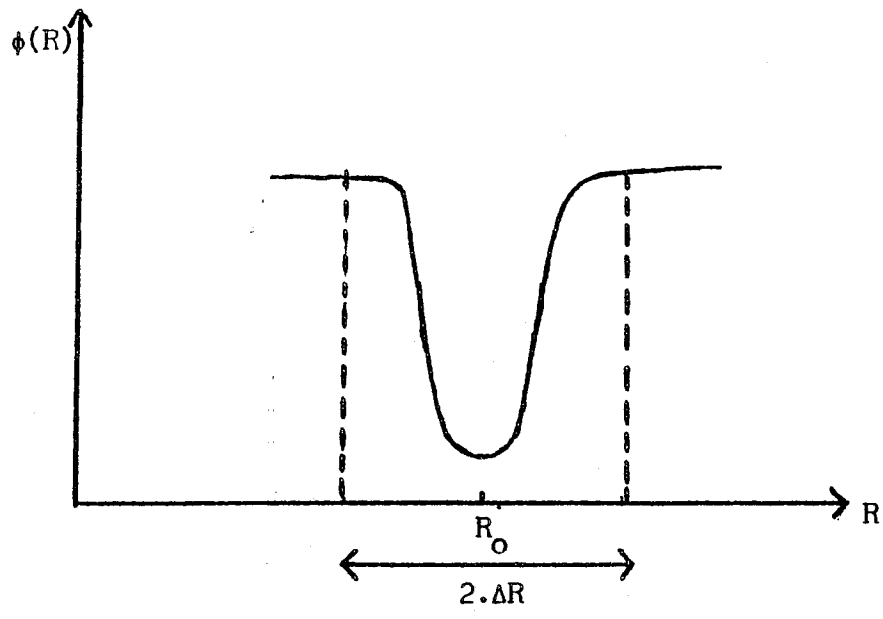
$$a_i < g_i(p) < b_i$$

Les a_i et b_i étant des constantes.

II.1.3 - Optimisation et analyse de tolérance

Comme nous l'avons vu, l'utilisation de l'optimisation dans le processus de conception aboutit à la détermination d'un ensemble de valeurs pour les paramètres de conception. Cet ensemble de valeurs sera utilisé dans la réalisation effective du circuit. Cependant, cette réalisation peut présenter la difficulté suivante que nous illustrons par un exemple :

Exemple :
Considérons une fonction objectif ϕ dépendant d'un seul paramètre R (par exemple la valeur d'une résistance).
Représentons la courbe $\phi(R)$ ci-dessous :



ϕ présente un minimum pour $R = R_o$. Supposons que la technologie ne permette de construire des résistances qu'avec une incertitude absolue ΔR. La valeur de la résistance effectivement utilisée sera dans l'intervalle $[R_o-\Delta R, R_o+\Delta R]$.

Dans l'exemple ci-dessus, on risque d'obtenir à la fabrication un circuit qui ne correspondrait plus aux spécifications.

Cet exemple montre qu'il est souvent nécessaire d'introduire à la fin du processus de conception une étude sur le comportement du circuit sous l'influence de la variation des paramètres technologiques.
Cette étude s'effectue grâce à l'analyse de tolérance.

Récemment, sont apparus des systèmes [ASTAP, etc...] permettant d'effectuer l'analyse de tolérance de circuits électroniques.

Signalons d'autre part qu'en dehors du processus de conception, l'analyse de tolérance est également utile dans la production de série de circuits. En effet, elle permet de déterminer la précision avec laquelle ils devront être réalisés pour satisfaire divers processus de fabrication possibles, de choisir le plus économique compatible avec la précision demandée.

Sans entrer dans le détail des techniques utilisées en analyse de tolérance, nous schématisons ci-dessous cette approche. Le lecteur désirant aborder plus en détail cette question, pourra se reporter à [Cal 72].

Soit C un circuit dont les variables de sorties S_i sont fonction de paramètres p_j.
Les tolérances sur S_i et p_j sont définies par :

$$p_{j_o} \leq p_j \leq p_{j_1}$$

$$S_{i_o} \leq S_i \leq S_{i_1}$$

On distingue alors deux catégories de problèmes de tolérances :

(a) - étant donné un ensemble de valeurs de tolérances (p_{j_o}, p_{j_1}), trouver l'ensemble des valeurs (S_{i_o}, S_{i_1}) correspondant à la variation la plus importante de S_i pour $p_{j_o} \leq p_j \leq p_{j_1}$.

(b) - étant donné un ensemble de tolérances admissibles (S_{i_o}, S_{i_1}), trouver les tolérances sur les paramètres permettant de les satisfaire.

Ces problèmes peuvent s'aborder en utilisant le calcul des sensibilités, c'est-à-dire celui des dérivées partielles $\frac{\partial S_i}{\partial p_j}$.

On peut écrire au premier ordre :

$$\Delta S_i = \sum_j \Delta p_j \frac{\partial S_i}{\partial p_j}\Bigg|_{p_j = p_j^\bullet}$$ où $p_j^\bullet$ est la valeur nominale de p_j.

Alors :

$$\Delta S_i^{\max} = \sum_j (p_j - p_j^\bullet)\left(\frac{\partial S_i}{\partial p_j}\right)_{p_j = p_j^\bullet}$$ où p_j est égale à p_{j_o} ou p_{j_1} de manière à rendre la somme maximum.

Toutefois, cette technique ne prend en compte que des variations au premier ordre. Aussi, dans les cas où l'on considère des variations importantes ou bien si le comportement du circuit est fortement non-linéaire au voisinage des valeurs nominales des paramètres, cette approche ne peut être utilisée.

Dans ces cas, on utilise une approche basée sur la méthode de Monte-Carlo. Elle consiste à effectuer un grand nombre de simulations en affectant aux paramètres dont on étudie l'influence, des valeurs correspondant à une répartition statistique fixée à priori. On peut ainsi calculer de façon précise la répartition statistique des variables de sorties.

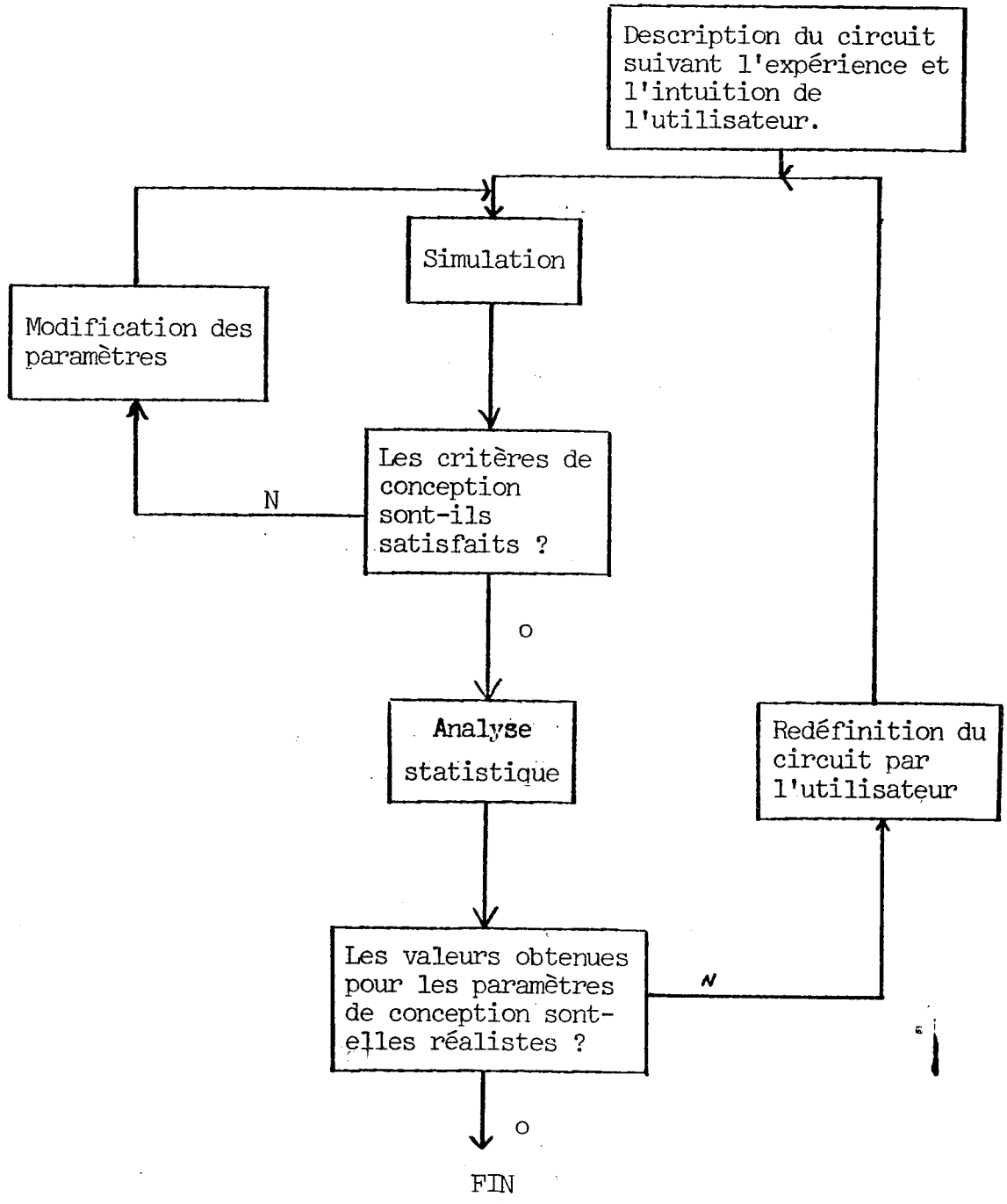
Il est clair qu'une telle méthode est coûteuse, car si l'on veut avoir une bonne représentation du phénomène, on doit effectuer un grand nombre de simulations. (Généralement, au moins une centaine). C'est pourtant la seule approche réellement utilisable dans le cas des circuits complexes actuels.

Ainsi, l'optimisation et l'analyse de tolérance apparaissent comme des techniques complémentaires dans le processus de conception.

Tandis que dans la première, on tente de déterminer les valeurs optimales d'un ensemble de paramètres du circuit de manière à satisfaire certains critères de fonctionnement, dans la seconde, on cherche à préciser l'influence sur le fonctionnement du circuit d'une répartition de la valeur des paramètres autour de leurs valeurs nominales.

L'utilisation conjointe de l'optimisation et de l'analyse de tolérance permet d'introduire plus de rigueur dans le processus de conception. En effet, on peut ainsi tester le circuit dans des conditions réelles de fonctionnement et déterminer si les optimums obtenus sont réalistes dans le contexte de la technologie de fabrication.

Le processus de conception peut alors se schématiser de la manière suivante :



Actuellement, IMAG3 ne permet pas d'effectuer automatiquement de l'analyse statistique. En effet, notre effort a porté sur la phase d'optimisation et ceci pour plusieurs raisons :

- l'analyse statistique semble encore coûteuse (environ 100 simulations pour une analyse).
- les utilisateurs du système étaient surtout des concepteurs de circuit qui sont les utilisateurs les plus concernés par l'optimisation.
- l'optimisation offre un domaine de recherche plus vaste.

Cependant, la conception modulaire d'IMAG3 permet d'envisager l'introduction de modules d'analyse statistique assez aisément.

III.2 - Langage d'entrée

En ce qui concerne l'optimisation, le langage d'entrée d'IMAG3 est identique à celui d'IMAG2. Nous n'en donnerons ici qu'un aperçu. Le lecteur intéressé pourra se reporter à la notice d'utilisation du système d'optimisation IMAG2 [CAR 73].

Comme pour le système de simulation, on peut distinguer deux parties dans ce langage :

- une première partie constituée de relations mathématiques permettant de définir la fonction objectif, les fonctions de seuil, ainsi que certaines contraintes sur les paramètres de conception et les variables du circuit.
- la seconde permet de spécifier les conditions de l'optimisation, les les algorithmes utilisés ainsi que des paramètres de contrôle pour ces algorithmes.

La première partie s'intègre à la description du circuit, tandis que la seconde est un langage de commande.

La nécessité de cette séparation est moins évidente que dans le cas de la simulation où la description correspondait à la définition d'un objet physique et la partie commande à des modes de fonctionnement.

En optimisation, la partie description correspond à la fois à la description de l'objet physique, et à la spécification de certaines propriétés qu'il devrait posséder.

Le langage de commande précise les propriétés que l'on désire vérifier, les moyens permettant d'atteindre à ces propriétés : (quels paramètres modifier ? etc...), ainsi que des spécifications techniques pour les algorithmes.

III.2.1 - Fonctions objectifs, seuils et contraintes

La fonction objectif apparaît dans la description du circuit comme une expression portant sur des variables ou des paramètres. Cependant, elle ne sera spécifiée comme fonction objectif que par une instruction du langage de commande.

Elle peut faire intervenir des valeurs d'une même variable relative à différents états du circuit. Il est alors nécessaire de mémoriser, au cours du calcul de chacun des états, la valeur correspondante de la variable dans une variable intermédiaire.

Exemple :

Etant donné un circuit alimenté par une source E_1, on désire que la tension de sortie V_1 reste égale à 12V quand E_1 prend les valeurs 2V, 3V et 4V.

On peut calculer pour chacun des trois états la valeur (V1 - 12) et minimiser la somme quadratique de ces trois quantités.

On définit trois variables A_1, A_2, A_3 et la fonction objectif prend la forme :

$$F = \{A_1 - 12\}^2 + \{A_2 - 12\}^2 + \{A_3 - 12\}^2$$

Une instruction du langage de commande permet d'affecter à A_1, A_2 et A_3 la valeur de V1 correspondante.

Des fonctions standards (sommation, etc...) permettent de simplifier l'écriture des fonctions objectives.

Remarque :

Pour avoir un maximum d'efficacité des algorithmes d'optimisation, on a intérêt à définir la fonction objectif de manière à ce que :

- elle soit toujours positive ou nulle
- son minimum soit zéro.

Une fonction de seuil est définie dans la description par une expression qui s'annule pour la valeur désirée et qui initialement est négative.

Exemple :

Reprenons le deuxième exemple du paragraphe (III.1.1). Initialement, V1 = o. La fonction de seuil s'écrit :

S = (V1 - 9)

Une expression est spécifiée comme fonction de seuil par une instruction du langage de commande.

Les contraintes sont introduites dans la description comme des expressions attachées à une variable. Elles sont reconnues comme contraintes par une instruction du langage de commande.

III.2.2 - Langage de commande

La commande OPTI permet d'entrer dans l'environnement optimisation du système.
Le langage de commande permet de préciser :

- la liste des paramètres de conception par rapport auxquels va s'effectuer l'optimisation :

  SENS : < liste de paramètres > $

- la fonction objectif :

  MIN : < nom de la fonction objectif >

- les contraintes :

  CONSTR : < liste de contraintes >

- le régime d'analyse choisi : continu, transitoire, alternatif.

- l'algorithme d'optimisation (algorithmes de Fletcher et Powell ou Fletcher et Reeves).

- le nombre maximum d'itérations et la précision.

- les valeurs que l'on attend en sortie.

D'autre part, une simulation préalable dans le régime choisi sert d'initialisation. Au cours de cette initialisation, sont précisées les fonctions de seuil, la signification de toutes les variables intermédiaires.

III.3 - Aspect interne

L'aspect interne du système d'optimisation, comme le système de simulation, se différencie en deux parties :

- la compilation du langage d'entrée (traduction)
- les algorithmes de résolution qui utilisent les résultats (invariants) générés par la première phase.

La compilation du langage d'entrée s'effectue par des techniques similaire à celles qui sont utilisées dans le système de simulation.

Aussi, nous n'y insisterons pas. Nous signalerons seulement au cours de l'exposé des algorithmes les invariants intéressants qui ont été générés.

Le problème de la "conception automatique" de circuits électroniques ayant été formulé sous la forme de minimisation d'une fonction objectif, les algorithmes numériques seront donc des algorithmes permettant de rechercher le minimum d'une fonction de plusieurs variables.

Après avoir rappelé quelques propriétés du minimum d'une fonction, nous exposerons les algorithmes effectivement utilisés dans la recherche de ce minimum.

Comme nous le verrons, les algorithmes réellement efficaces dans le domaine qui nous concerne, utilisent le gradient $\{\frac{\partial \phi}{\partial p}\}$ de la fonction objectif.

Nous avons développé et mis en oeuvre une méthode de calcul du gradient de la fonction objectif que nous examinerons dans la troisième partie.

L'aspect utilisation du système est bien entendu lié à la sémantique du langage d'entrée. Cependant, étant donnée la nature particulière du problème et des limitations dues essentiellement aux algorithmes numériques utilisés, l'aspect utilisation est fortement lié à l'aspect interne sans que l'on puisse toujours établir ces liens au seul examen du langage d'entrée.

Aussi, nous terminerons en indiquant à l'utilisateur les conséquences de la structure interne du système sur la formulation de son problème.

III.3.1 - Minimum - Domaine convexe

Si ϕ est une fonction d'une seule variable p_i, définie sur $[a, b] \subset R$, une condition nécessaire et suffisante pour que ϕ soit minimum pour $p_i = \tilde{p}_i$

$$\left.\frac{\partial \phi}{\partial p_i}\right)_{p_i = \tilde{p}_i} = o \qquad \text{et} \qquad \left.\frac{\partial^2 \phi}{\partial p_i^2}\right)_{p_i \in [a,b]} > o \qquad \text{(II.30)}$$

Si p est un vecteur à n composantes, notons $\tilde{p}$ la valeur de p pour laquelle $\phi(p)$ est minimum. Développons ϕ en série de Taylor au voisinage de $\tilde{p}$ au second ordre :

$$\phi(p) = \phi(\tilde{p}) + \{grad(p)\}^T . \Delta p + \frac{1}{2} \Delta p^T \left[\frac{\partial^2 \phi}{\partial p_i \partial p_j}\right] \Delta p$$

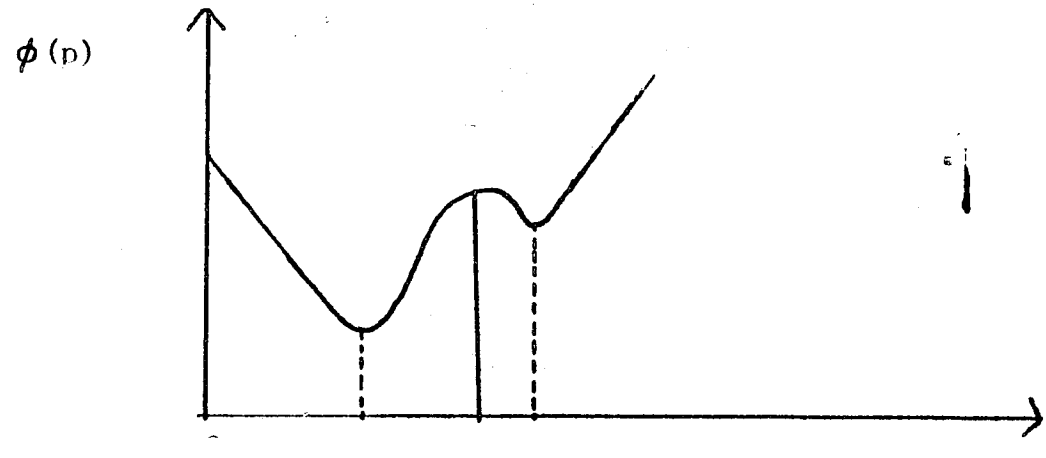
ou : grad(p) est un vecteur colonne et Δp le vecteur de composantes $\{p_i - \tilde{p}_i\}$.

$\tilde{p}$ réalisant un minimum de $\phi(p)$, un changement sur p doit accroître $\phi(p)$. Ceci est équivalent à exiger que les termes dépendants linéairement de Δp, soient nuls et que la forme quadratique constituée par les termes du second degré dans le développement de Taylor, soit définie positive (c'est-à-dire puisse s'écrire comme une somme de carrés).
La matrice $\left\{\frac{\partial^2 \phi}{\partial p_i \partial p_j}\right\}$ est la matrice Hessienne H de ϕ.

Les notions de minimum local et de domaine convexe jouent un rôle important en optimisation et marquent les limites de ce que l'on peut en attendre à l'heure actuelle.

Nous nous contenterons de donner une idée intuitive de ces notions.

Considérons la figure ci-dessous qui représente une fonction $\phi(p)$ d'une seule variable :



On constate sur cette figure que deux minimas existent :

- un minimum dit <u>global</u> pour $p = p_1$
- un minimum dit <u>local</u> pour $p = p_2$

Seul le minimum global correspond à la solution du problème. L'existence de minimas locaux introduit une incertitude sur la solution du problème car il n'existe aucun test local pour déterminer la nature d'un minimum.

Une façon d'aborder ce problème est de choisir la valeur initiale des paramètres de conception de manière à se situer dans un voisinage du minimum global (localisé par des considérations sur la nature du phénomène étudié).

Si cette localisation ne peut se faire, il faut explorer $\phi(p)$ sur tout le domaine présentant un intérêt pour le problème.

Ceci devient rapidement prohibitif quand la dimension du vecteur p croît.

<u>Définition</u> : un domaine $D \subset E_n$ est convexe si :

$\lambda x_1 + (1-\lambda)\, x_2 \in D$ quel que soit $x_1 \in D$ et $x_2 \in D$, et quel que soit $\lambda \in [o, 1]$.

La convexité d'une fonction ϕ définie dans D, se définit alors de la manière suivante :

<u>Définition</u> : soit D un domaine convexe et ϕ une fonction définie dans D. ϕ est convexe dans D si :

$$(a) \quad -\phi\,[\lambda x_1 + (1-\lambda)\, x_2] \leq \lambda\phi(x_1) + (1-\lambda)\,\phi(x_2)$$

$x_1, x_2 \in D$ et $\lambda \in [o, 1]$.

ϕ est strictement convexe dans D si l'on a l'inégalité stricte dans (a) quand $o < \lambda < 1$ et $x_1 \neq x_2$.

On peut énoncer le théorème suivant :

<u>Théorème</u> : Si ϕ est strictement convexe dans D, alors un minimum de ϕ dans D est unique.

Dans le cas où ϕ est seulement convexe, alors tout minimum local est également un minimum global.

Les algorithmes actuellement connus de recherche de minimums, donnent un minimum local sans que l'on puisse préciser s'il est global. Ainsi, seule la convexité de $\phi(p)$ permet d'assurer que le minimum trouvé est global. Malheureusement, la plupart des problèmes pratiques rencontrés ne sont pas convexes et c'est ce qui explique les solutions approximatives utilisées pour lever l'incertitude sur la nature du minimum calculé.

Nous envisagerons dans le paragraphe III.3.2.2. le problème du "minimum aux bornes", que l'on rencontre dans la recherche d'un minimum en présence de contraintes sur les paramètres.

III.3.2 - Recherche effective du minimum

La généralité du système IMAG3 entraîne une très grande diversité dans les formes de la fonction objective et élimine de ce fait l'utilisation d'algorithmes spécialisés particuliers.

L'utilisation d'un algorithme pour un problème tel que l'optimisation doit être jugé suivant deux critères principaux :

- son efficacité, c'est-à-dire sa faculté d'obtenir le minimum de la fonction, dans le meilleur temps possible.

Dans ce cas, le coût est lié au nombre de fois où l'on évalue la fonction objectif, le déroulement de l'algorithme lui-même étant de durée négligeable.

- son encombrement. La taille d'un programme conditionne souvent son utilisation par une personne intéressée, car il faut que le programme soit adaptable au matériel dont elle dispose.

Depuis quelques années, la méthodologie de recherche d'un optimum (maximum ou minimum), a connu un grand essor.
Citons dans l'ordre chronologique :

- les méthodes à métrique variable (Davidson - Fletcher - Powell : 1963),

- gradient conjugué (Fletcher - Reeves : 1964),

- directions conjuguées (Powell : 1964, Zangwill : 1967, Broyden - Fletcher Goldfarb - Shanno : 1968),

- les méthodes du 2° ordre (utilisation des dérivées secondes), en particulier, celle de Fiacco - Mc Cormick : 1968.

Les algorithmes ci-dessus ne tiennent pas compte de contraintes éventuelles dans le problème à traiter.

Aussi, on transforme généralement le problème avec contraintes en un problème sans contrainte, de manière à appliquer un de ces algorithmes.

Cette transformation s'effectue en introduisant dans la fonction objectif originale des fonctions auxiliaires (barrières, pénalités), permettant de tenir compte des contraintes.
Nous examinons successivement ces deux aspects :

- algorithmes pour un problème sans contrainte,
- transformation d'un problème avec contraintes.

III.3.2.1 - Algorithmes pour un problème sans contrainte

Les algorithmes procèdent tous à peu près de la même manière pour trouver la solution.
On définit un schéma de récurrence qui doit conduire à celle-ci :

$$p_{q+1} = p_q + L_q S_q$$

où :

- p_q et p_{q+1} sont les valeurs du vecteur des paramètres de conception aux itérations q et q + 1.
- S_q est un vecteur
- L_q est un scalaire

Les algorithmes, tous itératifs, se distinguent par la façon de choisir L_q et S_q.

S_q indique la direction dans laquelle va s'effectuer la recherche du point p_{q+1} à partir de p_q.

On distingue trois modes de calcul pour S_q :

- ceux où S_q se calcule directement à partir des valeurs de la fonction aux points précédents,
- ceux qui font intervenir les dérivées partielles de la fonction objectif ,
- ceux qui font intervenir les dérivées partielles secondes.

L'expérience a montré que les algorithmes des deuxième et troisième catégories étaient plus efficaces (surtout dans le cas où le calcul de ϕ est coûteux).

Cependant, le calcul de la matrice Hessienne est coûteux et souvent peu précis (obtenue à partir des dérivées premières par dérivation numérique).

Aussi, on a intérêt à choisir des algorithmes de la 2° catégorie ou de la troisième, dans le cas où ils peuvent se ramener à ceux de la seconde.

Plusieurs algorithmes ont été mis en oeuvre par G. CARRY [CAR 73] au fur-et-à-mesure de l'élaboration du système d'optimisation construit à partir d'IMAG2 de manière à en juger les qualités respectives.

Nous n'exposerons pas ici le détail de tous les algorithmes testés. Le lecteur intéressé pourra se reporter à [CAR 73].

Parmi les divers algorithmes envisagés, les plus intéressants ont semblé être :

- l'algorithme de Davidon - Fletcher - Powell.

C'est un algorithme de la troisième catégorie, ramené à un algorithme de la deuxième catégorie.
La récurrence s'écrit :

$$p_{q+1} = p_q - L_q \, H_q \, \text{grad}_p \, (\phi). \qquad \text{(II.32)}$$

La direction de recherche est donc :

$$S_q = - H_q \, \text{grad}_p \, (\phi).$$

où H_q est l'inverse de la matrice Hessienne.

Cependant, le calcul de H ne s'effectue pas par l'intermédiaire de la matrice Hessienne (trop long), mais par une formule d'approximation :

$$H_{q+1} = H_q + M_q + N_q$$

ou :

$$M_q = L_q \, \frac{S_q \, S_q^T}{S_q^T \, Y_q}$$

$$N_q = \frac{(H_q \, Y_q)(H_q \, Y_q)^T}{Y_q^T \, M_q \, Y_q}$$

$$Y_{q+1} = \text{grad}(\phi)_{q+1} - \text{grad}(\phi)_q$$

H_o étant égal à la matrice unitaire.

On démontre que H_{q+1} tend vers l'inverse de la matrice Hessienne lorsque p tend vers la solution.

Cet algorithme donne dans la plupart des cas des résultats satisfaisants et semble le plus efficace parmi ceux qui ont été testés. Cependant, la place mémoire requise est importante puisqu'il faut mémoriser la matrice H_q, ce qui demande beaucoup de place lorsque le nombre de paramètres de conception devient important.

Fletcher et Reeves ont proposé un algorithme un peu moins efficace que le précédent, mais moins encombrant. Pour cette raison, il est intéressant, dans les cas limites d'utilisation du système.

Il part du principe que le gradient de la fonction en un point, doit donner la direction dans laquelle elle croît le plus rapidement. Cependant, l'utilisation du gradient conduit souvent à un phénomène de "zig-zag", c'est-à-dire que la courbe joignant les points successifs est une ligne brisée dont les angles sont proches de l'angle droit. Aussi, pour éviter cet inconvénient, Fletcher et Reeves construisent la direction de recherche non seulement en fonction du gradient au point atteint, mais aussi du gradient au point précédent.
La récurrence s'écrit :

$$p_{q+1} = p_q + L_q S_q$$

$$S_q = - \text{grad } \phi_q + \beta_{q-1} S_{q-1} \qquad \text{II.33}$$

$$\beta_{q-1} = \{(\text{grad}(\phi)_q / \text{grad}(\phi)_{q-1})\}^2$$

Cette solution restant empirique, on limite périodiquement les risques d'erreurs en prenant, comme pour l'initialisation, la direction S_q dans la direction grad ϕ_q.

Fletcher et Reeves ont constaté que, si le nombre de paramètres était n, une réinitialisation tous les n+1 pas, donnait de bons résultats.

La direction de recherche S_q étant choisie par un des algorithmes précédents, il s'agit de choisir L_q tel que :

$\phi(p_q + L_q S_q)$ soit minimum dans la direction S_q.

Notons que ϕ est alors une fonction de la seule variable L_q. Le problème initial est ainsi ramené à la minimisation d'une fonction d'une seule variable.

Ce problème, souvent appelé recherche linéaire, semble extrêmement simple. En réalité, c'est une des difficultés majeures dans la mise en oeuvre de ce type de méthodes. En effet, c'est de sa solution correcte que dépendra en dernier ressort l'efficacité des algorithmes d'optimisation. D'autre part, sa résolution ne doit pas être trop coûteuse car elle peut intervenir un grand nombre de fois. En particulier, elle ne devra demander qu'un nombre minimum d'évaluations de ϕ (il ne faut pas oublier qu'une évaluation de ϕ demande une simulation du circuit).

Diverses méthodes peuvent être utilisées. Dans IMAG2 et IMAG3, $\phi(L_q)$ est approchée par une cubique :

$$\phi(L_q) = aL_q^3 + bL_q^2 + cL_q + d$$

dont on peut calculer le minimum de manière analytique [CAR 73].

Ces algorithmes de recherche du minimum, développés autour d'IMAG2, ont été repris intégralement dans IMAG3, car :

- ils ont été jugés satisfaisants
- ils ne préjugent pas de la forme des équations de fonctionnement du circuit, ni même de celle de la fonction objectif.

III.3.2.2 - Problèmes avec contraintes

La méthodologie (adoptée ici) de résolution d'un problème avec contraintes, est de le ramener à un problème sans contrainte par transformation de la fonction objectif.

Les travaux les plus marquants dans ce domaine sont ceux de Fiacco - Mc Cormick [FIA 65, FIA 73], et ceux de Lootsma [LOOT 70]. Ces travaux ont été largement utilisés dans l'élaboration du système d'optimisation.
Soit donc à minimiser $\phi(p)$ sous les contraintes :

$$g_i(p) \geq o \quad \text{pour } i = 1, \ldots, m.$$

Nous supposerons que les contraintes sont toujours données sous cette forme, de manière à pouvoir appliquer les algorithmes, mais cette restriction est faible.
Considérons la fonction :

$$G(p) = \phi(p) - r \sum_{i \in I_1} \mathrm{Log}\{g_i(p)\} + \frac{1}{r} \sum_{i \in I_2} \{\min[o, g_i(p)]\}^2$$

où ϕ est la fonction objectif et r un paramètre auxiliaire.

- $I_1 = \{i : g_i(p) > o\}$

- $I_2 = \{i : g_i(p) \leq o\}$

Le terme $r \sum_{i \in I_1} \mathrm{Log}\{g_i(p)\}$ est appelé <u>fonction barrière</u> parce que lorsque $g_i(p) \to o$, il tend vers l'infini constituant ainsi une barrière infranchissable.

Le terme $\frac{1}{r} \sum_{i \in I_1} \{\min(o, g_i(p))\}^2$ est appelé <u>fonction pénalité</u> car il "pénalise" G(p) lorsque $g_i(p) < o$.

Généralement, on utilise une seule des deux fonctions auxiliaires. Dans IMAG3, ainsi que le préconise LOOTSMA, on utilise une combinaison des deux [LOOT 70] :

$$G(p) = \phi(p) - r \sum_{i \in I_1} \mathrm{Log}\, g_i(p) + \frac{1}{r} \{\sum_j h_j(p) + \sum_{I_2} \{\min[o, g_i(p)]\}^2\}$$

où les contraintes sont de la forme :

$$\begin{cases} g_i(p) \quad o \ ; \ i=1 \ldots m \\ h_j(p) = o \ ; \ j=1, \ldots p \end{cases}$$

et où I_1 et I_2 sont définis par :

$I_1 = \{i / g_i(p) > o \ ; \ 1 \leq i \leq m\}$

$I_2 = \{i / g_i(p_o) \leq o \ ; \ 1 \leq i \leq m\}$

Où p_o est la valeur initiale donnée aux paramètres.

LOOTSMA a pu montrer que :

si pour chaque valeur r_k de r, on peut trouver une valeur $p(r_k)$ rendant G minimum, la suite $p(r_k)$ converge vers la solution du problème initial lorsque $k \to \infty$, si la suite r_k est monotone décroissante vers o.

Ainsi, on est ramené à une suite de problèmes sans contrainte (minimisation de G_k).

Dans IMAG2 et IMAG3, le traitement des contraintes est construit à partir de ces résultats.
Nous n'entrerons pas dans le détail des difficultés pratiques soulevées par cette méthode. Le lecteur pourra se reporter à la notice de programmation du système d'optimisation IMAG2 [CAR 73 a], cette mise en oeuvre ayant été reprise intégralement dans IMAG3. Signalons qu'un des problèmes majeurs est le choix de la valeur initiale de r : r_o, et la manière dont celui-ci va être modifié.

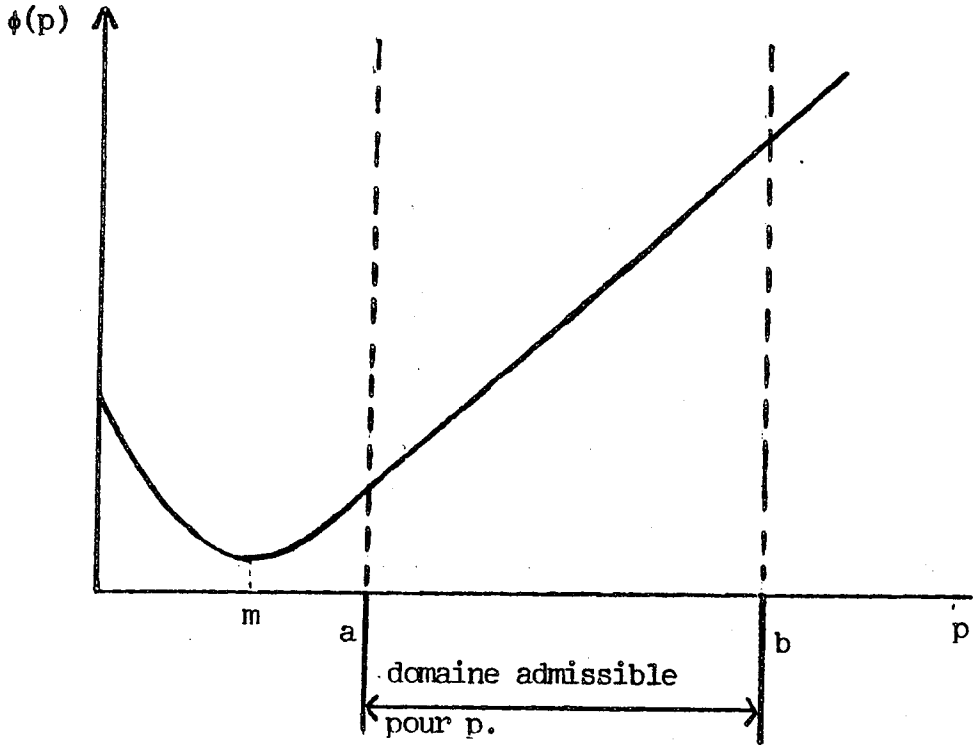
Lootsma propose de choisir :

$$\begin{cases} r_o = \max\{10^{-2}\ ,\ V^*/100\} \\ r^k = r^{k-1}/\sqrt[3]{10} \end{cases}$$

où V^* est une estimation du minimum.

Cette solution, adoptée dans IMAG2, est arbitraire et une fois encore, l'expérience du problème traité joue un rôle essentiel.
Les résultats obtenus sur des exemples réels sont satisfaisants.

Dans le cas d'une recherche de minimum en présence de contraintes, se pose le problème du minimum se trouvant à l'extérieur du domaine où sont vérifiées les contraintes.
Cette situation peut se schématiser par la figure suivante :

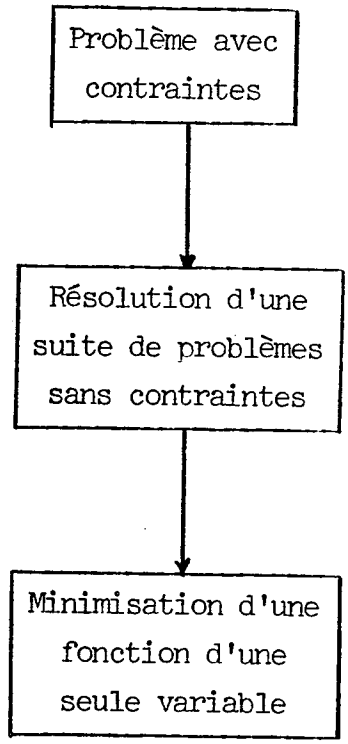


Les contraintes sur p sont de la forme : $a \leq p \leq b$. Si l'algorithme de minimisation converge, il donnera a comme valeur de p, rendant $\phi(p)$ minimum. Cependant, le minimum de $\phi(p)$ en l'absence de contraintes est obtenu pour $p = m$.

On parlera, dans ce cas, de <u>minimum aux bornes</u>.

On ne peut donner une règle générale pour interpréter une telle situation. Cela dépendra de la forme de $\phi(p)$ et de la manière dont elle traduit le fonctionnement du circuit.
Ce sera donc à l'utilisateur de refuser ou d'accepter la valeur de p obtenue.

Finalement, la résolution d'un problème avec contraintes se décompose en trois phases :



Dans IMAG2 et IMAG3, chaque phase est réalisée par un module (ou un ensemble de modules) particulier. Bien entendu, certains tests utilisés dans une phase ne sont pas indépendants des algorithmes mis en oeuvre dans les autres phases. Cependant, cette organisation fournit un canevas dans lequel il est relativement aisé de remplacer un algorithme par un autre équivalent, et que l'on pense plus approprié. Ceci a pu se vérifier lorsque A. BENSASSON [BEN 74] a voulu tester les algorithmes qu'il a développés sur des exemples réels. D'autre part, cela permet une adaptation aisée à l'évolution des algorithmes disponibles.

III.3.3 - Calcul du gradient de la fonction objectif

Les algorithmes les plus efficaces de recherche d'un minimum dans le cas d'une fonction non-linéaire - comme ceux indiqués dans le paragraphe précédent - utilisent son gradient. Il convient donc de rechercher des algorithmes performants pour ce calcul.

En fait, les systèmes d'optimisation de circuits électroniques se distinguent essentiellement (d'un point de vue interne), par la méthode utilisée pour évaluer ce gradient.

Nous nous intéressons ici au cas général du calcul en régime transitoire, le calcul en régime continu n'en étant qu'un cas particulier.
On cherche à minimiser la fonctionnelle :

$$\phi(p) = \int_0^{T(p)} I\,(y, y', u, p, t)\,dt \qquad (II.34)$$

où $u(p,t)$ et $y(p,t)$ sont les variables algébriques et différentielles associées au circuit. $T(p)$ est donné par une fonction de seuil :

$$S(T(p), y(p,T), u(p,T)) = o \qquad (II.35)$$

Posons dans la suite $Y = \{u, y\}$.

La difficulté pour évaluer le gradient $\{\frac{\partial \phi}{\partial p}\}$ est que la valeur du vecteur Y au temps t dépend de p.
Cependant, $\frac{\partial Y}{\partial p}$ ne peut être évalué analytiquement car la dépendance de Y par rapport à p et t n'est connue qu'au travers de la solution des équations de fonctionnement :

$$F(y, y', u, t) = o \qquad (II.2)$$

Une approche que l'on peut envisager facilement est la suivante :

- soit m le nombre de paramètres de conception.

On effectue d'abord une simulation permettant d'évaluer $\phi(p)$, puis m simulations successives. Chacune d'entre elles étant réalisée avec une perturbation Δp_i de chacun des m paramètres. On obtient ainsi $\phi(p + \Delta p)$. On évalue ensuite $\{\frac{\partial \phi}{\partial p}\}$ par une dérivation numérique.

Cette méthode - relativement coûteuse - demande donc m + 1 simulations du circuit pour chaque calcul du gradient. Il est bien évident que ce coût est prohibitif pour des circuits volumineux. Elle a été utilisée récemment dans le programme OPTISEM [ESSL 74].

Deux autres approches, réduisant le nombre de simulations, mais beaucoup plus complexes à mettre en oeuvre, ont été développées :

- la première due à Hachtel et Rohrer [Hacht 67] , utilise des techniques tirées du calcul des variations.

- la seconde, due à Director et Rohrer [Dir 69], utilise le théorème de Tellegen [Tel 52].

Nous les résumons ci-dessous.

L'introduction de ces méthodes dans un système tel qu'IMAG3 était complexe. Aussi, il nous est apparu souhaitable de développer une méthode plus facile à mettre en oeuvre et permettant d'utiliser judicieusement l'ensemble des invariants générés lors de la phase de traduction. Elle s'inspire d'une méthode utilisée dans IMAG2 pour le calcul des sensibilités [JAC 70].

III.3.3.1 - Méthode de Hachtel et Rohrer

Supposons pour simplifier les équations de fonctionnement sous la forme : $\frac{dY}{dt} = F(Y, t)$.

Hachtel et Rohrer introduisent la fonction de Lagrange :

$$L(Y, p, \lambda, \mu, t) = I(Y, p, t) + \lambda^T(Y' - F(Y, p, t)) + \mu^T p'$$

où :

$\lambda^T = (\lambda_1, \lambda_2, \ldots \lambda_N)$ est un vecteur ligne de même dimension que Y et

$\mu^T = (\mu_1, \mu_2, \ldots, \mu_M)$ est un vecteur ligne de même dimension que p.

λ et μ sont les vecteurs adjoints (respectivement des variables et des paramètres).

Si les équations de fonctionnement sont satisfaites et si $\frac{dp}{dt} = o$, on peut écrire :

$$\phi(p) = \int_o^T L(y, y', p, p', t, \lambda, \mu)\, dt$$

Il s'agit de rechercher la variation par rapport à p de la fonction ϕ

Soit $z(t) = \text{col}(y^T, p^T, \lambda^T, \mu^T)$, alors :

$$\delta\phi = \int_o^T \{\frac{\partial L}{\partial z}\ \delta z + \frac{\partial L}{\partial z'}\ \delta z'\}\ dt + \int_T^{T+\delta T} L.dt$$

Appliquant le théorème de la moyenne au second membre et intégrant par partie $\frac{\partial L}{\partial z'}\ \delta z'$, il vient :

$$\delta\phi = \int_o^T (\frac{\partial L}{\partial z} - \frac{d}{dt}\ \frac{\partial L}{\partial z'})\ \delta z\ dt + (T)\ \delta T + (\frac{\partial L}{\partial z'}\ \delta z')_{t=T} - (\frac{\partial L}{\partial z'}\ \delta y)_{t=o}$$

$\phi(p)$ est fonction de p seul, donc $\delta\phi$ dépend seulement de p et δp, donc les seules trajectoires intéressantes sont celles pour lesquelles l'intégrale dans la relation ci-dessus est nulle.

On démontre que ceci entraîne :

$$\frac{\partial L}{\partial z} - \frac{d}{dt}\ \frac{\partial L}{\partial z'} = o$$

Ces équations sont connues sous le nom d'équations d'Euler.
A l'aide de ces équations et de l'expression donnant $\delta\phi$, on trouve finalement :

(II.36) $$\frac{\partial \phi}{\partial p} = -\lambda^T(o) + \frac{\partial Yo(p,o)}{\partial p} + \int_o^T \{\frac{\partial I(y,p)}{\partial p} - \lambda^T(t)\ \frac{\partial F(y,p)}{\partial p}\}\ dt$$

L'application de cette méthode peut se schématiser de la manière suivante :

(1) Intégrer les équations de fonctionnement sur $[o, T]$, T étant déterminé par la fonction de seuil.

(2) Calculer $\lambda(T) = \{I_T \frac{\partial S}{\partial Y_T}\} / \{\frac{\partial S}{\partial Y}\ \frac{\partial F}{\partial Y} + \frac{\partial S}{\partial t}\}$

(3) Déterminer $\lambda(t)$ sur $[T, o]$ en intégrant $\lambda' = -\lambda \frac{\partial F}{\partial y} + \frac{\partial I}{\partial y}$

sur cet intervalle (équations d'Euler).

(4) Evaluer $\frac{\partial \phi}{\partial p}$ par (II.36).

Cette méthode demande l'intégration de deux systèmes différentiels, l'un dans le sens direct, l'autre dans le sens inverse. Les pas d'intégration n'étant pas forcément les mêmes, on est conduit à des interpolations qui peuvent être coûteuses.

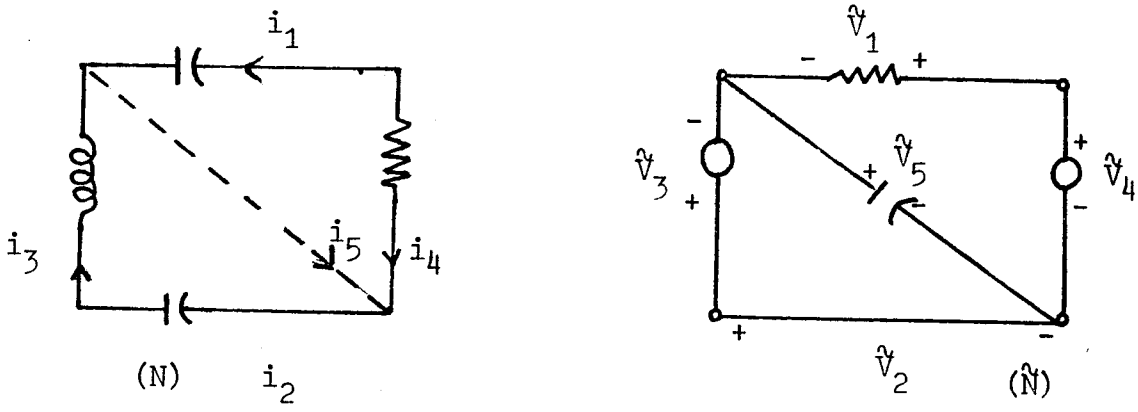
A notre connaissance, elle n'a jamais été utilisée dans un système opérationnel, mais elle a le mérite de montrer que dans ces problèmes, le calcul du gradient de la fonction objectif peut s'effectuer avec deux intégrations seulement, quelque soit le nombre de paramètres.

III.3.3.2 - Méthode du réseau adjoint (Director et Rohrer).

Cette méthode peut être considérée comme une traduction "physique" de la précédente, plus compréhensible et plus facile à mettre en oeuvre. Elle repose sur un théorème dû à Tellegen et dont la liaison avec le problème posé n'est pas évidente.

Soit N et $\hat{N}$ deux circuits ayant la même topologie, mais pas nécessairement les mêmes composants sur les branches correspondantes.

Exemple



$\hat{N}$ sera dit circuit adjoint de N.

Soit $v_B(t)$ et $i_B(t)$ les courants et tensions de branche dans N et $\hat{v}_B(t)$ et $\hat{i}_B(t)$ les courants et tensions dans $\hat{N}$.
Le théorème de Tellegen permet d'écrire :

$$\sum_B v_B(t).\hat{i}_B(t) = o \quad \text{et} \quad \sum_B i_B(t)\,\hat{v}_N(t) = o.$$

(la sommation s'effectue sur les branches de N et $\hat{N}$).

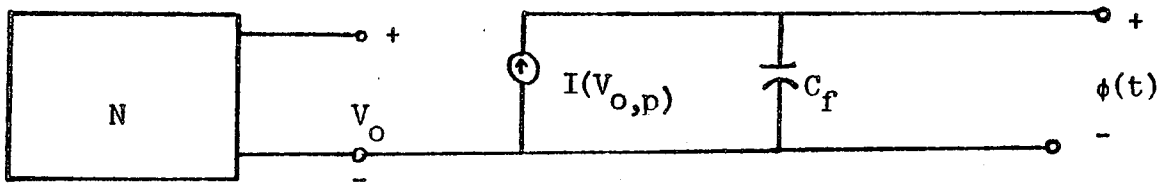
Considérons maintenant une perturbation des valeurs des éléments de N provoquant une variation des tensions et des courants de branches, soit $\Delta v_B(t)$ et $\Delta i_B(t)$.

On montre facilement l'importante relation :

$$\sum_B \{\Delta v_B(t).\tilde{i}_B(t) - \Delta i_B(t).\tilde{v}_B(t)\} = o. \qquad \text{(II.37)}$$

La clé pour calculer $\frac{d\phi}{dp}$ est d'inclure $d\phi$ dans la sommation (II.37).

Supposons que le circuit que l'on désire étudier est le circuit N précédent. Ajoutons à N un circuit dit "objector" comme indiqué sur la figure suivante :



I étant tel que $\phi(p) = \int_o^T I\,dt$, la tension aux bornes de la capacité C_f est précisément égale à la valeur de la fonction objectif pour $t = T$ si $C_f(o) = o$.

La méthode peut alors se schématiser ainsi :

- construire le circuit adjoint du circuit étendu.

- déterminer la somme (II.37).

Puisque ϕ est représentée comme une tension de sortie, un terme de la forme $d\phi - \tilde{i}_\phi$ apparaîtra dans (II.37) d'où l'on pourra déduire le gradient de ϕ.

Remarques :

(1) - Quand T est variable, la somme (II.37) ne reflétant que les changements dans des paramètres du circuit, la variation de T ne peut être représentée. On peut tout de même résoudre le problème en ayant recours à un artifice :

- la variation de ϕ s'exprime par :

$$d\phi = \frac{\partial \phi}{\partial T} dT + d\phi_{tcl} \quad \text{où} \quad d\phi_{tcl} \text{ représente la variation à T cons}$$

On est conduit à introduire dans la somme (II.37) l'expression :

$$d\phi - \frac{\partial \phi}{\partial T} dT = d\phi - I_T \, dT \quad \text{pour prendre en compte dT.}$$

(2) - Cette méthode entraîne des manipulations symboliques relativement complexe (écriture des équations du circuit adjoint) qu'il était difficile de mettre en oeuvre dans un système comme IMAG3. D'autre part, ses manipulation augmentent sensiblement le coût de la phase de traduction.

III.3.3.3 - Méthode proposée.

La méthode ci-dessous a le mérite de se mettre en oeuvre aisément dans IMAG3 car les quelques manipulations symboliques supplémentaires requises dans la phase de traduction sont du même ordre que celles qui sont utilisées dans le calcul formel des termes du jacobien tel qu'il est exposé au chapitre I.

Cette approche est une extension de la méthode utilisée dans IMAG2 pour le calcul des sensibilités - c'est-à-dire le calcul de la dérivée partielle d'une variable du circuit par rapport à un paramètre - au cas d'une fonction objectif donnée sous la forme (II.34) et tenant compte de la contrainte à la limite (II.35).

$$\phi(p) = \int_o^{T(p)} I(Y, y', p, t) \, dt \qquad (II.34)$$

$$S\{T(p), y(p, T), p\} = o \qquad (II.35)$$

On pose : $Y = \{y, u\}$.

Dans (II.34), ϕ est fonction de p seul. En effet, Y et y' sont liées à p par l'intermédiaire des équations de fonctionnement du circuit : $F(Y, y', p, t) = o$.

D'autre part, si l'on est à un état stable du circuit pour $t_o = o$, $Y(t_o)$ dépend de p seul. On en déduit que T, lié à p et Y par (II.35), dépend également de p seul.

Dans ces conditions, on peut appliquer à ϕ la formule de dérivation d'une intégrale à borne variable (1).

Il vient :

$$\frac{\partial \phi}{\partial p} = \int_o^{T(p)} \left\{ \frac{\partial I}{\partial p} + \frac{\partial I}{\partial Y} \frac{\partial Y}{\partial p} + \frac{\partial I}{\partial y'} \frac{\partial y'}{\partial p} \right\} dt + \frac{\partial T}{\partial p} I(Y_T, y'_T, p, T) \qquad (II.38)$$

Les équations de fonctionnement sont vérifiées quel que soit p, $dF_{/p} = o$, ce qui s'exprime par :

$$\frac{\partial F}{\partial Y} \frac{\partial Y}{\partial p} + \frac{\partial F}{\partial y'} \frac{\partial}{\partial p} \left(\frac{dy}{dt}\right) + \frac{\partial F}{\partial p} = o \qquad (II.39)$$

T(p) est déterminé par la fonction de seuil qui va nous permettre d'évaluer $\frac{\partial T}{\partial p}$.

$S\{T(p), y(p, T), p\} = o$ quel que soit p donc $dS_{/p} = o$ et

$$\frac{\partial S}{\partial p} + \frac{\partial S}{\partial Y} \frac{\partial y}{\partial p} + \frac{\partial S}{\partial Y} \frac{\partial y}{\partial T} \frac{\partial T}{\partial p} + \frac{\partial S}{\partial T} \frac{\partial T}{\partial p} = o.$$

$$\frac{\partial y}{\partial T} = \left\{\frac{dy}{dt}\right\}_{t=T} \quad \text{par les équations de fonctionnement.}$$

Donc :

$$\frac{\partial T}{\partial p} = - \frac{\left\{\frac{\partial S}{\partial p} + \frac{\partial S}{\partial y} \cdot \frac{\partial y}{\partial p}\right\}}{\left\{\frac{\partial S}{\partial y} \frac{\partial y}{\partial T} + \frac{\partial S}{\partial T}\right\}} \qquad (II.40)$$

Pour évaluer $\frac{\partial \phi}{\partial p}$, il nous reste à évaluer $\frac{\partial Y}{\partial p}$ qui intervient dans (II.38) et (II.40) (les autres quantités peuvent s'évaluer directement par un calcul formel analogue au calcul des termes de la matrice de Jacobi).

---

(1) En supposant satisfaites les conditions de dérivabilité sur I.

Revenons à (II.39). Si l'on remarque que

$\frac{\partial Y'}{\partial p} = \frac{\partial}{\partial p} \left(\frac{dy}{dt}\right) = \frac{d}{dt} \left(\frac{\partial Y}{\partial p}\right)$, et si k est la dimension du vecteur p, (II.39)

représente k systèmes algébro-différentiels en $\frac{\partial Y}{\partial p}$.

Les équations de fonctionnement sont intégrées par une formule à pas liés d'ordre q du type :

$$y'_n = - \frac{Lo}{h\beta_o} y_n + \sum_{i=1}^{q} a_i y_{n-i}$$

$$\frac{\partial yn}{\partial p} = \frac{\partial}{\partial p} \left(- \frac{Lo}{h\beta_o} y_n + \sum_{i=1}^{q} a_i y_{n-i}\right)$$

Faisons l'hypothèse suivante :
le pas h d'intégration des équations de fonctionnement est indépendant localement de p, c'est-à-dire :

$$\frac{\partial h_i \{p=p_o\}}{\partial p_o} = o \qquad \forall i.$$

Ceci signifie que la séquence de pas nécessaire pour intégrer F(Y, y', p, t) est la même que celle que l'on utiliserait pour intégrer F(Y, y', p+δp, t) = o.

Cette hypothèse se trouve justifiée dans la pratique.
En fait, si cette hypothèse n'était pas vérifiée, une variation δp autour d'une valeur nominale p_o modifierait le comportement du circuit de façon importante. Dans la pratique, on tente justement d'éviter ce type de situation.

Dans ces conditions,

$$\frac{\partial y'_n}{\partial p} = - \frac{Lo}{h\beta_o} \frac{\partial yn}{\partial p} + \sum_{i=1}^{q} a_i \frac{\partial yn-i}{\partial p} \qquad (II.41)$$

Reportant (II.41) dans (II.39), il vient :

$$(II.42) \qquad \frac{\partial Y_n}{\partial p} = -\left[\frac{\partial Fn}{\partial Yn} - \frac{Lo}{h_n \beta_o}\frac{\partial Fn}{\partial \dot{y}_n}\right]^{-1}\left\{\frac{\partial Fn}{\partial y'_n}\sum_{1}^{q} a_i \frac{\partial yn-i}{\partial p} + \frac{\partial F}{\partial p}\right\}$$

(II.42) nous permet d'évaluer $\frac{\partial Yn}{\partial p}$ et donc $\frac{\partial T}{\partial p}$, puis $\frac{\partial \phi}{\partial p}$.

Remarquons que la matrice $\left\{\frac{\partial F}{\partial Y} - \frac{Lo}{h\beta_o}\frac{\partial F}{\partial y'}\right\}$ est justement l'invariant obtenu dans la phase de traduction et utilisée dans l'analyse transitoire.
De même, la résolution du système linéaire (II.42) peut s'effectuer en utilisant la séquence de résolution utilisée dans l'intégration des équations de fonctionnement.
Les dérivées $\frac{\partial F}{\partial p}$, $\frac{\partial S}{\partial p}$, $\frac{\partial S}{\partial y}$, $\frac{\partial S}{\partial T}$ qui interviennent dans les expressions ci-dessus sont évaluées formellement au cours de la phase de traduction.

Nous pouvons comparer le volume de calcul exigé par cette méthode avec celle du réseau adjoint.

Soit n le nombre de pas nécessaire à l'intégration des équations de fonctionnement et k le nombre de paramètres de conception. Prenons comme mesure le nombre d'itérations de Newton avec une moyenne de 2 itérations par pas.

- Méthode ci-dessus : 2n + (n.k résolutions de systèmes linéaires creux - résolution codée -)

- Méthode du réseau adjoint : 2n + $n^{(1,2)}$ + (interpolations de matrices dans l'intégration du système adjoint).

Les volumes de calcul sont comparables mais cette méthode simplifie la phase de traduction puisque celle-ci reste sensiblement identique à la phase de traduction utilisée en simulation.

---

(1) - Le nombre de pas d'intégration pour le système adjoint est supposé égal à n. Il peut bien entendu être différent.

(2) - Selon certains auteurs, l'intégration du système adjoint ne demande qu'une itération de Newton par pas car on peut initialiser les itérations au "voisinage" de la solution compte tenu des résultats obtenus sur le système

Remarque : Le calcul des sensibilités est un cas particulier de cette méthode où ϕ est une variable du circuit et où il n'intervient pas de fonction de seuil.

Pour ne pas modifier le langage d'accès au système, la méthode a été mise en oeuvre pour des fonctions objectifs de la forme $\phi(p) = \phi(Y,p,t)$. Toutefois, les problèmes numériques y sont du même ordre et d'un point de vue strictement numérique, il suffirait d'ajouter au système une procédure d'évaluation d'intégrale.

Cette méthode a jusqu'ici donné satisfaction et a été utilisée avec succès par certains utilisateurs (en particulier le LETI).

III.3.4 - Aspect utilisation.

Le système d'optimisation réalisé doit être utilisé avec précaution pour obtenir des résultats cohérents et ne pas être conduit à un volume de calculs prohibitifs.

Nous avons rencontré dans les paragraphes précédents deux types de difficultés :

- le problème des minimums locaux, qui, dans une certaine mesure, peut être résolu en initialisant les paramètres de conception au "voisinage" d'un minimum global.

- le problème du minimum "pointu", définissant un fonctionnement du circuit qui ne peut être obtenu compte-tenu de l'imprécision sur la fabrication des composants. Nous avons vu que cette difficulté peut être résolue par l'utilisation conjointe de l'optimisation et de l'analyse statistique.

Un troisième problème important est celui du choix de la fonction objectif. Cette fonction doit véritablement refléter les exigences du concepteur, c'est-à-dire que celui-ci doit être sûr que la minimisation de cette fonction traduira le fonctionnement qu'il veut assigner au circuit.

D'autre part, elle doit être suffisamment simple pour ne pas prendre en défaut les algorithmes numériques et conduire à des coûts d'utilisation prohibitifs.

Le choix de telles fonctions pour des types de circuits particuliers est un domaine d'étude en lui-même qui échappe à notre compétence. Seule la connaissance des problèmes étudiés et une bonne expérience de l'utilisation du système permettent de choisir correctement ces fonctions et les contraintes éventuelles.

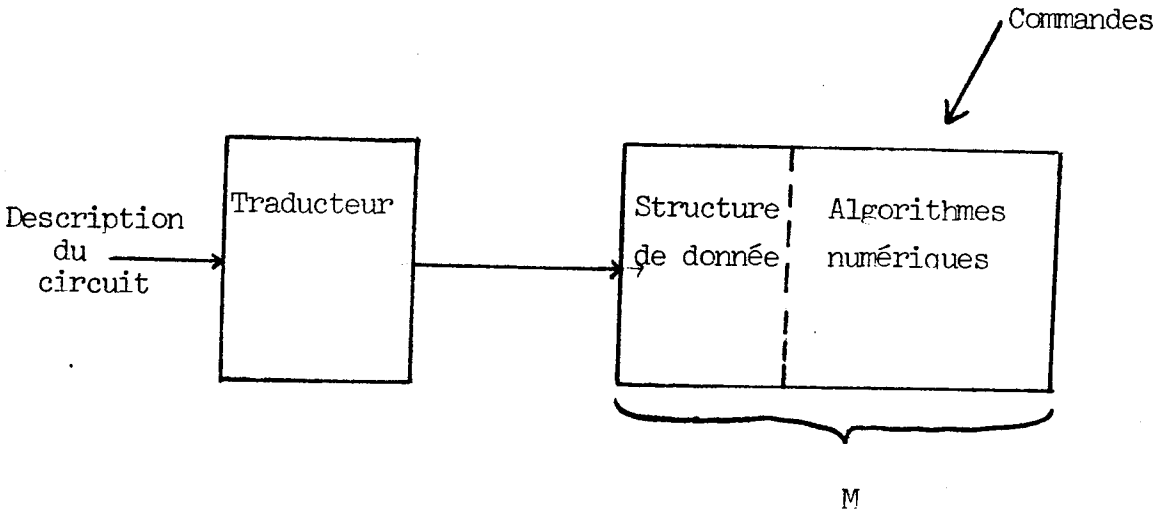
Dans une utilisation conversationnelle, l'utilisateur peut suivre chaque pas du processus d'optimisation et donc contrôler l'évolution des calculs. En particulier, il peut interrompre l'exécution à tout moment si les résultats sont soit suffisants, soit, au contraire, mauvais ou longs à obtenir. Bien entendu, il n'en est pas de même dans une utilisation en traitement par lots. Aussi, une simulation des réactions de l'utilisateur devant l'évolution du calcul a été réalisée par l'introduction de tests sur l'évolution de la fonction objectif, son gradient, et des paramètres d'optimisation [CAR 73].

S'il existe un grand nombre de systèmes de simulation de circuits, - en particulier aux Etats-Unis où chaque université se doit de posséder le sien - il n'en est pas de même des systèmes d'optimisation pour lesquels il n'existe quasiment pas de réalisation vraiment opérationnelle. La plupart des personnes travaillant dans le domaine de la C.A.O. en électronique et voulant dépasser le stade de la simulation, se sont tournées vers l'analyse statistique qui fait appel à des techniques beaucoup plus simples et dont la mise en oeuvre est beaucoup plus facile. En fait, l'application des techniques d'optimisation à la C.A.O. en électronique en est encore aujourd'hui au stade du laboratoire. Si elle répond à certaines préoccupations des électroniciens - et nous le pensons -, peut-être est-ce dû au fait que les techniques qui y sont utilisées ne sont encore ni suffisamment éprouvées, ni suffisamment fiables.

IV - Conclusion.

Nous avons examiné, dans ce chapitre, l'ensemble des algorithmes qui constituent la partie active de la machine M, ainsi que les caractéristiques essentielles du système d'optimisation.

Avec le chapitre I décrivant la phase de traduction, nous avons ainsi une vue d'ensemble sur les techniques utilisées dans le système IMAG3 dont la structure peut finalement se schématiser ainsi :



La machine M est ainsi constituée d'un ensemble d'algorithmes numériques, lesquels travaillent sur une structure de donnée générée par le traducteur.

Cette structure de donnée constitue une représentation des différentes variables et un codage permettant d'évaluer les équations de fonctionnement, la matrice de Jacobi de ces équations, les dérivées partielles intervenant en optimisation, et de résoudre un système linéaire creux JX = B.

Cette structure de donnée et ces algorithmes - comme nous le verrons dans le chapitre III - ont été choisis de manière à :

- étendre les possibilités du système,
- minimiser son coût d'utilisation.

D'autre part, la structure de données et les algorithmes permettent la simulation (et l'optimisation) d'un circuit électrique, mais sont théoriquement indépendants (1) de la nature de l'objet que l'on désire simuler.

---

(1) - Une certaine dépendance existe au niveau des paramètres choisis pour certains algorithmes numériques en fonction des équations décrivant un circuit électronique. Cependant, cette dépendance est locale.

Ils supposent seulement que le fonctionnement de cet objet puisse se décrire par un ensemble d'équations algébriques et différentielles couplées. Ceci signifie qu'ils peuvent être utilisés pour la simulation de tout objet pouvant se décrire par un schéma différentiel, si toutefois l'on dispose de traducteurs appropriés. Ils sont en particulier intéressants pour les phénomènes décrits par des systèmes différentiels "stiffs" (cinétique chimique, etc...).

De même, la partie du traducteur qui consiste en la manipulation formelle d'équations (calcul de dérivées partielles, etc...) et en la génération de la structure de données pour M, peut être utilisée. Seule la partie analyse du langage de description et écriture des équations de fonctionnement doit être reformulée.

CHAPITRE III

La troisième partie de ce document sera consacrée à situer IMAG 3 dans l'ensemble des programmes existants à l'heure actuelle en simulation et optimisation de circuits électroniques.

Nous utiliserons une étude récente effectuée par une équipe de l'Université de Floride animée par le Professeur James Bowers [Blatt 76, Bow 76]. Cette étude porte sur l'ensemble des programmes existants les plus connus.

Dans le premier paragraphe, nous schématiserons les résultats de cette étude pour les points qui paraissent les plus intéressants.

Les travaux ayant exercé l'influence la plus profonde dans ce domaine au cours des dernières années, sont certainement ceux de Hachtel, Brayton et Gustavson, qui se trouvent exposés dans leur article : "Sparse Tableau Approach to Network Analysis and Design" [Hacht 71].

L'idée est à l'opposé de l'approche par "variables d'états" utilisée dans IMAG 2.

En effet, contrairement à cette dernière où l'on tente d'écrire les équations de fonctionnement à l'aide d'un ensemble de variables minimums, ils proposent d'écrire ces équations en utilisant toutes les variables électriques du circuit. Ils obtiennent ainsi un système d'équations de dimension beaucoup plus élevée, mais aussi beaucoup plus "creux". C'est-à-dire que dans chaque équation intervient un nombre très faible de variables. C'est pourquoi cette approche a été nommée "approche du tableau creux" (Sparse Tableau Approach).

Ces travaux ont été repris en France pour aboutir au système ASTEC [HEYDE 74].

D'autre part, I.B.M. a développé le système ASTAP [Weeks 73] en s'inspirant largement de cette approche - mais en s'en écartant toutefois sur certains points.

Nous examinerons succintement ces approches, puis nous mettrons en évidence leurs différences avec IMAG 3.

Les auteurs d'ASTAP ayant publié des résultats de simulation sur un exemple, nous pourrons effectuer une comparaison avec IMAG 3.

D'autre part, nous montrerons les progrès réalisés depuis les premières versions d'IMAG 2 sur un exemple.

Nous ne pensons pas qu'il soit utile de surcharger ce document d'exemples particuliers de simulation ou d'optimisation. En effet, ceux-ci existent dans divers articles ou rapports sur IMAG 2 et IMAG 3 dont on pourra trouver une liste dans la bibliographie, (et bien entendu chez les utilisateurs du programme).

Aussi, nous nous contenterons de deux exemples très simples que le lecteur pourra éventuellement calculer "à la main".

I - Caractéristiques essentielles de divers programmes disponibles.

Le contenu de ce chapitre est tiré de l'étude de Bowers et Coll : "A Survey of Computer-Aided Design and Analysis Programs".

Ceux-ci ont étudié 16 programmes différents et ont fait un inventaire de leurs possibilités.

Les tableaux suivants indiquent les caractéristiques générales de ces programmes.

	Origine	Taille max. des circuits	Possibilités d'analyse
ANP 3	Danemark	n = 40, b = 100	- circuits linéaires
ASTAP	U S A	M	- AC, DC, transitoire, analyse statistique, analyse de pannes (circuits non linéaires)
BELAC	U S A	M	- AC, DC, transitoire, sensibilité, analyse statistique, plus mauvais point de fonctionnement, transformée de Fourier rapide, optimisation
CIRCUS 2	U S A	M	- DC, transitoire
COD	U S A	30 variables	- optimisation, DC, Transitoire (diodes et transistors seulement)
IMAG 3	France	M	- AC, DC, transitoire, sensibilité, optimisation, circuits non linéaires
ISPICE	U S A	M	- AC, DC, transitoire, analyse de température
ITAP	Angleterre	n = 48	- DC, et transitoire (diodes et transistors seulement)
ITRAC	Angleterre	n = 50, b = 200	- DC et transitoire (circuits non linéaires)
NAP 2	Danemark	n = 50/500 e = 195/1500	- DC, AC, transitoire, analyse de bruit, sensibilités, optimisation plus mauvais point de fonctionnement.
NET 2	U S A	M	- DC, AC, transitoire, radiation, analyse statistique, optimisation.
REDAP 30	Angleterre	n = 40, b = 80	- transitoire
SATAN	Angleterre	e = 30 ou 80	- DC et transitoire
SPICE 2	U S A	M	- AC, DC, transitoire, sensibilité, analyse de température, bruit, transfert.
SUPER-SPECTRE	U S A	n = 301, e = 300	- DC, transitoire, fonctions de transfert, utilisation de modèles digitaux, AC, DC sensibilité, optimisation en DC, plus mauvais point de fonctionnement et analyse statistique.
UCCAP	U S A	n = 400	- AC, DC, transitoire, plus mauvais point de fonctionnement, pannes en AC et DC.

n : nombre de noeuds

b : nombre de branches

e : nombre d'éléments

M : dépend de la place mémoire disponible (le programme doit en tenir compte).

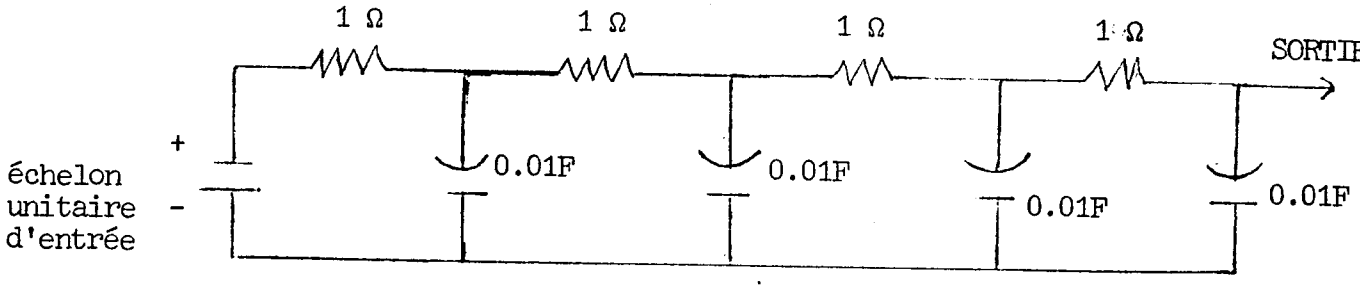
	Avantages	Inconvénients
ANP 3	bonne rapidité excellente précision	Circuits linéaires seulement Possibilités limitées d'arrêt en analyse transitoire. Une seule sortie par simulation.
ASTAP	possibilité de circuits complexes et de grandes tailles. Bonne rapidité.	en location seulement
BELAC	bonne précision	Lent - pas de conditions initiales, pas de modèle créé par l'utilisateur.
CIRCUS 2	possibilités de modèles générés par l'utilisateur et de sous-programmes FORTRAN pour créer à volonté des éléments non linéaires	Lent - Seulement analyse en DC et transitoire.
COD	pratique pour spécifier les contraintes sur les éléments et les réponses des circuits pour la corrélation entre les paramètres des circuits.	Nécessité d'écrire des équations pour toutes réponses désirées.
IMAG 3	bonne précision. Création de modèles à volonté. Fonction FORTRAN. Fonction logique IF	en location seulement pas d'analyse de tolérances
ISPICE	coût modéré création de modèles	les éléments non-linéaires ne peuvent être fonction de courants ou de tensions de circuits. En location seulement.
ITAP	très rapide, précision passable	Uniquement DC et transitoire. Excitation impulsionnelle et sinusoïdale uniquement. Taille des circuits limitée à 48 noeuds Impossible de créer des modèles.
ITRAC	analyse transitoire de rayonnement.	Analyse DC et transitoire seulement. Taille des circuits limitée Les modèles doivent être définis mathématiquement.
NAP 2	très bonne précision. Création d'éléments non-linéaires à volonté	Relativement lent.
NET 2	possibilités à l'utilisateur de créer des modèles.	Assez lent. Dans certains cas, manque de précision.
REDAP 30	programme très souple bonne maintenance	Problèmes de convergences pour les grands circuits. Très lent.

	Avantages	Inconvénients
SATAN	possède des méthodes d'intégration implicites et explicites.	Uniquement DC et transito Parfois précision insuffi Impossibilité de créer de modèles. Taille limitée.
SPICE 2	très rapide	Parfois précision insuffi Problème de convergence
SUPER-SPECTRE	possibilité de fonctions de transfert logique - digitale et modèles mécaniques. Possibilité de créer des modèles Bonne précision.	Relativement lent.
UCCAP	rapide - possibilité de créer des sous-circuits.	Parfois précision insuffis Location seulement.

Bien entendu, les listes d'avantages et d'inconvénients ne font pas apparaître certains détails importants, mais cette étude permet d'avoir une bonne idée de l'ensemble des programmes existants actuellement.

Il est intéressant de reproduire ici l'étude faite pour ce qui concerne la précision. Cette étude était construite à partir d'une analyse transitoire de deux circuits tests dont on comparait la réponse calculée et la réponse théorique.

* <u>Premier circuit test</u>

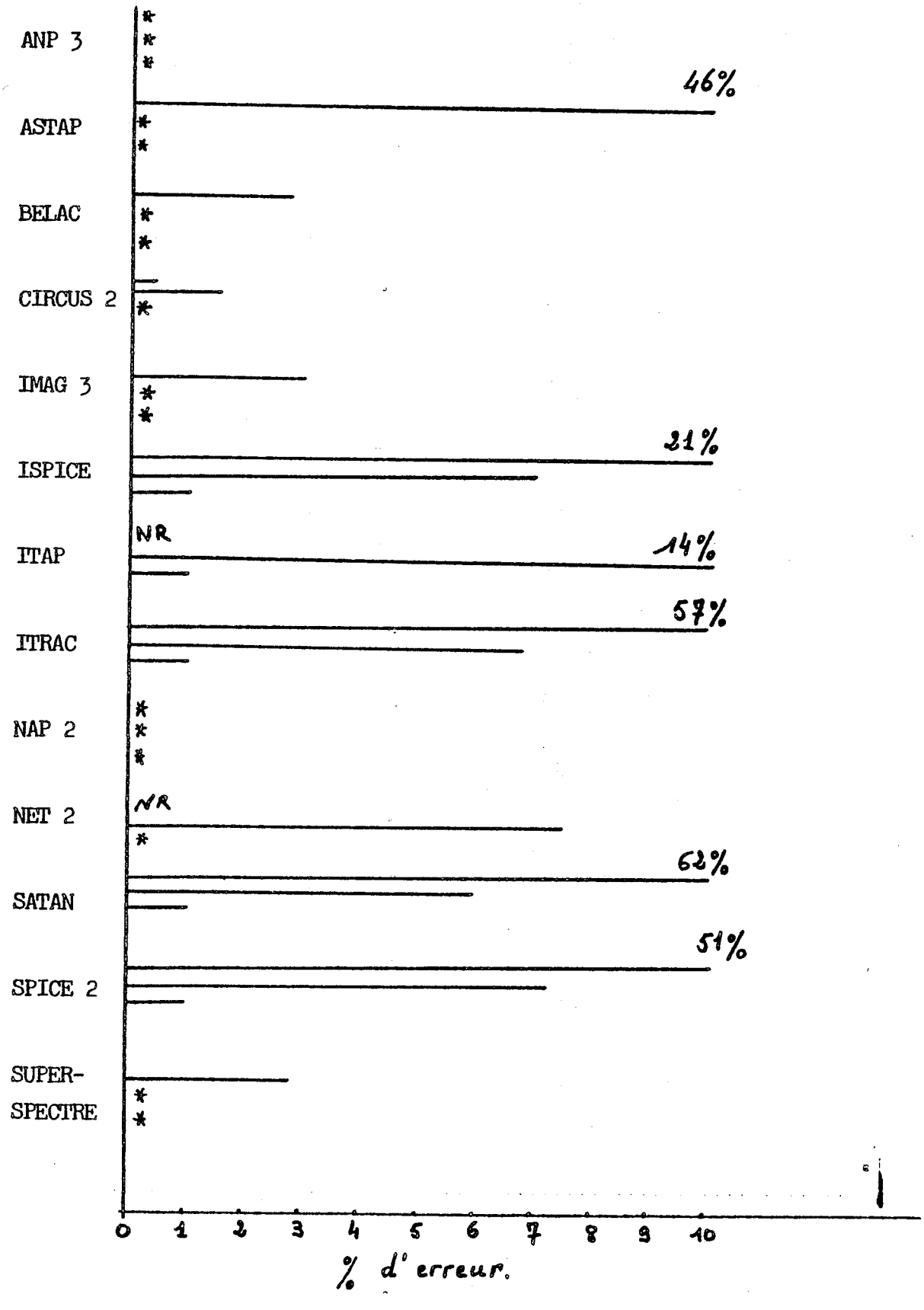


3 relevés sont effectués : 1, 10 et 40 millisecondes.

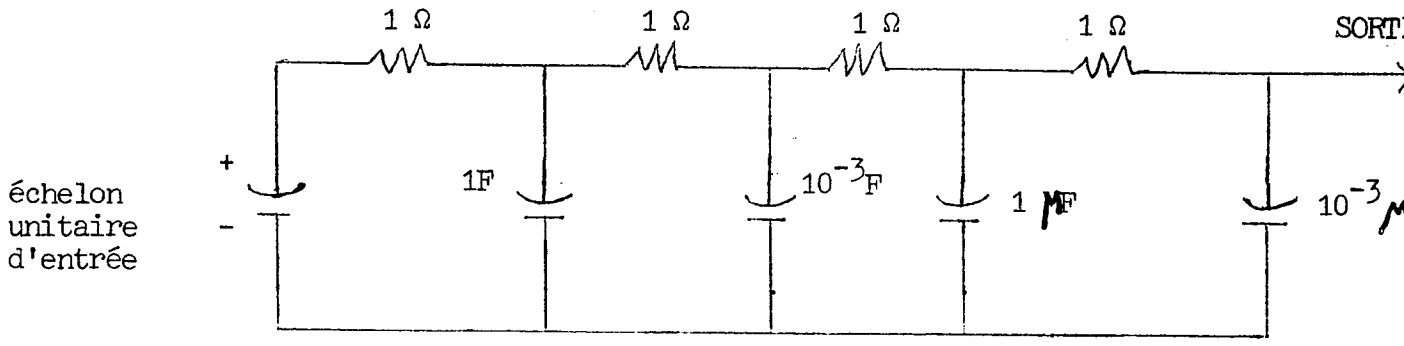
Les traits successifs correspondent à ces 3 relevés.

* indique une erreur < 0.10 %

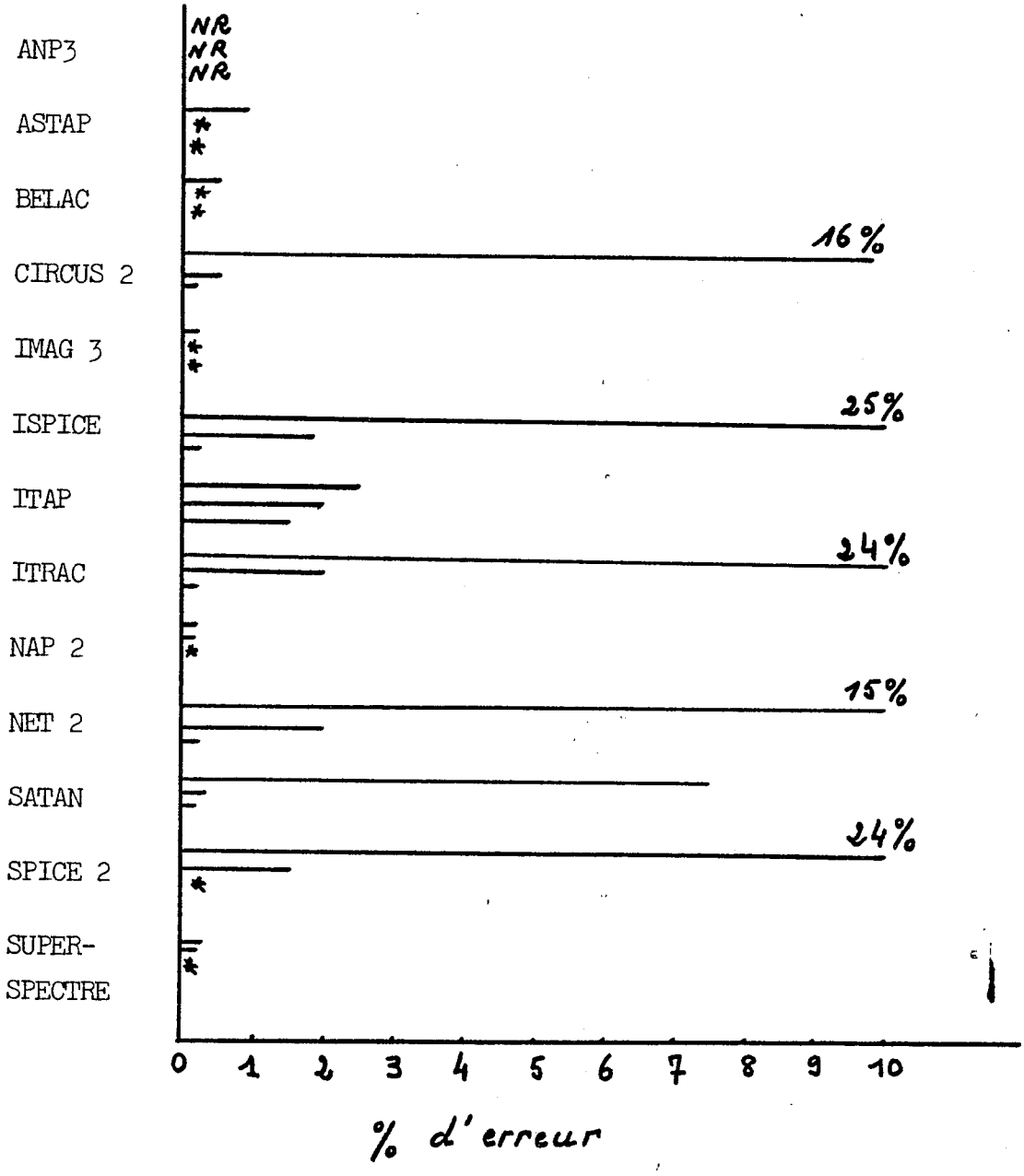
NR indique que la lecture n'a pas été possible.



Deuxième circuit test.



Les auteurs obtiennent les résultats suivants : (0.1,1 et 4 sec.)



IMAG 3 apparaît ici particulièrement bien placé en ce qui concerne la précision. On peut remarquer également que les programmes donnant les résultats les plus précis sont généralement :

- soit des programmes peu rapides
- soit des programmes aux possibilités limitées

- par exemple aux circuits linéaires comme ANP 3 -.

Il semble donc qu'IMAG 3 réalise un excellent compromis entre :

- la généralité
- la précision
- la rapidité (comme on le verra dans la comparaison avec ASTAP au paragraphe 3.

II. Approche du tableau creux - {Sparse tableau approach}

Prenant une direction tout à fait opposée à l'approche par variables d'état dont le but est d'écrire les équations de fonctionnement avec un nombre minimum de variables et de les formuler sous la forme canonique

$$\frac{dy}{dt} = F(Y, t),$$

Hachtel, Brayton et Gustavson considèrent toute l'information relative au circuit sous une forme non réduite.

Les variables électriques inconnues d'un réseau sont les tensions aux noeuds V, les courants et tensions de branches i et v, les énergies q (charges ou flux) stockées par les éléments réactifs.

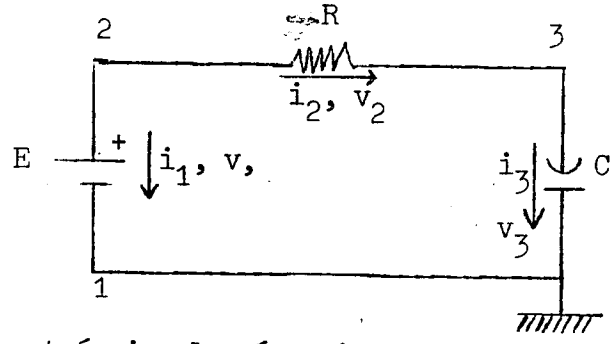
Les équations du circuit peuvent s'écrire :

$$F \times \begin{pmatrix} V \\ i \\ v \\ q \end{pmatrix} = f$$

où F est un opérateur algébro-différentiel que l'on peut représenter sous forme matricielle.

Exemple :

Soit un circuit RC



On peut écrire les équations suivantes :

$$\left.\begin{aligned} i_1 + i_2 &= 0 \\ -i_2 + i_3 &= 0 \end{aligned}\right\} \text{ lois des noeuds}$$

$$\left.\begin{aligned} V_2 - v_1 &= 0 \\ V_1 - V_2 - v_2 &= 0 \\ V_3 - v_3 &= 0 \end{aligned}\right\} \text{ lois des mailles}$$

$$\left.\begin{aligned} E - v_1 &= 0 \\ R - i_2 - v_2 &= 0 \\ C\, v_3 - q_3 &= 0 \end{aligned}\right\} \text{ relations de branche}$$

$$i_3 - \frac{dq_3}{dt} = 0 \quad \} \text{ équations dynamiques}$$

Dans ce cas, l'opérateur F prend la forme = (les inconnues étant $(V_2, V_3, i_1, i_2, i_3, v_1, v_2, v_3, q_3)$}.

$$
F = \begin{array}{c}
\begin{array}{ccccccccc} V_2 & V_3 & i_1 & i_2 & i_3 & v_1 & v_2 & v_3 & q_3 \end{array} \\
\left[\begin{array}{ccccccccc}
 & & +1 & +1 & & & & & \\
 & & & -1 & +1 & & & & \\
+1 & & & & & -1 & & & \\
+1 & -1 & & & & & -1 & & \\
 & +1 & & & & & & -1 & \\
 & & & & & -1 & & & \\
 & & & R & & & -1 & & \\
 & & & & & & & C & -1 \\
 & & & & 1 & & & & -\frac{d}{dt}
\end{array}\right]
\end{array}
$$

L'application de l'opérateur F conduit finalement au système algébro-différentiel :

$$G\left(X, \frac{dX}{dt}, u, t\right) = 0 \qquad \text{(III.1)}$$

que l'on peut intégrer par une méthode d'intégration analogue à celles envisagées dans le chapitre II. Comme nous l'avons déjà vu, le point crucial de ces méthodes est la résolution du système linéaire

$$\left\{\frac{\partial G}{\partial (u,x)}\right\} \left\{\begin{matrix}\Delta u \\ \Delta X\end{matrix}\right\} = B \qquad \text{(III.2)}$$

Dans le système (III.1), la matrice $J_G = \left\{\frac{\partial G}{\partial (u,x)}\right\}$ est extrêmement creuse. Aussi, les auteurs ont-ils développé un algorithme de résolution de (III.2) que nous avons exposé dans le chapitre I.

Le but de cette approche est bien entendu d'analyser le circuit en effectuant un nombre minimum d'opérations.

En effet, résoudre le système linéaire (III.2) consiste à éliminer successivement les inconnues par combinaison des équations selon un ordre approprié pour minimiser le nombre d'opérations correspondantes.
Or, c'est déjà un processus d'élimination entre les relations élémentaires d'Ohm et de Kirchhoff qui fournit les équations réduites au sous-ensemble des variables d'états : Y_E.

Choisir ces dernières comme base de calcul serait donc équivalent à optimiser l'élimination entre les équations élémentaires avec la contrainte d'éliminer en premier, dans un ordre déterminé, les variables du circuit n'appartenant pas à Y_E.

Traiter directement sans contrainte, les équations de base, autorisera donc une meilleure performance de l'ordre résultant, sans imposer les servitudes liées à l'obtention des équations réduites (non-linéaires fonctions des variables d'états par exemple).

Théoriquement, cette approche est donc supérieure à l'approche par variables d'états classique - c'est-à-dire conduisant à un système différentiel sous la forme canonique.-.

Cependant, à notre avis, l'inconvénient majeur de cette méthode est qu'elle repose entièrement sur l'algorithme de factorisation L/U qui est un algorithme numérique. Or, lors de la génération de la séquence d'opérations destinée à réaliser cette factorisation, seules les valeurs numériques initiales sont connues. Ceci est vraisemblablement la source d'un certain nombre de difficultés numériques.

Aussi, nous pensons que des contraintes topologiques doivent être introduites dans eet algorithme.

D'autre part, il semble que la génération de cette séquence codée soit relativement coûteuse (cf.exemple d'ASTAP). Ceci est vraisemblablement lié à la complexité de l'algorithme et à la dimension importante de la matrice de Jacobi.

Ce coût exclut pratiquement le choix d'une nouvelle séquence codée de factorisation en cours de calcul.

Ces travaux ont eu le mérite d'introduire concrètement la notion de "nombre minimum d'opérations" nécessaires à l'analyse d'un circuit et ont donné une impulsion nouvelle à de nombreuses recherches dans ce sens.

Signalons qu'en France, un programme a été développé : ASTEC [Heyd 74] selon ces principes, et semble avoir donné des résultats intéressants.

III. Approche utilisée dans ASTAP [WEEKS 73].

ASTAP est un programme général de simulation de circuits électroniques permettant également d'effectuer une analyse de tolérance de manière statistique.

Développé par I.B.M., il utilise à la fois des concepts topologiques et une variante de la méthode du tableau.

Ainsi, alors que dans l'approche du tableau, on utilise comme variables les tensions et les courants de chaque élément, les tensions de noeuds, les charges capacitives et les flux selfiques, on utilise ici seulement les tensions et courants de chaque élément.

Le lecteur intéressé par les algorithmes utilisés pourra se reporter à [Weeks 73].

Les concepts topologiques utilisés sont ceux de F-coupures et de F-circuits permettant d'écrire les équations de mailles et de noeuds sous la forme :

$$[-Q^t, U_c] \begin{matrix} V_B \\ V_c \end{matrix} = 0 \qquad \text{(III.3)}$$

$$[U_B, Q] \begin{matrix} I_B \\ I_c \end{matrix} = 0 \qquad \text{(III.4)}$$

ou : les V_B et V_c sont des tensions d'éléments

- les I_B et I_c sont des courants traversant les éléments.

- U_B et U_c sont les matrices unitaires

- Q la matrice des F-coupures.

Si N est le nombre d'éléments du circuit, (III.3) et (III.4) fournissent N équations, les N équations restantes sont fournis par les caractéristiques (V, i) appliquées à chacun des éléments :

$$I_i = C_i \frac{dV_i}{dt} \quad : \text{capacité}$$

$$V_i = L_i \frac{dI_i}{dt} \quad : \text{self}$$

(III.5)

$$V_i = \sum_j L_{ij} \frac{dI_j}{dt} \quad : \text{mutuelle inductance}$$

$$I_i = J_i(t)$$
$$V_i = E_i(t) \qquad \text{sources indépendantes}$$

$$I_i = f_i(V_1, V_2 \cdots, V_2, I_1, I_2 \cdots, I_{i-1}, I_{i+1}, \cdots I_e)$$

$$V_i = F_i(V_1, V_2 ;;; V_{i-1}, V_{i+1} \cdots, V_e, I_1, I_2 \cdots, I_e) \qquad \text{sources lié}$$

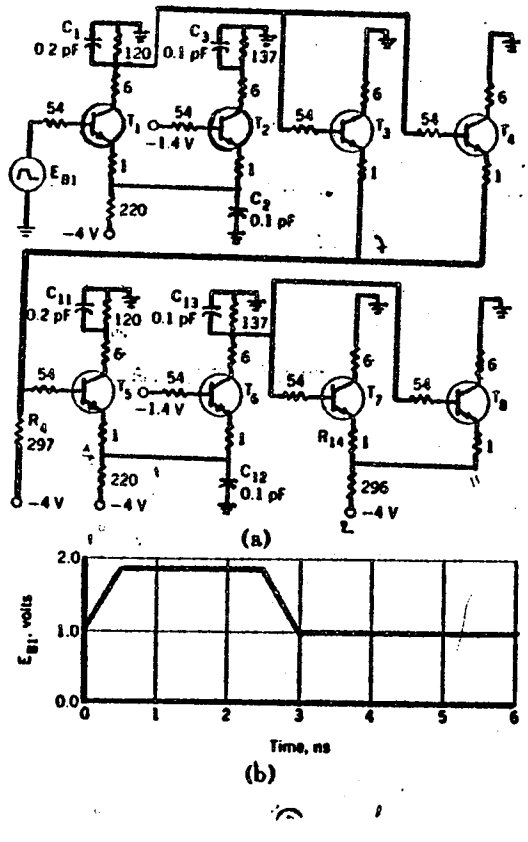
L'ensemble des équations (III.3), (III.4) et (III.5) caractérisent le fonctionnement du circuit. Elles sont résolues en utilisant des algorithmes adaptées aux matrices creuses.

Les auteurs d'ASTAP ont publié les résultats de la simulation d'un circuit logique qui nous a permis d'effectuer une comparaison de ses performance avec celles d'IMAG 3.

IV. Etude d'un circuit logique avec ASTAP et avec IMAG 3.

IV.1. Description du circuit.

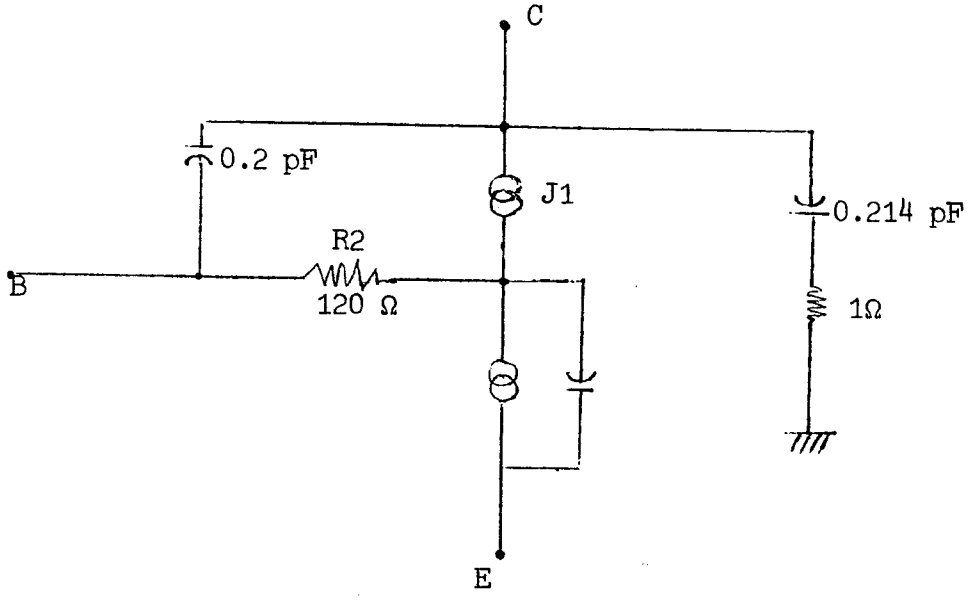
Considérons le circuit logique ci-dessous constitué de 8 transistors déjà utilisé dans les tests du système ECAP 2 et repris par les auteurs d'ASTAP.



Tiré de:IEEE Journal of Solid -State Circuit,Vol.SC6, N.4,august 71

Le circuit est attaqué par la source de tension E_{B1} représenté sur la figure (b) ci-dessus.

Le schéma équivalent permettant de représenter les transistors est le suivant :



Dans ces conditions, le circuit contient 60 noeuds, 105 branches et 30 éléments réactifs se traduisant par 30 équations différentielles. Quand l'analyse transitoire de ce circuit - de 0 à 5 ns - est effectuée en utilisant une méthode explicite telle que la méthode de Runge-Kutta d'ordre 4, une estimation à partir d'une analyse partielle montre qu'il faut environ 800.000 pas d'intégration pour effectuer une analyse complète.

C'est dire que le coût de cette analyse est prohibitif.

IV.2. Résultats.

Pour mettre en évidence l'augmentation du coût d'utilisation du système en fonction de l'augmentation de la dimension du circuit, nous étudions successivement le couplage de 1, 2, 3, 4 et 5 circuits de base.

			IMAG 2 V. 73	IMAG 3			ASTAP (4)		
Cellules	éléments	équa. dif.		Compil.	analyse	Total	Compilation	Analyse	Total
1	105	30	27	9	8	17	63.4	15.9	79.4
2	209	60	90	15.5	15.5	31	120.5	24	144.5
3	313	90	190	30	20	50	172	33	205
4	417	120	(2)	(3)	(3)	71	238	43	281
5	521	150	(2)	(3)	(3)	100	323.7	53.3	377

(1) Les temps indiqués en secondes correspondent à des passages sur un IBM 360/65 avec le système OS/MVT.
Les temps de compilation correspondent à :

- l'analyse syntaxique de la description
- la mise en équations
- la génération des différents invariants utilisés par la phase d'analyse :
- au calcul d'un état stable (résolution d'un système algébrique)
- au calcul d'une réponse en transitoire (intégration du système).

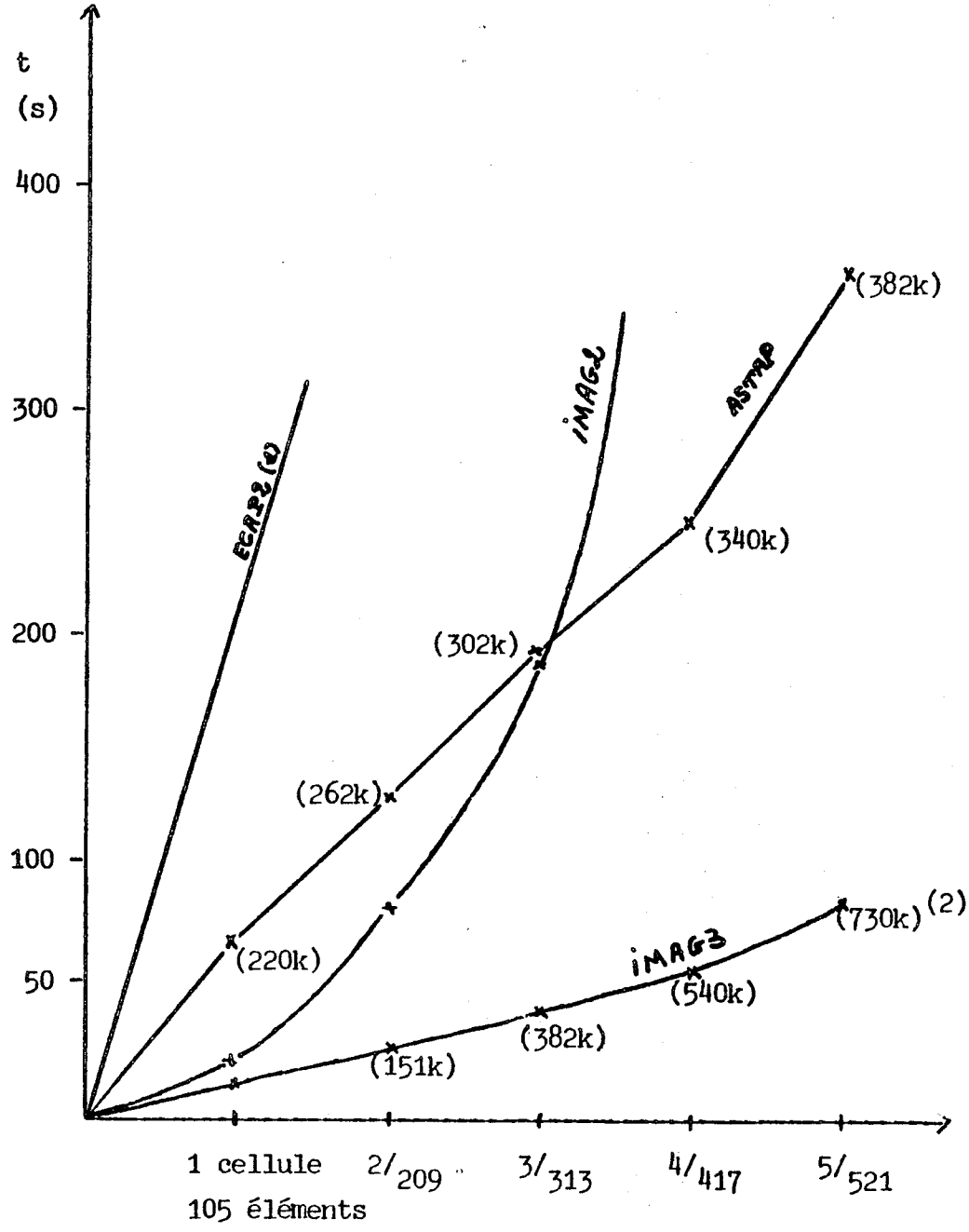
(2) Les passages correspondant n'ont pas été effectués pour des raisons évidentes de coûts.

(3) Ces temps n'ont pas été mesurés.

(4) Les temps correspondants sont tirés de [Weeks 73] (en supposant un rapport de rapidité de 5.4 entre le 360-65 et le 360-85).

La précision des résultats obtenus avec les systèmes IMAG2,IMAG3 et ASTAP est comparable. Tout les résultats se situent dans une fourchette acceptable par l'electronicien. Il est difficile de dire quelle est - sur cet exemple - la précision obtenue, car la réponse théorique n'est pas connue.

La courbe d'évolution des coûts pour les différents systèmes est la suivante :



(1) ECAP 2 est le système qui a précédé ASTAP dans la série des programmes de simulation de circuits proposés par IBM.

(2) Les nombres entre parenthèses indiquent l'espace mémoire nécessaire au traitement du circuit.

III.3. Conclusions/

L'examen des résultats précédents suggère un certain nombre de remarques :

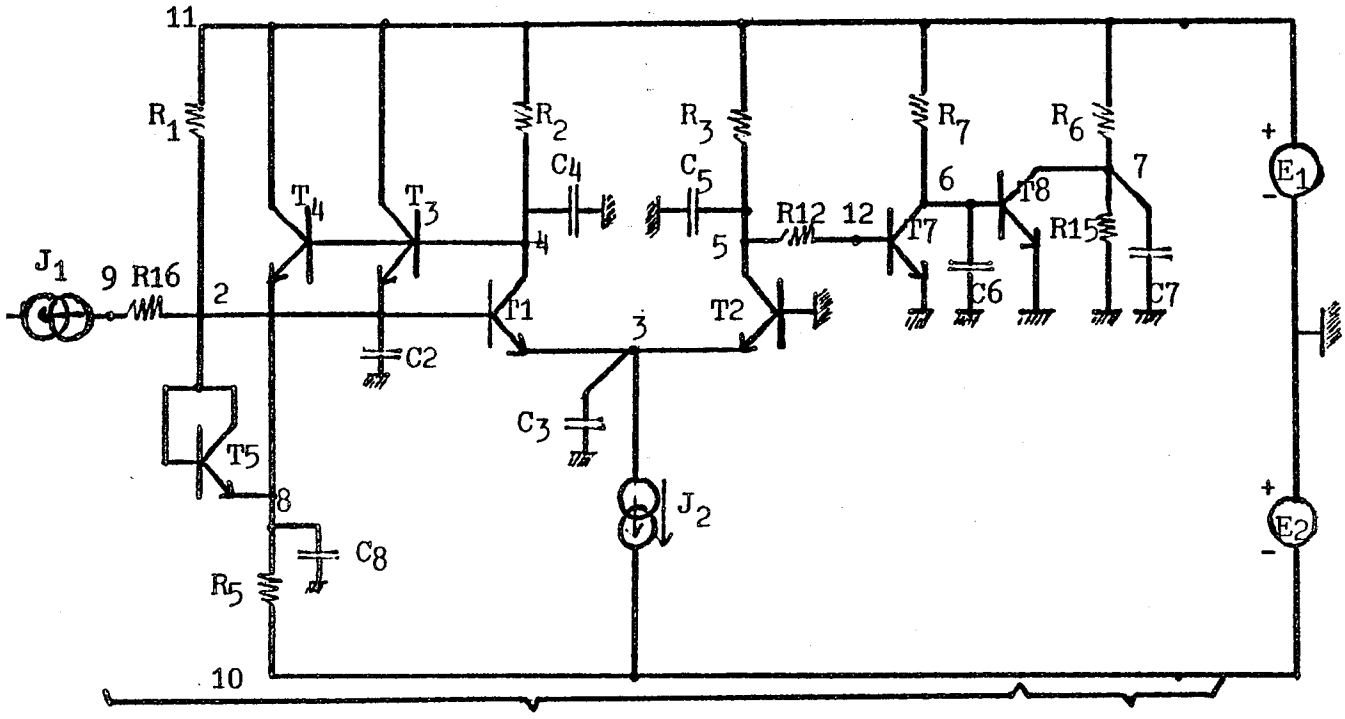
(a) Contrairement à IMAG 2, les coûts d'utilisation des systèmes IMAG 3 et ASTAP croissent de manière sensiblement linéaire en fonction de la dimension du circuit. Ceci est évidemment dû à l'utilisation dans les deux derniers systèmes de techniques de résolution de systèmes linéaires creux.

(b) L'espace mémoire nécessaire croît rapidement dans IMAG 3. Ceci est dû au fait que la matrice de Jacobi est stockée sous la forme d'un tableau carré. On constate dans IMAG 3 un équilibre au niveau des coûts entre la phase d'analyse et c'est ce que nous avons tenté d'obtenir dans le développement d'IMAG 3.
Il semble en effet qu'une optimisation trop poussée du coût de la phase d'analyse se traduise par une augmentation sensible du coût de la phase de compilation.

(c) Si les coûts d'analyse sont comparables, les temps de compilation sont beaucoup plus importants avec ASTAP qu'avec IMAG 3. Ceci nous semble lié au fait que la dimension du système d'équations associé est plus importante dans ASTAP, entraînant, pour rendre cette approche efficace au cours de l'analyse, l'utilisation d'un algorithme de recherche des pivots pour la factorisation L/U plus sophistiqué.
Il semble dont - sur cet exemple - que les équations de fonctionnement que nous avons adoptées et les traitements qu'elles impliquent, se traduisent par une meilleure efficacité que les solutions adoptées dans ASTAP.

V. Evolution des performances des systèmes IMAG 2 et IMAG 3.

Pour mettre en évidence les progrès réalisés au cours du développement des systèmes IMAG 2/3, nous considérons un circuit de 8 transistors bipolaires fourni par le CENG-LETI :



Les caractéristiques de ce circuit sont les suivantes :

- nombre de noeuds : 39
- nombre de branches : 120
- nombre d'équations différentielles : 30

On trouvera ci-dessous la description du circuit exprimée à l'aide du langage d'entrée d'IMAG 3.

```
desc**
**AL4 RESEAU ******11-2-74 $
MODELE:TINT(1,2,3)K,IE0,IC0,AN,AI,NE,NC,IES,ICL,CTE0,CTC0,CCAI,
RB,RBB,RC,RCC,TE,TC,TL,KSU$
J1(5,1)$  J2(4,5)$ J3(6,1)$
V1(5,1) $ V2(5,4) $ V3(6,1)$ V4(2,1)$
V5(3,1)$ C1(1,5)$ C2(5,4)$
C3(1,6)0.1P$
R1(2,6)=RB $ R2(6,5)=RBB $ R3(3,4)=RC+RCC $
KNE=K/NE$
KE=EXP(V1*K)-1 $ KES=EXP(V3*KNE)-1 $
J1=IE0*KE $
J2=J1*AN $
J3=IES*KES $
C1=CTE0 $
C2=CTC0  $$
MODELE:TINTS(1,2,3) K,IE0,IC0,AN,AI,NE,NC,IES,ICL,CTE0,CTC0,CCAI,
RB,RBB,RC,RCC,TE,TC,TL,KSU,AP $
J3(6,1) $  J4(6,7) $ J5(7,1) $  J1(5,1) $ J2(4,5)$
V1(5,1) $ V2(5,4) $ V3(6,1) $ V4(6,7) $
R1(2,6)=RB $ R2(6,5)=RBB $  R3(3,7)=RC $ R4(7,4)=RCC $
C1(1,5)$ C2(5,4) $  C3(6,7) $ C5(1,6)0.1P $
KNE=K/NE$ KNC=K/NC$
KE=EXP(V1*K)-1 $  KES=EXP(V3*KNE)-1 $
KC=EXP(V2*K)-1 $  KCL=EXP(V4*KNC)-1 $
A11=IE0*KE  $  A12=A21*AI $ A22=A11*AN $
A21=IC0*KC $  J1=A11-A12 $ J2=A22-A21 $ J3=IES*KES $
J4=ICL*KCL $ J5=(A21+J4)*AP $
C1=CTE0 $ C2=CTC0*KSU $C3=CTC0*(1-KSU) $
$
TYPE:
'TA'(TINT)  38.8,4.84E-16,7.23E-16,0.986,0.41,1.53,1.41,
2.315E-14,5.75E-14,2.1P,1.5P,2.9P,13,40,60,21,6E-10,12E-9,12E-9,0.42$
'05TA'(TINT)  38.8,2.68E-16,3.93E-16,0.986,0.41,1.53,1.41,1.45E-14,
4.32E-14,1.05P,0.82P,1.8P,24,83,110,35,6E-10,12E-9,12E-9,0.41 $
'TSA'(TINTS) 38.8,4.84E-16,7.23E-16,0.986,0.41,1.53,1.41,2.31E-14,
5.75E-14,2.1P,1.5P,2.9P,13,40,60,21,6E-10,12E-9,12E-9,0.42,0.909 $
'2TSA'(TINTS) 38.8,9.24E-16,1.38E-15,0.986,0.41,1.53,1.41,4E-14,
8.56E-14,4P,2.8P,3.7P,7,24,33,10,6E-10,12E-9,12E-9,0.45,0.909$$
** CIRCUIT ** $
TINT1(3,2,4)'TA' $
TINT2(3,1,5)'TA' $
TINT3(2,4,11)'05TA' $
TINT4(8,4,11)'05TA' $
TINT5(8,2,2) '05TA' $
TINTS7(1,12,6) 'TSA' $
TINTS8(1,6,7) '2TSA' $
R1(11,2) 30K $ R2(11,4)3342 $ R3(11,5) 3230 $
R5(8,10) 3342 $ R7(11,6) 4345 $ R8(11,7)1.5K $
R12(5,12) 27.6 $ R15(7,1)3.3K $ R16(2,9) 22)$ R103(3,1)1E9 $
R102(2,1) 1E9 $
E1(1,11)5 $ E2(10,1) 5 $ J1(9,1)'PULSE1'$ J2(3,1) 2.7324ML $
C2(2,1) 3.05P $ C3(3,1) 1.06P $ C4(4,1)1.24P $ C5(5,1)2.31P$
C6(6,1)1.30P $ C7(7,1)8.3P $  C8(8,1)0.63P $
  V7(7,1) $
R43(4,3)25K$ R53(5,3)25K $ $
TABLE:'PULSE1'1N/0,3N/0.4ML,30N/0.4ML,32N/0 $$$

/
fin
R; T=7.35/9.08 10:06:10
```

Les passages - sur IBM 360-65, système OS/MVT - obtenus à partir de différentes versions d'IMAG 2/3 sont résumés dans le tableau ci-dessous :

	IMAG 2 Version 30/10/70	IMAG 2 Version 1/7/72	IMAG 2 Version 1/9/73	IMAG 3 expérimental 74 (3)	IMAG 3
temps total compilation + analyse (1)	> 300 sec (2)	70,98 sec	42.45 sec	27 sec	17 sec

(1) Pour une analyse en continu (sans définir de conditions initiales) suivie d'une analyse en transitoire.

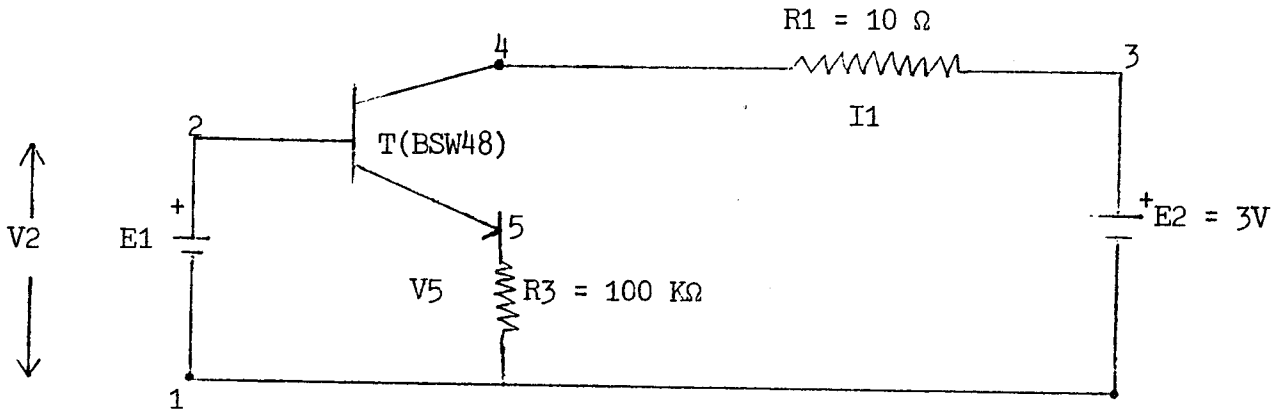
(2) Dépassement des 5 minutes allouées après les 7/8 du calcul de la réponse transitoire. On peut donc estimer le temps total à 340 s environ.

(3) Dans cette version d'IMAG 3, la phase de compilation n'avait pas sa forme définitive. En particulier, la résolution du système linéaire AX = B n'était pas entièrement codée et un certain nombre de redondances dans l'analyse topologique n'avaient pas été supprimées.

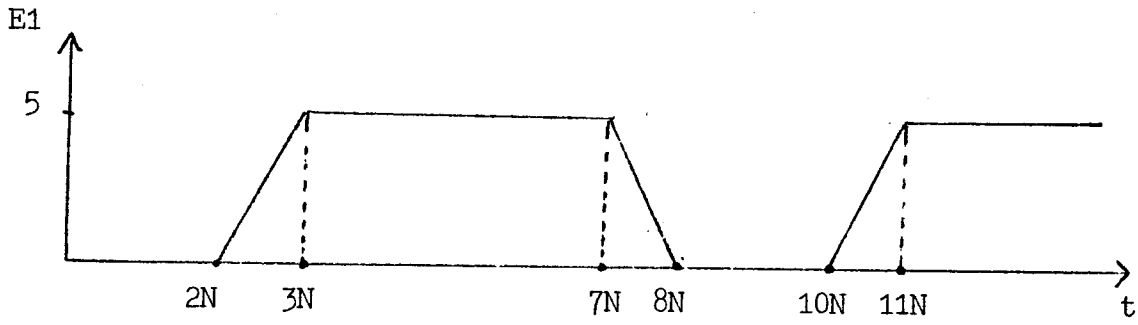
On a affaire ici à un circuit de taille "moyenne". Si la taille du circuit augmente, le gain apporté par IMAG 3 est encore plus apparent puisque la croissance du coût est linéaire dans IMAG 3, alors qu'elle est en N^3 dans IMAG 2.

VI. Exemple d'un calcul en sensibilités.

Considérons le circuit suivant :



E1 :



Nous étudions la réponse transitoire de 0 à 30 N (nanosecondes) et sur cet intervalle, nous souhaitons connaître :

$$\frac{\partial\,(V5,\ V2,\ I1)}{\partial\,(E2,\ R1,\ R3)}$$

On trouvera dans les pages suivantes les résultats obtenus à partir d'une console connectée au 360/67 par l'intermédiaire du système CP/CMS.

```
GO
EXECUTION BEGINS...

  DOIS-JE ETRE BAVARD ?      OUI/NON
oui

 ***NOM DU CIRCUIT ?***...OU COMMANDE STOP(T)...
boxcar
ADRESSE D'INTERRUPTION : [illegible]

/
 lasc**
** boxcar reseau
MODELE:T(1,2,3) EPS,KT,IE,IC,ALPHAN,ALPHAI,IEO,ICO,TE,TC,CTE0,
VPHIE,KE,CTC0),VPHIC,KC,R1,R2$
APPELLATION:EPS(-1,1)$
R1(2,4)$R2(5,3)$
J1(SELON(EPS)(4,1)(1,4))$
J2(SELON(EPS)(4,5)(5,4))$
V1(J1)$ V2(J2)$
C1(J1)$ C2(J2)$
BE=KT/IE$ BC=KT/IC$
AE=EXP(BE*V1)$ AC=EXP(BC*V2)$
IDE=IEO*(AE-1)$ IDC=ICO*(AC-1)$
J1=IDE-ALPHAI*IDC$
J2=IDC-ALPHAN*IDE$
BTE=BE*TE*IEO$ BTC=BC*TC*ICO$
C1=BTE*AE+CTE0/((VPHIE-V1)**KE)$
C2=BTC*AC+CTC0/((VPHIC-V2)**KC)$$
E1(1,2) 'PULSE'$
R1(3,4)1K$C1(1,5)2P$E2(1,3)[illegible]$T1(5,2,4) 'BSW48'$ [illegible]
V2(2,1)$V5(5,1)$$
TYPE:'BSW48'(T)-1,[illegible],1.33,1.05,[illegible],0.1,3P,2.[illegible],0.5[illegible],0.[illegible],
5.3P,1,0.5,1P,1.5,0.33,5[illegible],[illegible]$
TABLE:'PULSE'2.[illegible],[illegible],[illegible],[illegible],[illegible],[illegible],[illegible],[illegible],[illegible],
[illegible],[illegible],[illegible],[illegible],[illegible],[illegible],[illegible],[illegible],[illegible],[illegible],[illegible]
[illegible]$$$

/
tran

***DONNEES?***
[illegible] sortie:v*,i*$ sens:e2,r1,r3$$

 T              V2              V5              I1

 **** REPRISE ***
 NITER =       [illegible]
 [illegible]      0.0

 0.0            0.0            0.20000E-0[illegible]   0.20000E-0[illegible]
 SENSIBILITE/SENSITIVITY : E2
 0.0            0.0           -0.[illegible]E-[illegible]    0.[illegible]E-[illegible]
 SENSIBILITE/SENSITIVITY : R1
 0.0            0.0           -0.[illegible]E-[illegible]    0.[illegible]E-[illegible]
 SENSIBILITE/SENSITIVITY : R3
 0.0            0.0            0.20000E-0[illegible]  -0.[illegible]E-[illegible]
 [illegible]   [illegible]=   0.[illegible]E [illegible]
 0.333750D-[illegible]   0.50000E [illegible]   0.[illegible]E 0[illegible]   0.[illegible]E 0[illegible]
 0.75000D-10
 SENSIBILITE/SENSITIVITY : E2
 0.333750D-[illegible]   0.0            0.[illegible]E-[illegible]   0.[illegible]E-[illegible]
 SENSIBILITE/SENSITIVITY : R1
```

- beside MODELE … C2: *Description du schéma équivalent du transistor*
- beside E1 … V5: *circuit*
- beside TYPE: *transistor BSW48*
- beside TABLE: *source E1 décrite par une table*
- bracket beside REPRISE … R3 results: *calcul des conditions initiales*
- arrows beside the sensitivity results at t = 0: $\left.\frac{\partial}{\partial E_2}\right|_{t=0}$, $\left.\frac{\partial}{\partial R1}\right|_{t=0}$, $\left.\frac{\partial}{\partial R3}\right|_{t=0}$
- column labels above the results: *t*, *V2*, *V5*, *I1*
- arrow from 0.75000D-10: *pas d'intégration*
- arrow beside the last result line: $\frac{\partial}{\partial E_2}$

III.25

0.???7?D-?? 0.0 -0.2?0??E-?2 -0.2????E-?2 → ∂/∂R1
SENSIBILITE/SENSITIVITY : R3
0.333?3D-03 0.0 0.27???E-?? -0.2?????E-10 → ∂/∂R3

0.333?3D-03 0.50000E 01 0.31745E 0? 0.27050E 00
0.50000D-00
sensibilite/SENSITIVITY : E2
0.333?3D-03 0.0 0.11?43E 00 0.?30?5E-02
SENSIBILITE/SENSITIVITY : R1
0.333?3D-03 0.0 -0.215?2E-01 -0.19550E-01
SENSIBILITE/SENSITIVITY : R3
0.333?3D-03 0.0 0.23065E-0? -0.20714E-10

0.334?0D-03 0.0 0.30772E 00 0.14404E-02
0.750000D-10
SENSIBILITE/SENSITIVITY : E2
0.334?0D-03 0.0 0.72101E 00 0.72867E-01
SENSIBILITE/SENSITIVITY : R1
0.334?0D-03 0.0 -0.?2100E 00 -0.?2420E-01
SENSIBILITE/SENSITIVITY : R3
0.334?0D-03 0.0 0.72501E-10 0.43073E-10

0.150?0D-07 0.50000E 01 0.34157E 00 0.21524E 00
0.150000D-00
SENSIBILITE/SENSITIVITY : E2
0.150?0D-07 0.0 0.35217E 00 -0.50004E-01
sensibilite/SENSITIVITY : R1
0.150?0D-07 0.0 -0.10130E 00 0.25366E-02
SENSIBILITE/SENSITIVITY : R3
0.150?0D-07 0.0 0.?1043E-0? -0.15531E-10

0.161?1D-07 0.50000E 01 0.12236E 01 0.12607E 00
0.300000D-00
SENSIBILITE/SENSITIVITY : E2
0.161?1D-07 0.0 0.63772E 00 0.24024E-01
SENSIBILITE/SENSITIVITY : R1
0.161?1D-07 0.0 -0.00035E-01 -0.13415E-02
SENSIBILITE/SENSITIVITY : R3
0.161?1D-07 0.0 0.04504E-00 -0.60340E-10

0.192150D-07 0.0 0.15010E 01 0.36532E-02
0.375000D-10
sensibilite/SENSITIVITY : E2
0.192150D-07 0.0 0.51167E 05 0.13003E 04
SENSIBILITE/SENSITIVITY : R1
0.192150D-07 0.0 -0.50141E 02 -0.51079E 03
SENSIBILITE/SENSITIVITY : R3
0.192150D-07 0.0 0.44360E-05 0.20151E-05

0.222530D-07 0.13163E 01 0.13856E 01 -0.21572E-02
0.750000D-10
SENSIBILITE/SENSITIVITY : E2
0.222530D-07 0.0 0.13307E 03 -0.14524E-01
SENSIBILITE/SENSITIVITY : R1
0.222530D-07 0.0 -0.22717E 02 0.20183E-02
SENSIBILITE/SENSITIVITY : R3
0.222530D-07 0.0 0.14791E-05 -0.13112E-10

0.255530D-07 0.50000E 01 0.15520E 01 0.36413E-01
0.300000D-00
SENSIBILITE/SENSITIVITY : E2
0.255530D-07 0.0 0.14307E 03 -0.10365E 02
sensibilite/SENSITIVITY : R1
0.255530D-07 0.0 -0.10741E 02 0.14035E 01
SENSIBILITE/SENSITIVITY : R3
0.255530D-07 0.0 0.?????E 0? 0.?????E ??

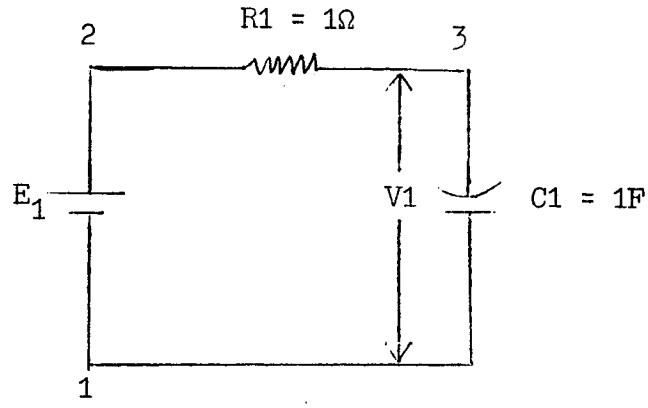
III.26

```
 0.238450-07  0.0          0.130515 01  0.205213-03
 0.150000-01
SENSIBILITE/SENSITIVITY : S2
 0.238450-07  0.0          0.457543 02 -0.190715 01
SENSIBILITE/SENSITIVITY : R1
 0.238450-07  0.0         -0.220405 02  0.291525 01
sensibilite/SENSITIVITY : R3
 0.238450-07  0.0          0.343285-07 -0.175723-03

   271 PAS/STEPS     25 PAS REFUSES/REJECTED STEPS
   823 ITERATIONS    331 JACOBIENS

/
Fin
R; T=27.33/32.00 11:21:21

Impr
R; T=0.37/1.13 11:21:43
```

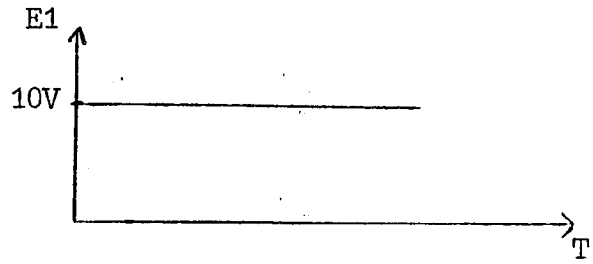
VII. Exemple de calcul en optimisation.

Nous donnons un exemple de calcul en optimisation. Il n'a pas d'autre but que de montrer l'enchaînement des commandes et la présentation de sortie des résultats.

Considérons le circuit simple suivant :



E1 prenant la forme suivante :



Avec les valeurs des composants donnés, V1 atteint 9 V au bout de 2,4 s. On désire que ce temps soit augmenté jusqu'à 5 s. Pour cela, on minimise la fonction B = (TA - 5) * (TA - 5) où TA est la valeur de T (temps) pour laquelle la fonction C = 10^3 * (V1 - 9) définissant un seuil, s'annule.

On trouve dans les pages suivantes les résultats obtenus sur un console relié au système CP/CMS.

```
go
EXECUTION BEGINS...

  DOIS-JE ETRE BAVARD ?      OUI/NON
oui

 ***NOM DU CIRCUIT ?***...OU COMMANDE DEBU(T)...
rc.circuit
ADRESSE D'INTERRUPTION : 000A9F0C

/
desc
+e1(1,2)'pulse'$    r1(2,3) 1$  c1(3,1) 1$
v1(c1) $
ta=0$
b=(ta-5)*(ta-5) $
c=1e3*(v1-9)  $$
table:'pulse' 0/0,1n/10,10/10    $$$

/
cont

***DONNEES?***
t=0 $  sortie:v1,c $$

  T            V1           C

 **** NEWTON ***

CODE L*U:     3

CODE L*U+CODE RESOL:    11
 NITER =     1
DC  MAXI     0.0

 0.0          0.0          -0.90000E 04

/
tran

***DONNEES?***
seuil:c(ta=t) $ tmax=10 $ sortie:v1 $$

  T            V1

 0.0          0.0
 GEAR    ERMAX=    0.100E 00

CODE L*U:     3

CODE L*U+CODE RESOL:    11

 0.11000D 01  0.66234E 01
 0.10000D 00

 0.22000D 01  0.88805E 01
 0.10000D 00

 0.32130D 01  0.95948E 01
 0.10000D 00

 0.43130D 01  0.98657E 01
 0.10000D 00

 0.53130D 01  0.99508E 01
```

description du circuit

fonction objectif

fonction seuil

Calcul des conditions initiales

c est effectivement precisée comme definissant un seuil.

longueur en mots des codes interpretables de resolution de $J\,\Delta X = -\Delta F$

calcul du regime transitoire

```
 0.10000D 00

 0.64130D 01  0.99837E 01
 0.10000D 00

 0.75130D 01  0.99946E 01
 0.10000D 00

 0.85130D 01  0.99980E 01
 0.10000D 00

 0.96130D 01  0.99993E 01
 0.10000D 00

   102 PAS/STEPS      0PAS REFUSES/REJECTED STEPS
   101 ITERATIONS    103 JACOBIENS

/
opti ← commande OPTIMISATION

***DONNEES?***
sens:c1(0,100)$ cont$ tran$ min:b$ sortie:b,c1,ta,c,v1$$
+
  T              B              C1             TA             C
                V1
```

← définition de b comme fonction à minimiser

← définition de c1 comme paramètre et bornes de variations de c1

```
***METHODE DE FLETCHER-POWELL***

CODE L*U:     3

CODE L*U+CODE RESOL:    11
 NITER =     1
  V1=    0.230D 01
0
```

← 1ère évaluation de b

```
 0.23130D 01  0.72200E 01  0.10000E 01  0.23130E 01  0.51668E 00
              0.90005E 01
 NITER =     1
  V1=    0.230D 01
1
```

← valeur de $\nabla T(c)$

← Recherche linéaire

```
 0.57652D 01  0.58553E 00  0.25019E 01  0.57652E 01  0.17745E 00.
              0.90002E 01
 NITER =     1
  V1=    0.230D 01
2
```

← itération de Fletcher et Powell.

```
 0.49852D 01  0.22047E-03  0.21629E 01  0.49852E 01  0.13262E 00
              0.90001E 01
 NITER =     1
  V1=    0.230D 01
3
```

← nouveau calcul de la matrice Hessienne

```
 0.49852D 01  0.22047E-03  0.21629E 01  0.49852E 01  0.13262E 00
              0.90001E 01
 NITER =     1
  V1=    0.230D 01
1

 0.50000D 01  0.31974E-11  0.21694E 01  0.50000E 01  0.13588E 00
              0.90001E 01
 NITER =     1
  V1=    0.230D 01
```

```
 0.50149D 01  0.22084E-03  0.21758E 01  0.50149E 01  0.12836E 00
              0.90001E 01
 NITER =    1
  V1=   0.230D 01
2

 0.50000D 01  0.31974E-11  0.21694E 01  0.50000E 01  0.13505E 00
              0.90001E 01
ARRET CAR PRX=      0.465183D-02 OU PRF=      0.0          INFERIEUR A
   3 PAS D OPTIMISATION                                    0.01
   7 EVALUATIONS DE      LA FONCTION
 IER=   0
/
fin
R; T=8.74/14.56 11:22:30
```

Valeurs finales

← l'algorithme a convergé.

On a pu voir que parmis les programmes effectivement opérationnels, IMAG3 occupe une place intéressante qui le situe au tout premier rang des progammes mondiaux,tant par sa généralité,sa rapidité,et sa précision d'analyse que par la puissance de son langage. (*)

Sous sa forme actuelle,il semble particulièrement bien adapté pour une analyse fine de circuits moyens(jusqu'à 50 transistors environ).

Les solutions adoptées réalisent un compromis intéressant entre une analyse efficace et un temps de compilation raisonnable.

D'autre part,dans certains cas présentant des difficultées numeriques,il est-à notre avis-supérieur à la méthode du tableau par la possibilité pratique qu'il y a d'effectuer de nouveaux choix de pivots peu couteux au cours de la phase d'analyse..

CONCLUSION

Le système IMAG 3, comme les systèmes qui ont été cités tout au long de ce document, correspondent à des outils adaptés à des circuits comportant au plus 100 à 200 dispositifs et marquent une étape de la recherche dans ce domaine.

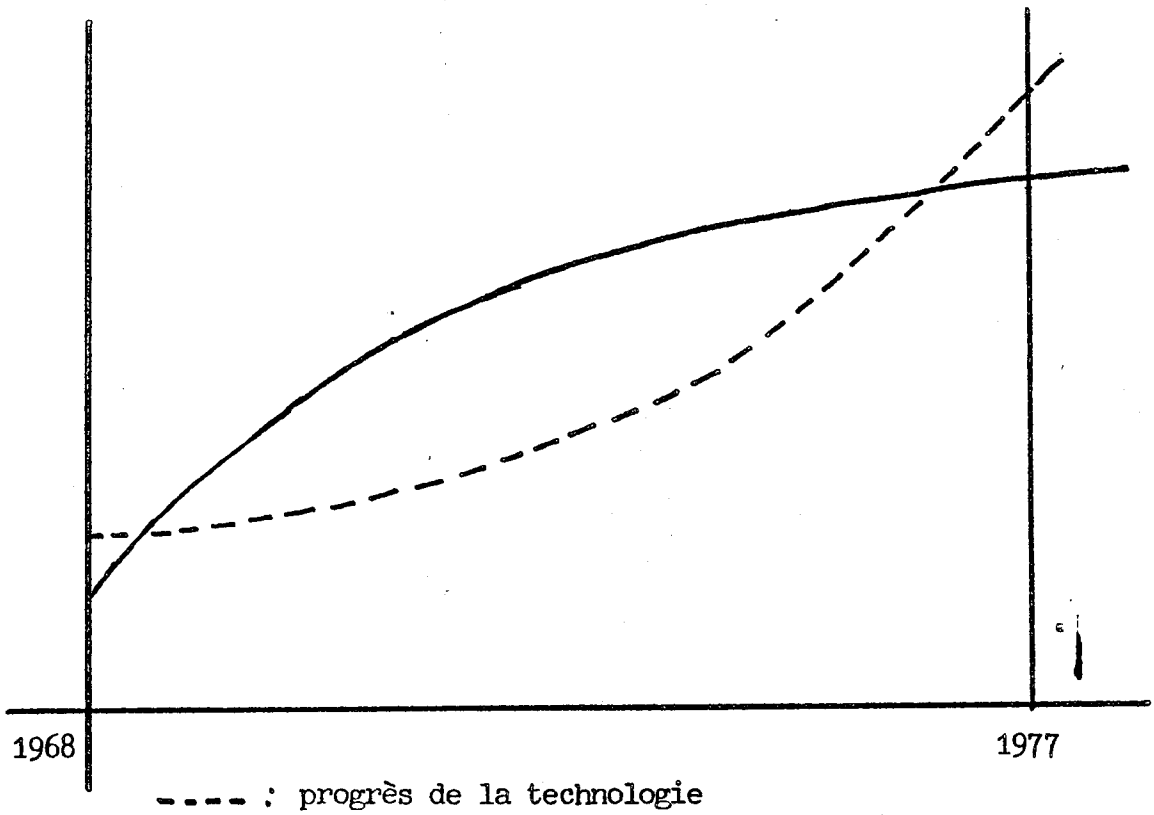
Ces outils conviennent à une simulation précise, mais il n'est pas concevable de simuler le comportement de circuits intégrés complets, le nombre de composants étant extrêmement grand et ne cessant d'augmenter.

C'est pourquoi le concepteur découpe le circuit en blocs fonctionnels de dimension aussi faible que possible.

Cependant, le nombre d'éléments composant ces blocs est en constante augmentation (depuis 1974, il double chaque année).

A l'heure actuelle, les techniques d'intégration à grande échelle permettent d'envisager des circuits comportant 10.000 composants actifs et il semble que des circuits en comportant 50.000 soient en préparation.

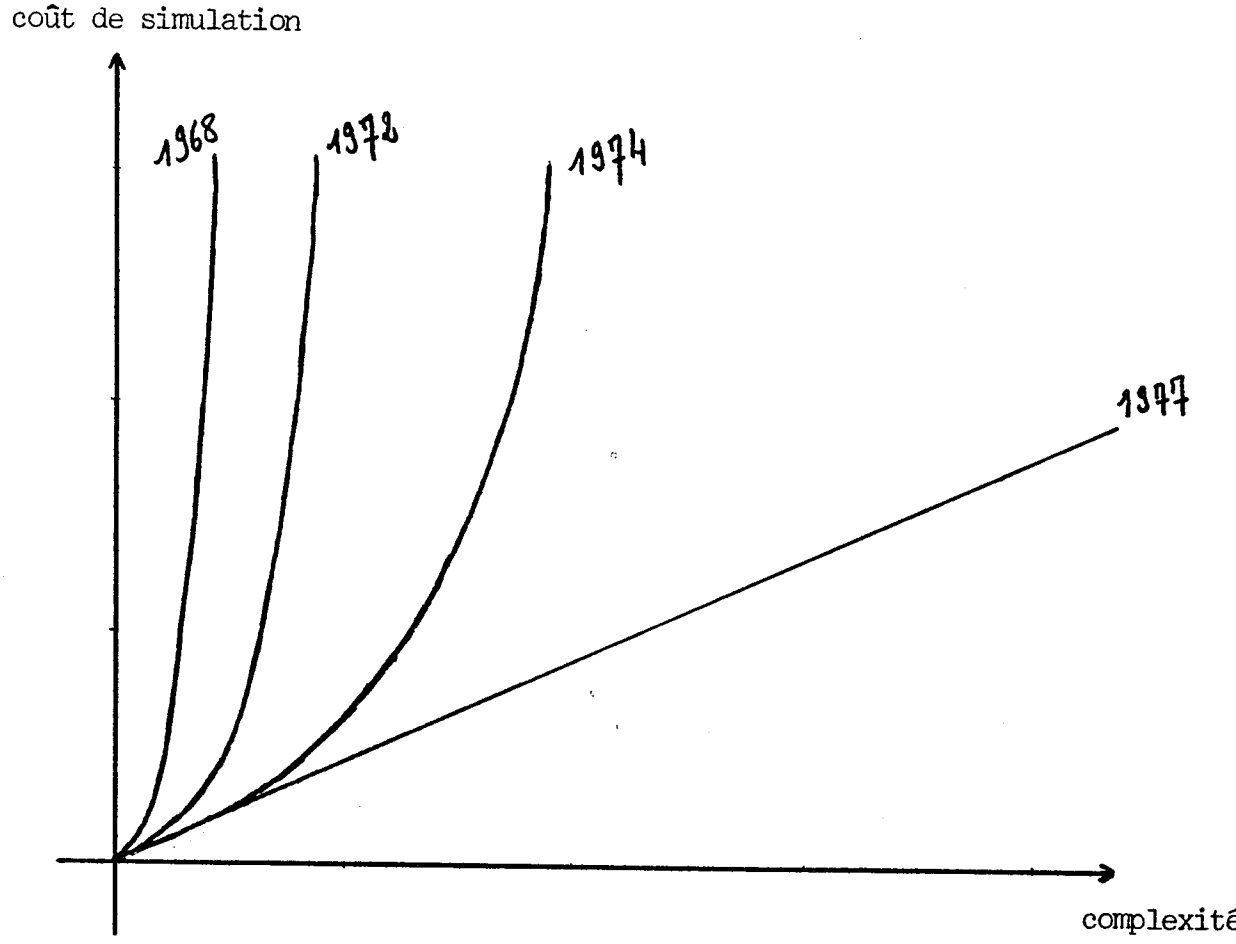
Le fossé se creuse entre les progrès de la technologie et les possibilités des outils de C.A.O. Ceci est illustré par les courbes ci-dessous.



- - - - : progrès de la technologie

——— : progrès de la C.A.O.

Il est intéressant de rapprocher ces courbes de celles traduisant les progrès accomplis en simulation pendant cette même période.



L'évolution que l'on peut constater sur ces courbes est due aux progrès accomplis dans les algorithmes mathématiques de simulation. Dans l'état actuel des connaissances en analyse numérique, il est illusoire d'attendre des progrès spectaculaires dans cette voie à brève échéance. Et de toute manière, ils seraient insuffisants comme le confirment les récentes études de Pederson à Berkeley, et de De Man à Louvain. Ces auteurs s'accordent pour dire que, dans les programmes de simulation les plus rapides, 70 à 80 % du temps de cette simulation est utilisé à l'évaluation des expressions décrivant les composants.

Ainsi, même si le coût des algorithmes numériques de résolution était réduit à zéro, cette approche serait insuffisante.

Ceci veut-il dire que des programmes de simulation tels qu'IMAG 3 sont devenus caduques ? Nous ne le pensons pas, simplement, ils semblent inadaptés aux circuits intégrés LSI actuels.

Un programme de simulation est toujours conçu en fonction d'un type de modèle que l'on donne de l'objet à étudier. Aussi, pour des analyses fines de circuits de dimension "moyenne", ils demeurent des outils irremplaçables.

Nous pensons qu'il faut, à l'heure actuelle, distinguer 2 types de problèmes, bien que ceux-ci ne soient pas généralement disjoints :

(a) le problème de la dimension de plus en plus importante des circuits.

(b) le problème de la difficulté de l'analyse essentiellement au niveau de la fiabilité des algorithmes numériques.

Lié étroitement aux deux problèmes précédents, existe un troisième problème qui est celui des modèles permettant de représenter un circuit.

En fait, les problèmes (a) et (b) ne peuvent s'envisager séparément de celui des modèles, car en simulation, on est toujours conduit à construire une machine à partir du modèle de l'objet et dont le fonctionnement traduira le comportement de cet objet.

Aussi, les problèmes (a) et (b) doivent-ils s'aborder en considérant le couple {Modèle de l'objet, algorithme de simulation} : $\{M_o, A_s\}$, et ceci par une adaptation réciproque algorithmes-modèles, modèles-algorithmes, dans la poursuite d'un but particulier que l'on se définit a priori.

Par exemple, si le but poursuivi est d'analyser des circuits contenant un grand nombre de dispositifs, les modèles et les algorithmes devront être différents de ceux utilisés pour simuler finement et avec une grande fiabilité des circuits plus petits. Ces contraintes étant introduites par les propriétés des algorithmes disponibles et par les modèles que l'on est à même de construire pour représenter les circuits.

Ceci peut sembler évident, et cependant, il faut bien constater que jusqu'ici, toutes les conséquences de cette démarche n'ont pas été tirées.

Ainsi, comme on a pu déjà le constater, le modèle du circuit est un schéma algébro-différentiel de conditions initiales, et l'on a tenté d'améliorer les algorithmes de résolution.

C'est-à-dire que dans le couple $\{M_o, A_s\}$, seul A_s a été modifié.

Nous pensons que cette façon de faire a atteint actuellement ses limites et qu'il est nécessaire d'agir maintenant sur M_o.

Des travaux dans ce sens sont actuellement en cours pour aborder le problème (a).

- soit en introduisant une simulation mixte (et donc des modèles mixtes) logique et analogique [De Man 77, Peder 77].
- soit en utilisant une approche "boîte noire" [Le Faou 77, Azen 77]. Les travaux sur l'approche boîte noire sont menés en collaboration par l'équipe de l'ENSIMAG et une équipe de l'Université de Lyon I, animée par D. VANDORPE.

Les premiers résultats aussi bien dans l'une que dans l'autre approche, semblent encourageants.

Le problème (b) n'a pas reçu jusqu'ici de solutions satisfaisantes. Une première approche à ce problème est celle décrite dans le premier paragraphe du chapitre II, pour le calcul des états stables. Rappelons qu'elle consiste finalement à anticiper sur le fonctionnement du circuit par des données qualitatives que l'on pourra utiliser ensuite dans l'analyse quantitative.

En fait, cette approche était déjà utilisée dans la simulation traditionnelle sur maquettes. En effet, si l'on veut étudier un circuit dans lequel on soupçonne que les courants y sont de l'ordre de 50 ampères, on n'utilisera pas, pour les mesurer, un milli-ampèremètre.

Le modèle M_o du circuit contiendra ainsi un certain nombre de renseignements qualitatifs sur le fonctionnement de ce circuit. Bien entendu, la nature même de ces renseignements dépendra du type de modèle quantitatif adopté, mais la méthodologie reste valable quel que soit ce type de modèle.

Les deux difficultés majeures dans cette approche sont :

- quels types de renseignements peut-on fournir ?
- comment utiliser ceux-ci dans l'analyse quantitative ?

Le premier problème est plus spécifique du concepteur de modèles, tandis que le second est davantage du domaine du concepteur de système C.A.O. Mais bien entendu, ils sont étroitement liés. Nous orientons actuellement nos travaux dans ce sens.

Malgré ce qui précède - qui peut sembler négatif - le système IMAG 3 continuera à être utilisé dans un grand nombre de cas pour lesquels il donne entière satisfaction.

D'autre part, il fournit une base à partir de laquelle peuvent se développer les idées précédentes et c'est bien ainsi, car comme le dit KEYSERLING :

"Seul l'insuffisant est productif".

BIBLIOGRAPHIE

BIBLIOGRAPHIE GENERALE

[AZEN 77] AZENCOT, TOSAN, VANDORPE, LE FAOU, REYNAUD :
"Modélisation et simulation fonctionnelle de circuits électroniques complexes"
Congrès AFCET Modélisation et maitrise des systèmes.
Versailles - Novembre 1977.

[BASH 57] T. R. BASHKOW : "The A-Matrix, New Network Description"
Trans. Inst. Radio Engrs.
CT - 4, p. 117 - 119, Septembre 1957.

[BEN 74] A. BENSASSON : "Etude comparative de différents algorithmes dans la minimisation des fonctions pénalités et des fonctions barrières".
Communication personnelle - Avril 74.

[BLATT 76] D. J. BLATTNER : "Choosing the right programs for computer-aided design".
Electronics, p. 102 - 105, Avril 1976.

[BLISS 46] G. A. BLISS : "Lectures on the Calculus of Variations".
Phoenix Science Series, 1946.

[BORR 76] D. BORRIONE : "LASCAR, un langage pour la simulation et l'évaluatio des architectures d'ordinateurs" Thèse 3ème Cycle,
Grenoble, Avril 1976.

[BOW 76] BOWERS et AL : "A survey of computer-aided design and analysis programs".
Report AFAPL - TR - 76 - 33
University of South Florida Tampa FL 3 36 20 - April 76

[BRAN 71] F. H. BRANIN, G. R. HOGSETT, R. L. LUNDE, L. E. KUGEL :
"ECAP2 - A new electronic circuit analysis program".
IEEE Journ.of Solid-State circuits - Vol. SC - 6, n° 4,
August 1971.

[BRAN 72] F. H. BRANIN : "Widely convergent method : a new attempt for solving non linear simultaneous equations".
IBM RA [1972].

[BROY 65] C. G. BROYDEN : "A class of methods for solving nonlinear simultaneous equations".
Mathematics of Computation, Vol. 19, p. 577 - 573, 1965.

[BRYANT 59] P. R. BRYANT : "The Order of complexity of electrical networks".
Proc. IEE (London) Monograph 335E, Vol. 106 C, p. 174 - 188, June 1959.

[BRYANT 62] P. R. BRYANT : "The explicit form of Bashkow's A-Matrix"
IRE TRans. on C.T. p. 303 - 306, September 1962.

[CA 72] D. A. CALAHAN : "Computer-aided network design"
2nd edition, Mac Graw-Hill, 1972.

[CAR 73] G. CARRY : "Optimisation avec IMAG 2 : Notice d'utilisation"
IMAG, 1973.

- G. CARRY, C. LE FAOU : "Optimisation de circuits électroniques".
Rapport Contrat DRME n° 70/563 - Juin 1973

[CESCH 63] CESCHINO et KUNTZMANN : "Problèmes différentiels de conditions initiales"
Dunod, 1963.

[DANTZ 69] G. B. DANTZIG et AL : "Sparse Matrix Techniques in two mathematical programming codes" in Sparse Matrix Proceedings.
IBM RA1 1969

[DAV 53] D. DAVIDENKO : "Dokl. Akad. Nauk. USSR, 88, p. 601, 1953.

[DE MAN 77] H. DE MAN and G. ARNOUT : "The use of threshold functions and boolean controlled network elements for macromodeling of LSI circuits".
ESSCIRC 77, September 20 - 22, 1977, ULM.

[DIR 69] S. W. DIRECTOR and R. A. ROHRER : "The generalized adjoint Network and network sensitivities."
IEEE Trans. on C.T., CT - 16, p. 318 - 323 (1969)

[ESSL 74] ESSL, MITTERER, REHN, DOMITROWITCH :
"Automated design optimization of integrated switching circuits"
IEEE Journal of Solid-State circuits, Vol. SC - 9,
n° 1, February 1974.

[FIA 68] A. FIACCO and G. P. MC CORMICK : " Non linear programming : sequential unconstrained minimisation techniques".
John Wiley (1968).

[FIA 73] A. FIACCO : "Barrier methods for non linear programming".
The Georges Washington University - 1973.

[FLETCH 63] R. FLETCHER and M. J. D. POWELL : "A rapidly convergent discret method for minimization"
Comp. J. 6 (1963).

[FLETCH 64] R. FLETCHER and C. M. REEVES : "Function minimization by conjugate gradient"
Comp. J. 7 (1964).

[FOW 67] FOWLER and WARTEN : "Numerical integration technique for ordinary differential equations with widely separated eigenvalues"
IBM Journal, September 1967.

[GAST 68] N. GASTINEL : "Equations différentielles, fonctions spéciales, équations aux dérivées partielles". Cours de maitrise d'informatiq
Grenoble 1968.

[GEAR 69] C. W. GEAR : "The automatic integration of stiff ordinary differential equations".
IFIP 68 - North Holland P.C. AMSTERDAM (1969).

[GEAR 71] C. W. GEAR : "Numerical initial value problems in ordinary differential equations".
Prentice - Hall inc. 1971.

[GEAR 71] C. W. GEAR : "Simultaneous numerical solution of differential-algebraic equations".
IEEE Trans. on C.T., Vol. CT 18, n° 1, p. 89 - 95,
June 1971.

[GOLD 63] J. GOLDBERG et J. RICHARD :
J. Struct. Div. ASCE, 89, p. 333 - 351. 1963

[GRIFF 69] M. GRIFFITHS : "Analyse deterministe et compilateurs".
Thèse, Grenoble 1969.

[HACHT 67] G. D. HACHTEL et R. A. ROHRER : "Techniques for the Optimal design and Synthesis of switching circuits".
Proc. IEEE, 55, p. 1864 - 1877 (1967).

[HACHT 71] HACHTEL, BRAYTON, GUSTAVSON : "The Sparse Tableau approach to network analysis and design".
IEEE Trans. C.T., Vol. CT 18, Janvier 1971.

[HEARN 73] A. C. HEARN : "Reduce 2, User's manual"
University of Utah,

[HENN 74] HENNION et MAITRE : "Amélioration des méthodes de résolution d'équations non linéaires rencontrées dans la simulation des circuits électroniques".
Projet de fin d'études de l'Ecole Polytechnique, Janvier 1974.

[HEN 62] P. HENRICI : "Discrete variable méthods in ordinary differential equations".
John Willey, N. Y., 1962.

[HEYD 72] M. HEYDEMANN : "Résolution numérique des équations bidimensionnelles de transport dans les semi-conducteurs".
Thèse n° 173, Université Paris-Sud, Octobre 1972.

[HEYD 73] M. HEYDEMANN : "ASTEC, premier pas vers la simulation à grande échelle des circuits électroniques".
L'onde électrique, Vol. 53, fasc. 10, p. 369 - 374, Novembre 1973.

[JAC 70] JACOLIN, LE FAOU, PIMORT, VERAN :
"IMAG 2 : Description du programme".
IMAG, Juillet 1970.

[JAC 70.a] JACOLIN, LE FAOU, PIMORT, VERAN :
"IMAG 2 : Notice de programmation"
IMAG, Juillet 1970.

[KLYU 65 V. V. KLYUYEV et N. I. KOKOVKIN-SHCHERBAK :
"On the minimization of the number of arithmetic operations for the solution of linear algebraic systems of equations".
(Traduit par G. Tee) Computer Science Department,
Technical Report CS24, Standford University.

[KNUTH 68] D. KNUTH : "The art of computer programming"
Vol. 1.
Addison Wesley Reading, Massachussets.

[KUNTZ 72] J. KUNTZMANN : "Théorie des réseaux"
Dunod, 1972.

[LAH 48] E. LAHAYE : Acad. Royal Belgian Bull. cl. Sc., 5
p. 805 - 822, 1948.

[LE FAOU 72] C. LE FAOU : "Problèmes d'analyse numérique dans le programme IMAG 2"
Séminaire d'analyse numérique, Grenoble, Février 1972.

[LE FAOU 76] C. LE FAOU et S. MATVITCHOUK : "Une méthode de résolution de systèmes algébriques linéaires mal conditionnés, son implantation dans IMAG 3".
Séminaire d'analyse numérique, Grenoble, Novembre 1976.

[LE FAOU 77] C. LE FAOU et J. C. REYNAUD : "Functional simulation of complex electronic circuits".
ESSCIRC' 77, Ulm, September 20 - 22, 1977.

[LISKOV 74] B. LISKOV : "A note on CLU"
Project MAC Computation Structures Group,
Memo 112, MIT, November 1974.

[LOOT 70] F. A. LOOTSMA : "Boundary properties of penalty functions for constrained minimization"
Thèse, Université de Eindhoven, May 1970.

[MERM 73] J. MERMET : "Etude méthodologique de la conception assistée par ordinateur des systèmes logiques : CASSANDRE"
Thèse, Grenoble, 1973.

[ORTE 70] J. ORTEGA et W. RHEINBOLDT : "Iterative solution of non linear equations in several variables".
Academic Press, N. Y., 1970

[PAL 59] R. PALAIS : "Natural operations on differential forms".
Trans. of Am. Math. Soc.,
Vol. 92, n° 1, p. 125-141, July 1959

[PEDER 77] A. R. NEWTON et D. O. PEDERSON : "A simulation program with large-scale integrated circuit emphasis".
RA, university of California, Berkeley.

[RAB 75] RABBAT, RUEHLI, MAHONEY, COLEMAN : "A review of macromodeling techniques".
Circuit and systems, p. 3-9, October 1975.

[REYN 74] J. C. REYNAUD : "Méthodes numériques en simulation de circuits électroniques".
Séminaire d'analyse numérique, Grenoble 1974.

[SESA 75] SESA : "IMAG 3 : Notice d'utilisation"
Mars 1975

[SESHU 61] S. SESHU et M. B. REED : "Linear graphs and electrical networks".
Addison-Wesley, 1961

[SHIC 70] H. SHICMAN : "Integration system of a non linear network analysis program".
IEEE Trans. on C.T., Vol. CT 17, n° 3, p. 378-386,
August 1970

[SUSS 75] G. J. SUSSMAN et R. M. STALLMAN :"Heuristic techniques in computer aided circuit analysis".
M.I.T., A.I.L., Memo 328, March 1975.

[TEL 52] B.D.H. TELLEGEN : "A general network theorem with applications'
Philips Res. Rept. p. 259-269 (1952).

[TEW 67] R. P. TEWARSON : "Sparse Matrices"
Academic Press, 1973.

[TIN 67] W. TINNEY et J. WALKER : "Direct solutions of sparse network equations by optimally ordered triangular factorization".
Proc. IEEE 55, p. 1801-1809, 1967.

[VAN 71] D. VANDORPE : "Etude mathématique de la fabrication et du fonctionnement des dispositifs semi-conducteurs".
Thèse, Lyon, 1971.

[WEEKS 73] WEEKS, JIMENEZ, MAHONEY, MEHTA, QASSEMZADEM, SCOTT :
"Algorithms for ASTAP-a network analysis program".
Proc. IEEE, Trans. Circuit Theory, Vol. CT 20, Novembre 1973.

[WILK 65] J. H. WILKINSON : "The algebraic eigenvalue problem".
Oxford University Press, London and New York.

[WINT 45] A. WINTER : "The non-local existence problem of ordinary differential equations".
Amer. Jour. Math., Vol. 67, p. 277-284, 1945.

[WULF 76] WULF, LONDON, SHAW : "Abstraction and verification in ALPHARD : Introduction to language and methodology".
Carnegie-Mellon University Technical report, March 1976.

REFERENCES SUR IMAG 2

(non citées dans la bibliographie générale)

JACOLIN : "Les programmes IMAG1 et IMAG2"
Onde électrique n° 502, Janvier 1969.

JACOLIN, LE FAOU, PIMORT, VERAN : "Simulation de circuits électroniques : le programme IMAG2".
Colloque international sur la Micro-électronique avancée.
Paris - Avril 1970.

LE FAOU : "Conception assistée par ordinateur : IMAG2".
The Jerusalem Conference on Information Technology.
Jerusalem - Août 1971.

LE FAOU : "Simulation de circuits électroniques"
IEEE - Eurocon 71 - Lausanne - Octobre 1971.

ARNOULD et LE FAOU : "Simulation de circuits analogiques : le programme IMAG2". Electronique et micro-électronique industrielle, p. 45-49, Mars 1972.

LE FAOU et VERAN : "IMAG2, a general Analysis and Optimization program for non-linear circuits".
Canadian Computer Conference - Session 72 - Montréal - Juin 1972.

ARNOULD, LE FAOU, SICOT : "IMAG2 - Simulation de circuits électroniques".
A.G.A.R.D. Computer Aided Design for Electronic Circuits - Copenhague - Mai 1973.

LE FAOU, SICOT : "IMAG3, un programme général de simulation de circuits électroniques".
Electronique et Micro-électronique industrielle, p. 39-43 - Octobre 1974.