Simulation et analyse interactives des systèmes dynamiques en sciences sociales

Francisco Eduardo Rivera Porto

THESE

présentée à

Université Scientifique et Médicale de Grenoble

Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR DE TROISIEME CYCLE

Génie Informatique

par

Francisco Eduardo RIVERA PORTO

SIMULATION ET ANALYSE INTERACTIVES DES SYSTEMES DYNAMIQUES EN SCIENCES SOCIALES

Thèse soutenue le 26 mars 1977 devant la Commission d'Examen

Président :	Monsieur	L. BOLLIET
Examinateurs :	Messieurs	A. DUSSAUCHOY
		H. CHAMUSSY
		Ph. JORRAND
		J. MERMET

Je tiens à remercier :

Monsieur L. BOLLIET, Professeur à l'I.N.P.G., qui a bien voulu me faire l'honneur de présider le jury de cette thèse ;

Monsieur A. DUSSAUCHOY, Maître de Conférences à l'Université de LYON 1, animateur et directeur du G R E C O Rhône-Alpes d'Analyse de Systèmes, pour son intérêt porté à ce travail et qui a accepté de participer au jury ;

Monsieur P. JORRAND, Maître de Recherches au C.N.R.S., qui a aimablement accepté d'être membre du jury ;

Monsieur J. MERMET, Chargé de Recherches au C.N.R.S., responsable de l'équipe où ce travail a été mené, pour son accueil, et ses encouragements ;

Monsieur H. CHAMUSSY ; Maître Assistant à l'U.S.M.G. et directeur adjoint de l'Institut de Géographie Alpine, pour son enthousiaste collaboration ainsi que ses intéressantes discussions sur la géographie quantitative.

Une reconnaissance particulière à mes camarades d'équipe P. UVIETTA, Y. BRESSY et F. RECHENMANN pour leur très fructueuse collaboration dans un travail de groupe.

Mes remerciements les plus sincères à Mmes J. BRICHET et T. RIVERA qui ont assuré la dactylographie et la mise en page de ce document.

Je tiens également à remercier l'aimable coopération de V. ROSSETTI pour les schémas et F. ZANELLI pour son sympathique dessin.

Ma profonde reconnaissance au Ministère des Affaires Etrangères (FRANCE) au Conseil National de la Science et la Technologie CONACYT (MEXIQUE) qui ont rendu possible mon séjour en France.

Enfin merci à tous mes collègues et amis qui à travers leurs nombreuses suggestions, critiques et lectures de ce texte m'ont apporté une aide incomparable. Je félicite aussi les membres du service de reproduction chargés du tirage de cet ouvrage.

"SIMULATION ET ANALYSE INTERACTIVES DE SYSTEMES DYNAMIQUES EN SCIENCES SOCIALES"

TABLE DES MATIERES

INTRODUCTION

La Simulation est une technique encore mal connue en Sciences Sociales. Jusqu'à maintenant sa maîtrise et son adaptation demandaient un effort et un temps d'apprentissage non négligeables ; ce qui a longtemps bloqué son utilisation et a parfois provoqué de mauvaises interprétations.

L'Informatique peut fournir de nouveaux outils et concepts pour la Simulation en Sciences Sociales ; mais surtout elle doit les rendre facilement accessibles. Deux moyens généraux se révèlent importants :

- un langage dans lequel on peut exprimer certains problèmes par un modèle et dans lequel on peut trouver les concepts reflétant les spécificités des systèmes étudiés en Sciences Sociales.

- un ensemble des moyens permettant de manipuler, d'analyser et de simuler de façon claire et simple les modèles ainsi réalisés.

Le premier moyen a déjà été abordé au sein de l'équipe * .
Le second fait l'objet de la présente thèse.

Les particularités des objets étudiés, les systèmes sociaux et leur dynamique, demandent une présentation du cadre général de travail : la Prospective et l'Analyse de Systèmes en Sciences Sociales.

---

* F. Rechenmann : "Analyse et modélisation descendantes des systèmes socio-économiques" Thèse Doct.-Ing., INPG - 1976

On situe par rapport à celles-ci la simulation, ses avantages et ses limites. Le premier chapitre (première partie) montre l'intérêt de la simulation et justifie son application en Sciences Sociales, les confusions rencontrées par le passé et les mauvaises interprétations de cette démarche méritent que l'on explique le pourquoi de l'approche.

Le tour de la question resterait incomplet si l'on ne présentait pas un survol critique des moyens apportés par les systèmes de simulation continue à l'heure actuelle (Chapitre I, deuxième partie). Ceci amène à proposer un vrai *système* de simulation et d'analyse dans le cadre de l'application : SAISYE.

Le choix conversationnel et la définition des diverses options sont spécifiés. Une structure de système, vis-à-vis de l'utilisateur et permettant un guidage de celui-ci, est proposée. L'architecture interne d'un tel système est aussi présentée, mais elle doit, bien évidemment, rester transparente à l'utilisateur (Chapitre II).

Un modèle régional hiérarchisé de la dynamique migratoire sert d'exemple d'utilisation du système pour la construction, la mise au point, le test, la simulation et l'analyse de scénarios (Chapitre III).

Les points les plus importants de la réalisation du système "SAISYE" sont exposés. Divers problèmes informatiques rencontrés lors de la définition et de la mise en oeuvre de ce système ainsi que les solutions adoptées sont présentés dans le chapitre IV.

A titre de conclusion (Chapitre V), on discute les limitations qui demeurent dans la présente réalisation, et les prolongements possibles du système ; enfin on envisage la voie vers différents problèmes encore ouverts en simulation et analyse de systèmes.

CHAPITRE I

SYSTÈMES DYNAMIQUES EN SCIENCES SOCIALES ET SYSTÈMES INFORMATIQUES DE SIMULATION

TABLE DES MATIERES

PREMIÈRE PARTIE

LA DYNAMIQUE DES SYSTÈMES SOCIAUX

TABLE DES MATIÈRES

DEUXIÈME PARTIE
LES SYSTÈMES INFORMATIQUES DE SIMULATION CONTINUE

PREMIERE PARTIE

LA DYNAMIQUE DES SYSTEMES SOCIAUX

1 . L'ANALYSE DE SYSTEMES

On parle de systèmes, systèmes socio-économiques, systèmes d'exploitation, systèmes de simulation ..., dans des contextes si différents que l'on a tendance soit à oublier le sens du mot "système", soit à lui donner un sens très vague et, par abus ou par mode, ou par confusion de terminologie, on continue cependant à l'utiliser. Il convient donc avant tout, de préciser de quoi l'on parle, sans pour autant répéter tout ce qui a été dit sur le sujet depuis L.V. Bertalanffy [1].

Selon l'acception la plus généralement admise, on entend par Système : un ensemble d'éléments ayant une certaine autonomie par rapport à leur environnement, interagissant entre eux, et dont le fonctionnement est orienté vers la réalisation d'objectifs.

Donc on parle de *Système*. Mais cela n'indique pas pour autant comment les étudier. Il n'y a pas encore de Science des Systèmes et on est encore loin d'avoir à leur sujet une vraie *théorie générale* ; on doit plus modestement se situer au niveau de *l'Analyse de Systèmes* ou de *l'Approche systémique*. Ceci est un esprit de travail, une approche méthodologique (mais qui n'est pas une méthode !) où l'étude des systèmes (par nature unificatrice) confronte leurs contours, leurs éléments et les inter-relations de ceux-ci aux objectifs. La démarche *d'analyse* (au contraire de la méthode analytique) n'isole pas les éléments pour les étudier, mais après les avoir identifiés en fonction des objectifs retenus, met en relief leurs inter-relations et les intégre dans une vision globale.

Quel est le but de l'Analyse de Systèmes ? Yves Barel [2] propose deux orientations différentes : l'orientation décisionnelle et l'orientation cognitive.

- Pour l'Analyse de Systèmes Décisionnelle, le problème fondamental (et plus difficile ?) réside en un choix d'objectifs (du système !) qui permettront par la suite de déterminer les moyens et ressources mis en oeuvre pour les atteindre (évaluation du coût, détermination d'une politique, structure interne du système, mesures à faire, etc). On débouchera finalement sur l'élaboration d'un modèle qui devra prévoir le comportement du système (ce modèle devra donc être validé ; car le fait qu'"un modèle soit meilleur ou moins bon qu'un autre dépend non pas de sa complexité ou de son réalisme apparent mais uniquement de la qualité des prévisions qu'il nous fournit" [3]. Puis selon un critère (minimum, maximum, etc) on chcisira les meilleures alternatives.

- L'Analyse Cognitive de Systèmes, par contre, se situe dans la plupart des cas *au pôle hypothético-déductif* par opposition *au pôle empirico-inductif* (qui par exemple s'appuie sur une théorie). Son problème n'est pas de trouver un ou des objectifs pour le système mais de cerner ce dernier en fonction des objectifs de l'étude. Le but de l'Analyse cognitive est donc saisir le système considéré non pas seulement comme ensemble d'objets mais surtout comme ensemble de relations entre les objets. L'approche cognitive est une approche relationnelle. Une grande complexité (créée par le nombre et le type des relations) se trouve au coeur même de cette vision. En conséquence se trouve remise en question la linéarité causale, "linéarité au double sens du terme : comme succession temporelle *en ligne* de cause et d'effets, et comme proportionnalité entre la cause et l'effet" [4].
Les concepts de degré d'agrégation (ou de résolution), d'ajustement rétroactif, d'interdépendance, d'invariance, de différentiation etc..., deviennent ainsi les notions de base à manipuler.

Les praticiens de l'Analyse Décisionnelle remarquent qu'une analyse cognitive peut servir d'aide à la décision* et souvent ils le déplorent.

---

* On rappelle à cet égard la lourde responsabilité du scientifique

En effet une révision totale de l'analyse est absolument nécessaire parce que la vision du système est totalement distincte suivant que l'on impose des objectifs au système (décisionnel), ou que l'on cherche ses objectifs en fonction de l'aspect de l'étude retenue (cognitif). Cette révision se traduit par une redéfinition du contexte, de la temporalité, de la "validité", de la mise en place des éléments (variables, mécanismes, critères, etc) qui permettent de prendre une décision, et par l'élimination de variables et relations qui sont "inutiles" à la décision.

2 . LA DYNAMIQUE DES SYSTEMES SOCIAUX ET LA PROSPECTIVE

La spécificité des Systèmes Sociaux se révèle notamment par la constatation que l'application pure et simple de méthodes d'analyse élaborées pour l'étude d'autres systèmes est rarement satisfaisante. Ceci tient au fait qu'un système social ne peut pas en général être réduit à "une collection d'objets liés entre eux que l'on isolera artificiellement du reste du monde" ... rassemblés dans une *boite* où tout ..." ce qui lui est extérieur peut agir par l'intermédiaire des entrées, lui même agit à l'extérieur (ou est observé de l'extérieur) par ses sorties" [5].

En effet on ne dispose pas pour les systèmes sociaux comme pour la plupart des autres systèmes de l'évidence de frontières par rapport à l'environnement. Ceci est d'autant plus vrai que "l'objectif examiné doit être replacé dans un contexte plus large que son système originel" [2]. Les systèmes sociaux étant par nature ouverts [1] sont tels que les flux d'entrée et sortie depuis et vers l'extérieur sont très souvent déterminés par le système lui-même. D'autre part toutes les variables du système modélisé doivent être potentiellement observables. La validation de systèmes sociaux par la voie de l'expérimentation en est d'autant plus précaire.

Les sciences du contrôle automatique s'intéressent plus spécifiquement aux liens qui agissent sur le système de l'extérieur. Dès que l'on travaille avec des hommes ou des sociétés le déterminisme décisionnel devient presque impossible à identifier. Car les systèmes sociaux comme la plupart des systèmes vivants peuvent réguler "les rapports avec l'extérieur moins par une politique *d'importation* que par des modifications internes, des variations structurelles du système" [4], [6]. Cette remarque ne met pas en cause la validité, toujours restreinte, de certains schémas décisionnels s'ils sont accompagnés d'une étude sur les limites et conditions d'application de ces techniques.

Lorsque l'étude de systèmes amène à s'interroger sur les options temporelles prises ou à prendre dans ces systèmes (*les possibles latéraux*), lorsque la temporalité est partie intégrante des structures étudiées en Sciences Sociales, l'Analyse de Systèmes rejoint la Prospective, une Prospective qui se veut anticipation, qui se veut décision mais qui avant tout se veut connaissance et compréhension. On ne va pas entrer dans les détails, qui certes mèneraient loin, des relations entre Prospective et Analyse de Systèmes. L'ouvrage de Yves Barel [2] marque le début d'une réflexion en ce sens.

Cette réflexion a déjà le mérite de démontrer l'insuffisance d'une approche sectorielle en Sciences Sociales. La Prospective, ayant pour vocation la convergence de plusieurs disciplines, a parfois revêtu la forme décevante d'une transposition ou d'une juxtaposition de méthodes issues de plusieurs disciplines. On est seulement au début d'un enrichissement réciproque des disciplines, de nombreux concepts des Sciences Sociales résultent déjà de cet enrichissement : régulation, transmission d'informations, reproduction, recherche de l'équilibre, rupture, tension, invariant, continuité, pour n'en citer que quelques uns.

Parmi les multiples outils auxquels la réflexion prospective a fait appel, il y a l'étude de la dynamique sociale. Celle-ci considérée comme évolution temporelle des variables qui composent le système est une partie limitée mais importante de la Prospective. Elle se concrétise pour l'instant en ensembles d'équations continues (certainement à cause du niveau d'agrégation) imbriqués les uns dans les autres. Ainsi les relations entre composants du système sont exprimées de façon quantitative. C'est à cette approche que l'on s'intéresse ci-après, et à laquelle on souhaite contribuer en prenant notre part de la vaste entreprise de formalisation qu'affrontent les Sciences Sociales aujourd'hui. But trop modeste, sans doute, pour les praticiens de la Recherche Opérationnelle, car son utilité ne se mesure pas par la recherche d'un optimum ; but trop ambitieux, sûrement, pour les théoriciens des Sciences Sociales, car la formalisation ne passe pas seulement par la quantification.

Conscients des limites de l'approche, nous constaterons que c'est une des rares méthodes, pour l'instant, qui permette de déceler la complexité du monde social. Mais d'autres outils se révèlent immédiatement complémentaires : par exemple l'analyse de données et la théorie des mutations.

3 . MODELES DE SIMULATION ET SCENARIOS

La classe de modèles à laquelle on fait référence *implique la mise en formule de relations entre certains facteurs, c'est-à-dire la sélection de variables et leur mise en place* [7]. Cette formulation doit rendre compte directement l'évolution temporelle. Sa forme la plus simple est l'équation différentielle de premier ordre, par rapport au temps.

Mais la mise en place d'un problème même de faible complexité est cependant très difficile, d'où le recours à deux techniques : une au niveau de la description du problème que nous appellerons MODELISATION, l'autre au niveau de l'utilisation de cette description que nous appellerons SIMULATION

La **Modélisation** est un ensemble de concepts, méthodes et langage (contraintes syntaxiques, fonctions, etc) utilisée comme outil en vue d'une formulation *rapide* et *correcte* du problème.

La **Simulation** dans ce contexte, doit être comprise non seulement comme **la solution à un (ou plusieurs) problème mathématique par des méthodes d'approximation**, là où les méthodes analytiques ont échoué, mais encore comme un **ensemble de ressources permettant de construire, tester, valider, résoudre et analyser un modèle formel**. C'est une sorte d'expérimentation virtuelle qui permet d'apprendre à travers les solutions particulières dans le temps, le fonctionnement dynamique et global du système modélisé.

On peut envisager d'étudier des systèmes d'une grande complexité en faisant des approximations tant au niveau de la simulation qu'au niveau de la modélisation, dans ce dernier cas pour rendre compte de certains phénomènes *qualitatifs* dont la quantification est difficile mais néanmoins nécessaire à la compréhension du système [8].

L'utilisation de techniques n'élude en rien le problème fondamental de la validité d'un modèle. En effet les modèles de simulation risquent de rester des jeux utopiques [9], *exercices sur les possibles latéraux* [10] (ce qui est déjà quelque chose !). Une **confrontation**, avec la réalité (même perçue subjectivement), reste nécessaire pour que l'analyse fournisse un apport concret à la Prospective, sans exiger pour autant une **parfaite conformité** des résultats à cette réalité.

Un modèle en Sciences Sociales n'est jamais neutre [11], [12], car la formulation est toujours entachée d'objectifs implicites. Il n'existe pas de modèle utilisable dans toute situation et son utilisation ne rend compte que de son fonctionnement dans cette situation.

On peut en expérimenter ainsi un grand nombre, mais puisqu'on est dans le champ du continu, il est exclu de l'explorer tout entier. Il est d'autant plus important de bien choisir les situations qui seront expérimentées parmi celles qui ont une signification, à travers l'ensemble des hypothèses structurelles, dynamiques et de situation (reflétées par le modèle et le jeu de données).

Cette dernière démarche doit être vue comme une confrontation entre les résultats obtenus par la simulation et un **scénario** de l'évolution du modèle de système construit auparavant.
Par **scénario** on entend une suite (croissante par rapport au temps) d'images des états futurs possibles du système, compte tenu de certaines hypothèses dynamiques ainsi qu'une connaissance du système dans l'état actuel [13], [14], [15] ; deux exceptions : le scénario d'anticipation et le scénario par contrastes dont les démarches sont en sens inverse dans le temps (décroissant).

La technique des scénarios aide à la construction d'un modèle car d'une part, elle permet "d'apprécier les inter-relations des paramètres caractérisant le phénomène étudié et de dégager les paramètres les plus importants à retenir" [16] (**Analyse synchronique ou causale**) et d'autre part "s'efforce d'appréhender l'évolution dans le temps de ces paramètres et de ces relations" [16] (**Analyse diachronique**).

Il est évidemment nécessaire en Prospective de concevoir non pas un seul, mais un ensemble suffisant de scénarios, car il n'y a pas un seul futur, mais une **multitude de futurs possibles** [17]. Dans ce sens un scénario n'est pas meilleur qu'un autre et le résultat des différentes simulations a toujours une valeur indicative. Celui qui va interpréter ce résultat conserve l'entière responsabilité des conséquences qu'il en tire, qui ne sont aucunement *validées* du fait de l'utilisation d'un modèle.

4 . ANALYSE, MODELISATION ET SIMULATION

Le processus d'analyse commence bien avant la modélisation et la simulation. Il les englobe et les dépasse, car une fois la simulation effectuée, il faut souvent revenir en arrière, confronter, changer, ..., en un mot réanalyser.

Les étapes de ce processus complexe ne peuvent être codifiées, mais elles évoluent en fonction des améliorations méthodologiques et techniques qui viendront les compléter. Les principales étapes actuellement envisageables sont montrées dans la figure I.1. L'ordre choisi pour leur enchaînement est purement indicatif, de nombreuses interactions se produisent entre une étape et une autre. Seule la définition claire des buts de l'étude est une étape préalable à toutes les autres.

Le degré de complexité que peuvent atteindre les modèles qui nous intéressent demande des aides programmées supérieures à celles offertes par un langage de programmation [18]. Rappelons en particulier, les principaux avantages que l'on a pu tirer de l'expérience dans la définition du langage LADESH [19] :

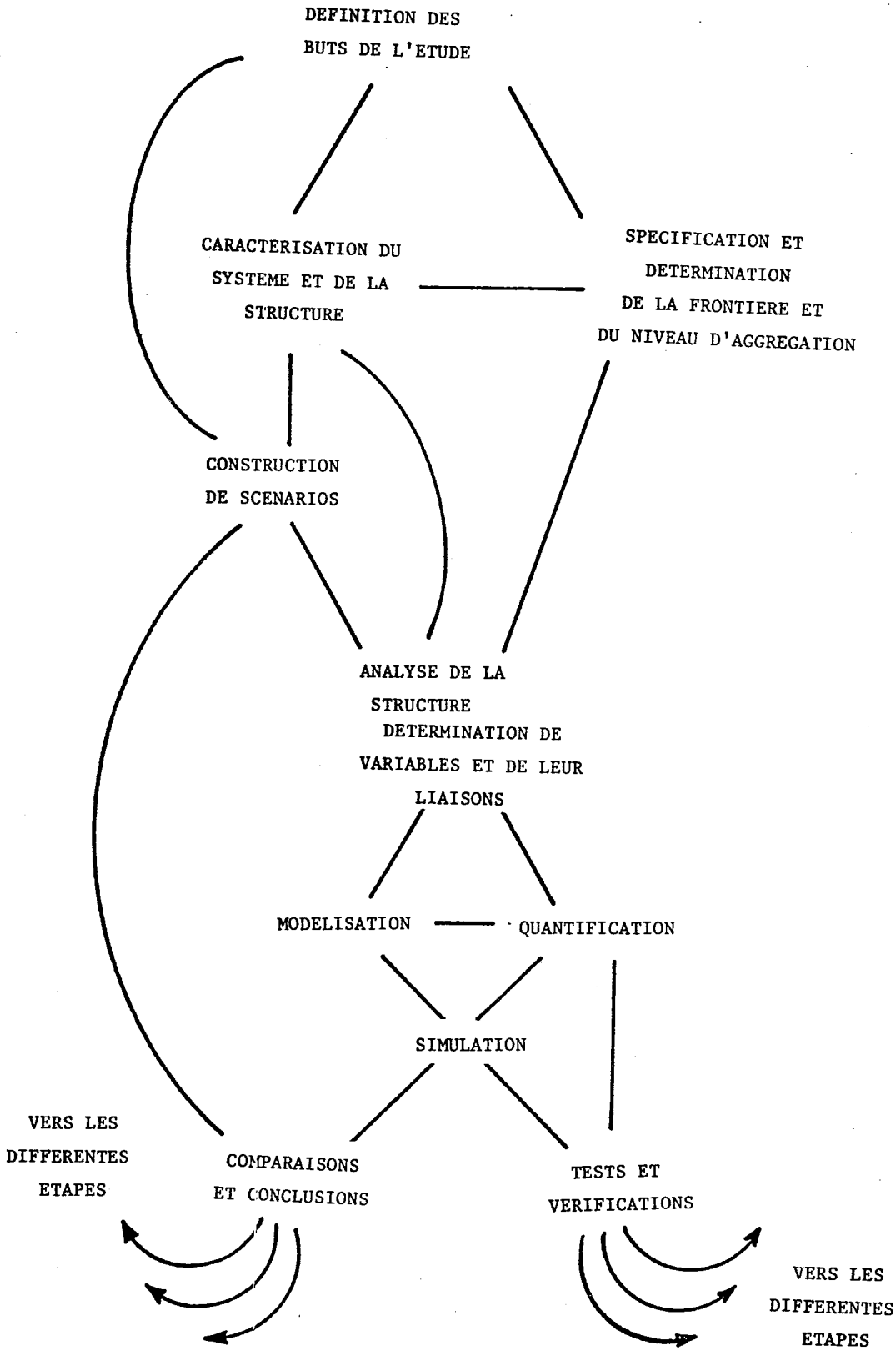
- L'intégration d'une certaine vision d'un type particulier de systèmes *socio-économiques* fait appel aux notions de modules et de hiérarchie, au tavers d'une double partition géographique et logique. Ceci requiert l'introduction d'une sémantique particulière.

- La séparation des données par rapport au corps de la description, permet d'effectuer à part le processus de quantification.

- Finalement une *méthodologie* de conception et description descendantes sont aussi intégrées et clairement exposées dans la thèse de F. Rechenmann [20].

Figure I.1

U N P R O C E S S U S D ' A N A L Y S E



Le processus de simulation qui n'est qu'une partie de l'analyse, mais aussi le point culminant d'une série d'efforts, nécessite des aides informatiques importantes pour accomplir cette tâche de façon claire et apparemment simple. Simuler, ce n'est pas seulement résoudre un ensemble d'équations mais fournir tout type d'aides informatiques dont on peut avoir besoin pour sa manipulation. C'est ainsi que nous avons cherché à concevoir un **Système Informatique de Simulation.**

Nous essaierons maintenant de faire un survol des principales caractéristiques de systèmes informatiques de simulation continue existants.

5 . "LES LANGAGES DE SIMULATION"

Pour l'instant tous les langages de modélisation continue ont un système de simulation qui leur est associé ; de ce fait on emploie couramment dans la littérature le terme impropre de *langage de simulation*. Nous nous sommes intéressés dans ce travail à l'aspect système de simulation, quoique l'on fasse toujours référence à son complément informatique : le langage de description de modèles.

Il existe à l'heure actuelle une telle quantité de "langage" dits *de simulation* que leur énumération et leur classification seraient trop longues : Nilsen et Karplus [21] en ont dénombré plus de quarante parus dans la seule année 1973.

Il ne s'agit pas de faire dans la présente partie du chapitre une étude exhaustive des *langages de simulation* ; ceci a déjà été tenté à plusieurs époques [21], [22], [23], [24], [25]. Nous allons par contre essayer de dégager les grandes lignes et options prises dans les **systèmes de simulation** les plus courants.

Nous nous intéressons essentiellement à la simulation continue c'est-à-dire celle des systèmes décrits par des équations différentielles et des relations algébriques.
En général on considère aussi l'intervention des évènements discrets qui changent soit un paramètre, soit une partie de la description continue du modèle. Il ne s'agit pas là des évènements discrets qui dirigent ou contrôlent la simulation, quoique certains langages continus ou plutôt mixtes, permettent des instructions algorithmiques du type : IF-THEN, GOTO etc...(voir § I.9).

6 . SPECIALISE CONTRE GENERAL

Il me semble que le point le plus controversé pour le concepteur des systèmes de simulation continue soit le dilemme systèmes de simulation généraux - systèmes spécialisés dans un domaine d'application.

En général les systèmes spécialisés sont plus faciles à apprendre et à utiliser car les termes et concepts employés sont mieux adaptés au type de modèles qu'on traite et aux utilisateurs. Ils sont aussi moins lourds à utiliser.

Le temps généralement nécessaire pour décrire le problème dans l'ordinateur est moindre pour les systèmes spécialisés que pour les systèmes généraux, la contrepartie étant une difficulté d'extension si le besoin s'en fait sentir.

Les extensions sont mieux adaptées et plus faciles à mettre en oeuvre dans les systèmes généraux.

La plupart des systèmes n'ont pas d'outils d'extensions ou ceux-ci sont très rudimentaires : macros mal adaptées, insertion de routines PL1 ou FORTRAN etc... Par contre au niveau du langage de modélisation la majorité des systèmes possèdent de tels mécanismes.

En ce qui concerne le temps de simulation pour un même modèle, il n'est pas évident qu'un système spécialisé soit plus performant qu'un système général ; cela dépend d'abord de sa mise en oeuvre en tant que logiciel sur un même matériel. Il est clair que les systèmes spécialisés du type question-réponse sont de gros consommateurs de temps-machine et leurs qualités pédagogiques sont souvent présentes.

Il existe en outre des systèmes de simulation assez généraux, mais qui visent un domaine d'application particulier ; c'est le cas de TROLL [26], [27] qui est fait pour des applications économétriques, c'est-à-dire pour des modèles décrits par des équations simultanées aux différences, (donc ce n'est pas de la simulation continue ni non plus de la simulation discrète classique ou évènementielle). TROLL est très orienté vers les problèmes de séries temporelles et d'ajustement statistiques. Son usage en continu demande à l'utilisateur de programmer ses méthodes d'intégration, de discrétiser et d'ordonner ses équations. D'autre part TROLL est réalisé comme un système spécialisé d'exploitation interactif pouvant être utilisé uniquement sous certains systèmes de temps partagé, ce qui accroît sa difficulté d'utilisation. Vouloir par ailleurs utiliser TROLL hors du domaine pour lequel il a été conçu [28] met en évidence ses limitations et ses performances réduites.

7 . ORIENTE MACHINE OU ORIENTE PROBLEME

Deux grandes options s'ouvrent alors à la conception d'un système de simulation : le faire "orienté-problème", c'est-à-dire avoir une structure qui s'approche du processus de la simulation, ou bien "orienté-machine" si le système s'approche d'un langage de programmation de bas niveau et si sa structure est déterminée par la machine et le système d'exploitation.

Ce que l'on gagne en vitesse de traitement avec un système orienté-machine est perdu en souplesse, compréhension, et facilité d'utilisation.

Pour l'instant rares sont les systèmes orienté-machine comme MIMIC [29], mais ils ont tendance à réapparaître avec l'introduction de minis et micro-ordinateurs sur le marché à des prix très compétitifs [30], [31].

Comme la grande majorité des concepteurs de systèmes de simulation nous pensons que le principal avantage (par rapport aux langages de programmation courants) est précisément de faciliter la tâche déjà complexe de la simulation avec un langage de haut niveau et une structure plus adaptée au problème et nous avons orienté les choix en conséquence.

8 . LES METHODES D'INTEGRATION NUMERIQUE

Nous sommes amenés à considérer à part les systèmes dans lesquels on traite des équations aux dérivées partielles (comme LEANS, FORSIM, DSS), car les méthodes d'intégration sont fort différentes dans ce cas.

En principe un système qui se veut général devrait pouvoir traiter aussi bien des dérivées partielles que des équations différentielles.

C'est l'application qui doit déterminer les méthodes d'intégration à employer. Ainsi les systèmes de simulation IMAG II [32] et IMAG III [33] ont des méthodes d'intégration qui s'adaptent aux besoins de différents types de circuits électriques et leur simulation.

En Sciences Sociales, les systèmes sont pour l'instant rarement formulés en termes d'équations aux dérivées partielles. C'est toutefois le cas pour certains phénomènes de diffusion spatiale [34] ; mais on peut parfois l'éviter [35]. En effet d'autres solutions comme le découpage de l'espace en cellules régulières pourraient être envisagées [36].

On considère qu'en Sciences Sociales une précision n'est pas souvent requise dans la formulation d'un modèle dynamique. Il n'est donc pas nécessaire d'offrir une large gamme de méthodes.

En effet des méthodes très précises comme Runge-Kutta (ordre 4) ou les méthodes très stables comme les méthodes implicites (Gear, Adams, etc...) dépassent de loin les besoins actuels ; de même la double précision dans la représentation interne de chiffres qu'offrent certains systèmes pour ces méthodes n'est pas non plus indispensable, néanmoins l'utilisation de la seule méthode d'Euler comme dans DYNAMO [37] se révèle insuf-sisante (voir annexe IV).

Les systèmes généraux comme CSMP/CMS [38], CSSL-III [39], CSMP-III [40] ou ACSL [41] prévoient et offrent davantage de méthodes d'intégration aux dérivées ordinaires. On leur ajoute parfois des moyens pour optimiser leur utilisation lorsqu'on traite des processus ayant des constantes de temps très différentes [42] ; on cherche parfois même à donner la possibilité de sélectionner différentes algorithmes d'intégration en fonction des différentes parties du modèle [43]. Mais en général les algorithmes d'intégration sophistiqués se montrent très inadéquats lorsque l'on traite des systèmes continus ayant des évènements discrets [25].

8 . LA GESTION DES EQUATIONS

La gestion des équations qui forment le corps d'un modèle est traitée de différentes façons en fonction des possibilités descriptives du langage de modélisation.

Le fait de pouvoir traiter des équations algébriques implicites nécessite des outils particuliers dans le système de simulation. En effet une équation algébrique implicite signifie l'existence d'une boucle algébrique sans intégration, et la résolution d'une telle équation demande un calcul iteratif jusqu'à l'obtention d'une solution acceptable, bornée par une erreur maximale donnée. Des systèmes comme CSMP/360 [38] et CSMP-III [40] permettent de calculer automatiquement ces équations implicites.

Des systèmes de simulation comme SLANG et PROSE [21] permettent en plus de l'intégration numérique, l'optimisation non-linéaire des ensembles d'équations algébriques implicites, en se servant en particulier du calcul de la matrice Jacobienne ($\partial f_i / \partial X_j$) à chaque iteration.
De ce fait, ces systèmes sont orientés plutôt vers l'optimisation des systèmes continus que vers leur simulation.

Les systèmes autorisant des évènements discrets en dehors de leur utilisation comme fonctions dans un système continu, posent des problèmes de gestion tout à fait particuliers (par exemple : la synchronisation de blocs discrets et de blocs continus).
GSL [44], [45] est un exemple d'un tel type de système de simulation ; et à moindre titre GASP [46], [47], qui est lui strictement discret avec quelques facilités pour l'intégration numérique. La segmentation (voir paragraphe I-15) est une solution plus répandue car elle permet que des produits-programme de simulation continue comme CSSL-III [39] ou DARE-P [55], puissent appeler (ou être appelés par) des programmes de simulation discrete.

Les équations simultanées purement algébriques, en général, ne sont pas acceptées dans les systèmes de simulation continue (car de tels types d'équations n'engendrent pas de dynamisme causal [18]). Certains systèmes généraux de simulation continue disent néanmoins prévoir cette possibilité en laissant à l'utilisateur le soin de décrire l'algorithme de résolution de son système simultané.

Le type et la gestion des équations dépendent évidemment du type de modèle (et donc du problème !) que l'on traite.

9 . PROCEDURE OU PARALLELISME

Un modèle est parallèle si tous ses composants agissent dynamiquement en même temps ; c'est-à-dire s'il n'y a pas de séquence obligatoire (ou procédure) qui détermine l'ordre dans lequel les composants doivent agir (modèles procéduraux).

Si un modèle est parallèle, cela ne veut pas dire que la simulation (calcul sur ordinateur digital) puisse être elle-même parallèle. Le parallèlisme lui aussi est donc *simulé* par l'ordinateur par exemple en déroulant un (ou plusieurs) algorithme d'ordonnancement des équations. Une conséquence en est que tout modèle parallèle est simulé entièrement à chaque iteration dans la simulation.

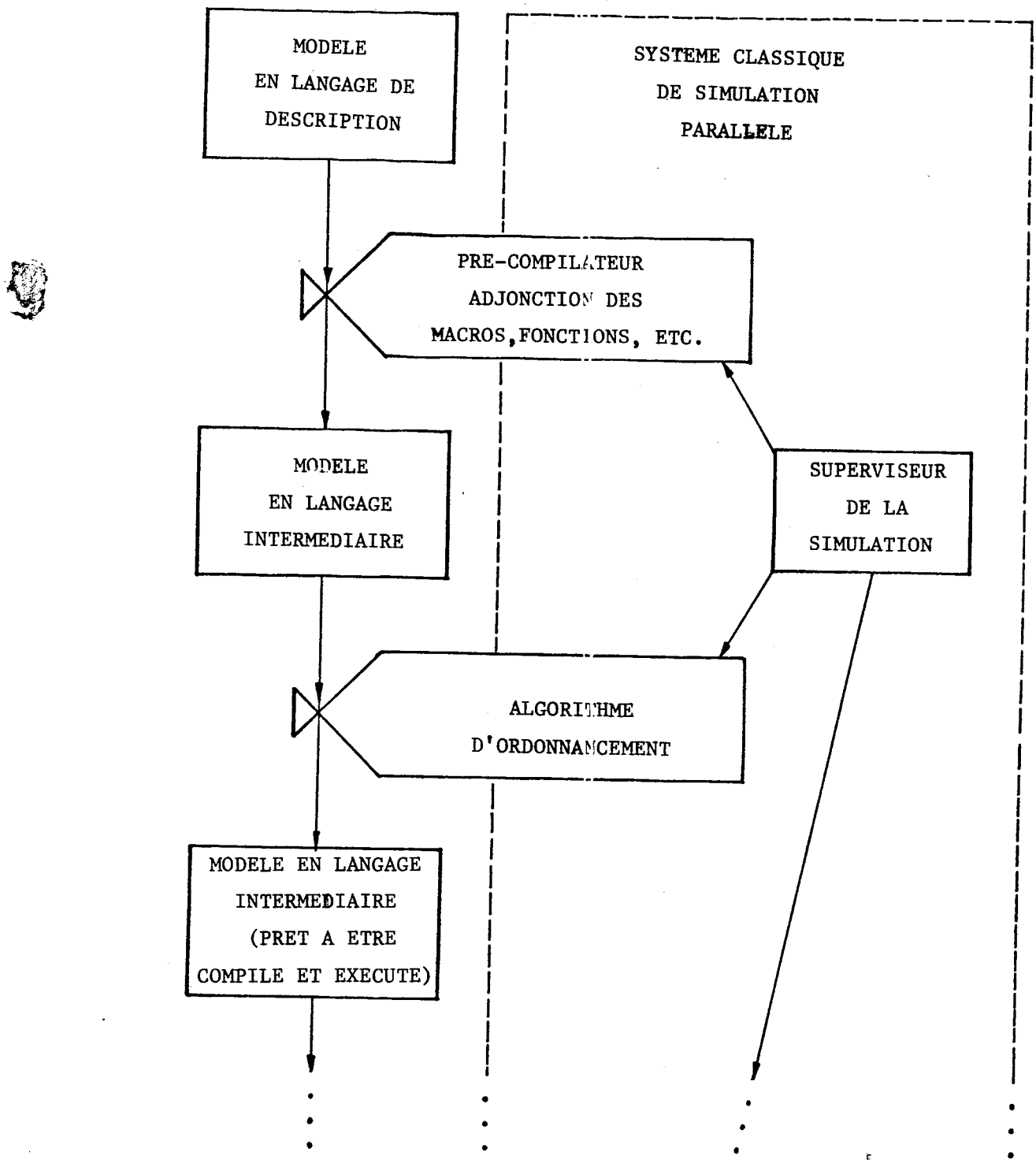
Les langages de modélisation qui acceptent de décrire des modèles parallèles s'appellent *langages parallèles*, dans le cas contraire ils s'appellent *langages procéduraux*.

L'avantage évident des langages parallèles est que l'utilisateur n'a pas à se soucier de l'ordre dans lequel ses équations sont écrites (quelque soit le type de relations qu'il y a entre les variables).

Un autre avantage particulièrement sensible en Sciences Sociales est de pouvoir écrire un modèle qui reflète la structure du système modélisé (ce qui est très important pour la compréhension) sans introduire dans la description du modèle des instructions de séquence qui ne font pas intrinsèquement partie de la description.

La plupart des systèmes de simulation fonctionnent selon le schéma suivant : (figure I.2).

Figure I.2


MODELE
EN LANGAGE DE
DESCRIPTION
SYSTEME CLASSIQUE
DE SIMULATION
PARALLELE
PRE-COMPILATEUR
ADJONCTION DES
MACROS, FONCTIONS, ETC.
MODELE
EN LANGAGE
INTERMEDIAIRE
SUPERVISEUR
DE LA
SIMULATION
ALGORITHME
D'ORDONNANCEMENT
MODELE EN LANGAGE
INTERMEDIAIRE
(PRET A ETRE
COMPILE ET EXECUTE)

Le parallèlisme étant une caractéristique du modèle et du langage, il devrait être aussi une caractéristique du pré-compilateur, parce que c'est le pré-compilateur qui est l'interface avec le langage.

Un pré-compilateur-parallèle, qui en plus de traduire, ordonne les équations, pourrait être alors construit et défini indépendamment du système de simulation.

Malheureusement aucun système de simulation n'a été construit sur cette indépendance ; c'est pour cela d'ailleurs que tous les systèmes de simulation existants sont mono-langages (associés à un seul pré-compilateur et donc associés à un seul langage de description de modèles. De là provient la confusion regrettable entre langage de modélisation et système de simulation.

Il existe des langages de simulation continue tout à fait procéduraux comme GASP-II [46], où l'utilisateur doit toujours décrire sa séquence d'exécution. Pourtant la plupart des langages de simulation, dits *généraux* voulant offrir des possibilités au delà de la simulation continue de systèmes dynamiques (comme par exemple les équations simultanées. Voir § I.8), permettent l'emploi de blocs procéduraux tout en gardant l'étiquette "parallèles". C'est ainsi que dans des langages comme CSSL [39], CSMP [38], [40], etc..., soit à travers des macros soit à travers des blocs dits *PROCEDURE* ou *NONSORT* l'utilisateur peut et doit définir sa propre séquence ; pour ce faire il dispose d'instructions algorithmiques typiques des langages de programmation : GOTO, DO, IF, THEN, etc. Dans ces cas, l'algorithme d'ordonnancement appliqué à la suite de la pré-compilation traite ces blocs comme une unité non-décomposable.

L'algorithme d'ordonnancement de base [49] est très simple, il s'agit d'ordonner les équations pour que toute variable soit calculée avant d'apparaître à droite d'un signe égal (exception faite de paramètres d'entrée, valeurs initiales et de fonctions "mémoire" - fonctions dans lesquelles intervient une intégrale par exemple les *délais* [48]).

A partir de cet algorithme de base on peut raffiner suivant les possibilités offertes par le langage de modélisation : par exemple boucles algébriques ou description par blocs hiérarchiques [50].

D'autres solutions, au niveau système cette fois-ci, sont possibles, comme le passage de n-fois par l'ensemble d'équations parallèles non-ordonnées pour chaque iteration ; à chaque passage on calcule seulement les équations dont les paramètres sont connus [51]. Ce procédé est parfois employé dans la simulation discrète, mais en simulation continue, vu le grand nombre d'iterations (par exemple dû à une méthode d'intégration d'ordre élevé, ou à un pas d'intégration petit) il n'est jamais employé.

10 . EXTENSIONS ET LANGAGE PROCEDURAL ASSOCIE

Lorsqu'on veut faire des extensions à un système de simulation, dont les plus courantes sont par exemple :

- introduire un nouvel algorithme d'intégration numérique,
- mettre de garde-fous avec avertissements dans le déroulement d'une simulation ou,
- simplement créer un format spécial pour l'entête de l'impression des résultats ; on recourre dans les systèmes de simulation existants à plusieurs mécanismes d'extension qui d'ailleurs ne sont pas mutuellement exclusifs. Ces mêmes mécanismes servent la plupart du temps à introduire aussi des nouvelles fonctions dans le langage de description.

• Les **mécanismes de macros** : soit ils sont spécialisés dans l'insertion des instructions formant partie de la description parallèle du modèle ("macro-handler"), en conséquence ces instructions sont vérifiées lors de la pré-compilation (c'est le cas de CMPS [38] ou DYNAMO [37],) soit ils permettent de considérer toutes les instructions d'un langage de programmation procédural comme FORTRAN comme parties du corps de la définition d'une macro (c'est le cas de CSSL [39]). Cette dernière solution n'est pas satisfaisante car le mécanisme est réduit à une insertion et subtitution sans une véritable vérification, le mécanisme ne gérant pas les objets du système n'assure pas l'interface et il est donc très délicat à programmer.

• Les **Blocs "Procédure"** : c'est un groupement d'instructions écrites dans un langage procédural associé comme FORTRAN ou PL1 à des marques de début et fin du Bloc. Ils doivent être insérés dans la description du modèle et sont destinés à être sautés par l'algorithme d'ordonnancement qui divise donc le modèle en sections parallèles. A ces possibilités CSMP [38] ajoute le branchement entre Blocs Procédure ainsi que l'appel à des macros parallèles (celles-ci permettent de faire des macros semi-procédurales ou des procédures semi-parallèles, et à la limite parallèles). En plus de la très mauvaise lisibilité des modèles ainsi décrits leur mise au point est difficile même pour un programmeur expérimenté.

Le langage ACSL [41] est le premier à définir des blocs "PROCED" contenant exclusivement certains ordres de visualisation et de lancement de la simulation.

• Les **fonctions** ou **sous-programmes** : (écrits dans le langage intermédiaire du système qui est en même temps son langage procédural associé). Ces fonctions ou routines doivent être déclarées avant pour les appels soient insérées dans le modèle et que le pré-compilateur les prenne comme des fonctions internes.
Le rôle de ces fonctions est donc réduit à des extensions du langage de description et éventuellement un espionnage du déroulement de la simulation.

... MIMIC [29] fixent même le nom de ces fonctions ...mbre de paramètres, les fonctions doivent être ...ORTRAN mais son langage intermédiaire n'est pas FORTRAN.

... cas l'analyse syntaxique n'est pas complète, et sa tâche ...erminée par le compilateur du langage intermédiaire, d'où l'apparition de certains messages "incompréhensibles".
Mais ce qui est plus grave, c'est que l'ordre d'exécution des équations n'est pas unique car les paramètres des routines peuvent aussi bien créer dans un cas une variable et dans un autre la modifier.

• Il a été suggéré [25] qu'un système de simulation soit divisé en deux programmes-produits indépendants, l'un "interprétatif" qui se charge de faire la pré-compilation et l'ordonnancement du modèle, l'autre "algorithmique" qui fournit les méthodes d'intégration et réalise la simulation. L'utilisateur pourrait agir entre les deux programmes-produits pour modifier, tester et même écrire des extensions en langage intermédiaire. Cette solution est acceptable pour la mise au point d'un produit-programme par ses concepteurs, mais il est hors de question qu'un utilisateur extérieur même averti puisse le faire aisément.

La plupart des systèmes de simulation offrent donc des extensions par le biais d'un langage procédural associé (ou auxiliaire) qui est généralement leur langage intermédiaire.
La généralité de ces systèmes découle de la grande ressemblance de leur langage à un langage de programmation classique mais qui n'offre aucun guide ni syntaxique, ni sémantique ni au niveau du modèle ni à sa simulation. Pourtant ces guides sont extrêmement utiles en Sciences Sociales.

11 . CONCEPTION ASSISTEE PAR ORDINATEUR

Deux grandes aides à la conception des modèles ont été déjà évoquées : la possibilité de décrire le parallélisme et l'utilisation des contraintes syntaxiques du langage de modélisation (comme dans DYNAMO [37] ou LADESH [19]).

Mais en ce qui concerne la simulation, presque rien n'a été fait. Il faut cependant mentionner DYNASYS [52] et OGACIMSS [53] langage-système dérivé de DYNAMO qui permet de construire, modifier et simuler un modèle en utilisant en outre les symboles graphiques de la Dynamique des Systèmes [54].

Deux autres programmes-produits [25] dérivés de DARE-P [55] sont en cours de développement, pour lesquels on prévoit des outils pour changer de façon interactive la structure d'un modèle décrit comme un circuit analogique.

Enfin, il existe des études pour la réalisation de systèmes interactifs de simulation qui devraient permettre un ajustement semi-automatique de données par des méthodes d'"essai et erreur" [56] ou si l'ajustement peut être défini par une fonction objective (contrôle) on s'attache à mettre en oeuvre des méthodes comme celle du Simplex Flexible [57].

12 . ANALYSE ET VISUALISATION

La sortie graphique ("display") des résultats tend à se développer. En particulier CSMP-III [40] - son prédécesseur CSMP/360 n'a pas cette possibilité, il n'est même pas conversationnel - a un programme-produit optionnel pour être connecté à un terminal graphique IBM-2250, les entrées étant toujours faites à travers le clavier. DYNASIS [52] implémenté sur B-6700 utilise un terminal graphique Tektronix T-4002-A, OGACIMSS [53] un écran graphique CSE-AFIGRAF connecté à un MITRA 15, les entrées se faisant soit par clavier, soit sur tablette ou sur l'écran avec l'aide d'un photostyle.

Le manque de standardisation du matériel graphique fait que tous ces programmes-produits doivent être mis en oeuvre sur un matériel informatique particulier et dans une configuration bien spécifique ; en conséquence ils ne sont pas compatibles entre eux.

L'analyse des résultats de la simulation est très souvent négligée. Les systèmes de simulation existants permettent rarement de faire les comparaisons de résultats pour une même variable entre plusieurs simulations ; l'utilisation d'autres moyens d'analyse comme le calcul de la moyenne temporelle, la comparaison à travers des histogrammes où les graphes en espace bidimensionnel n'ont pas été inclus, COLTS [58] et ACSL [41] permettent toutefois la visualisation d'espaces de phases.

13 . SYSTEMES CONVERSATIONNELS

Il se révéle une tendance de plus en plus grande à utiliser des systèmes et des langages interactifs dans le domaine informatique en général ; la simulation ne fait que suivre de loin cette tendance [59]. En effet, les systèmes de simulation ayant une interface conversationnelle [60], sont encore peu nombreux.

Mais dans certains systèmes des interfaces interactives plus ou moins modestes ont été incorporées. Dans d'autres systèmes on trouve un moniteur conversationnel, qui surveille la simulation pas à pas, et permet en outre d'arrêter la simulation et de changer des paramètres en cours de route [60] ; cela a des inconvénients, surtout si la visualisation est graphique, ce qui donne un cadrage et une échelle en général mauvais, d'autre part l'exploitation est généralement très coûteuse.

Il existe enfin, des systèmes de simulation spécialisés qui offrent des possibilités de choix en cours de simulation ; par exemple WISSIM [61] permet de simuler des modèles énergétiques, avec changement interactif des politiques énergétiques.

14 . SYSTEMES DECISIONNELS

Certains systèmes de simulation continue sont interactifs parce que les modèles eux mêmes sont interactifs, (modèles qui demandent à l'utilisateur d'indiquer dans certaines étapes programmées la (les) option(s) à suivre).

L'interface interactive est incorporée au modèle, ce qui fait un système de simulation figé sur un seul modèle.

Il y a deux raisons principalement pour lesquelles les décisions sont prises de façon dynamique :

- Soit parce que la simulation est partie d'un jeu de stratégie décisionnelle où l'objectif n'est ni la simulation, ni le modèle, (que d'ailleurs on suppose correct parce qu'on connaît très bien le système modélisé) mais le comportement du (des) joueur(s).
En effet, il s'agit soit d'aider le décideur à évaluer les conséquences d'une politique qu'il doit appliquer (par exemple dans les jeux d'entreprise), soit d'étudier les réactions, enseigner ou entraîner un gestionnaire aux problèmes de la prise de décision.

- Soit parce que le processus de décision est fort complexe à saisir (et donc à simuler) et qu'on évite le problème en laissant à l'utilisateur le soin de prendre la décision comme s'il était une partie du système.
Un exemple de cette dernière option est l'ensemble des modèles faits pour le deuxième rapport au Club de Rome par Mesarovic Pestel [62], [63].

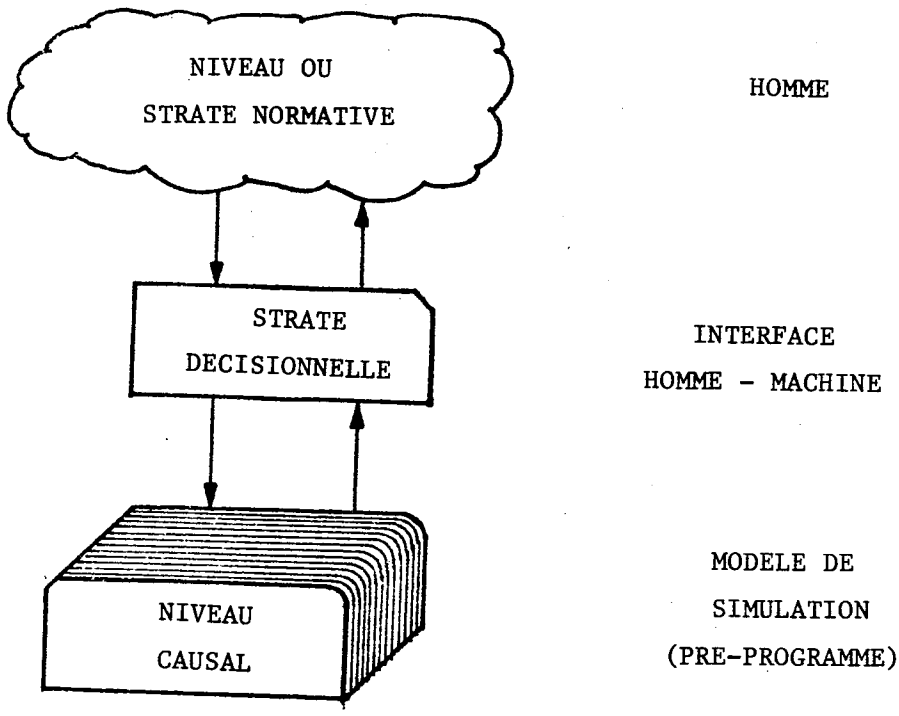
Ces derniers modèles correspondent à une philosophie des systèmes bien précise [64], où dans tout système social, il existe trois strates ou niveaux : normatif, décisionnel et causal. Le normatif est pris en main par l'homme, le causal par le modèle de simulation et le décisionnel par l'interaction homme-machine [65], [66], [67], [68], (voir figure I.3).

On a donc tendance à introduire le plus de normatif au niveau décisionnel et de décisionnel au niveau causal. Cette idée a même été envisagée avec plusieurs strates normatives (plusieurs joueurs) interconnectées [69].

Figure I.3

N I V E A U X O U S T R A T E S

S E L O N M E S A R O V I C



La critique des options de simulation prises par Mesarovic Pestel a été developpée en [70], [71].

Un système de simulation décisionnel en Sciences Sociales devrait permettre non pas seulement de lancer des simulations pré-programmées et observer ces réactions, mais de connaître et tester le modèle.

15 . LE RAPPORT CSSL [72]

(Continuous System Simulation Languages)

L'intérêt de ce rapport est que la grande majorité des systèmes de simulation construits depuis, ont suivi de près ou de loin ses spécifications.

L'apport principal du rapport a été sans doute, de conceptualiser et donc de permettre une construction des étapes de la simulation. Une étude selon CSSL doit être composée de plusieurs simulations à un changement près des valeurs initiales.

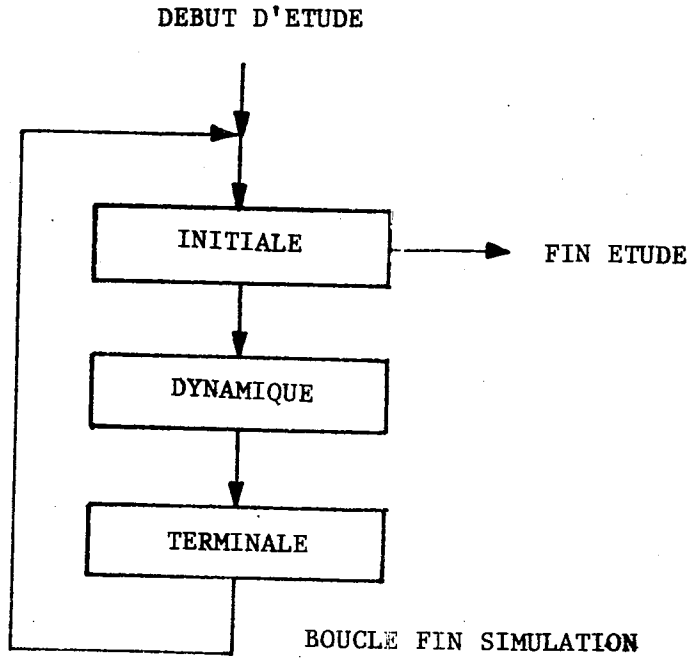
Une **simulation** a trois étapes :

. initiale,
. dynamique, et
. terminale

(voir figure I.4).

Figure I.4

E T A P E S D E L A S I M U L A T I O N



• L'**étape initiale** est totalement procédurale, celle-ci est subdivisée en deux parties. Une, "interprétative", qui initialise les variables indépendantes et les vecteurs d'état, fixe les conditions d'arrêt de la simulation, et détermine la méthode d'intégration et ses paramètres. La deuxième partie fait une fois pour toute les calculs initiaux.

• L'**étape dynamique** déroule la simulation proprement dite, c'est-à-dire les calculs intermédiaires et les opérations d'intégration pour chacun des ensembles d'équations indépendantes. Certains auteurs divisent cette étape en sections parallèles et sections dérivatives (tout ce qui n'est pas parallèle)

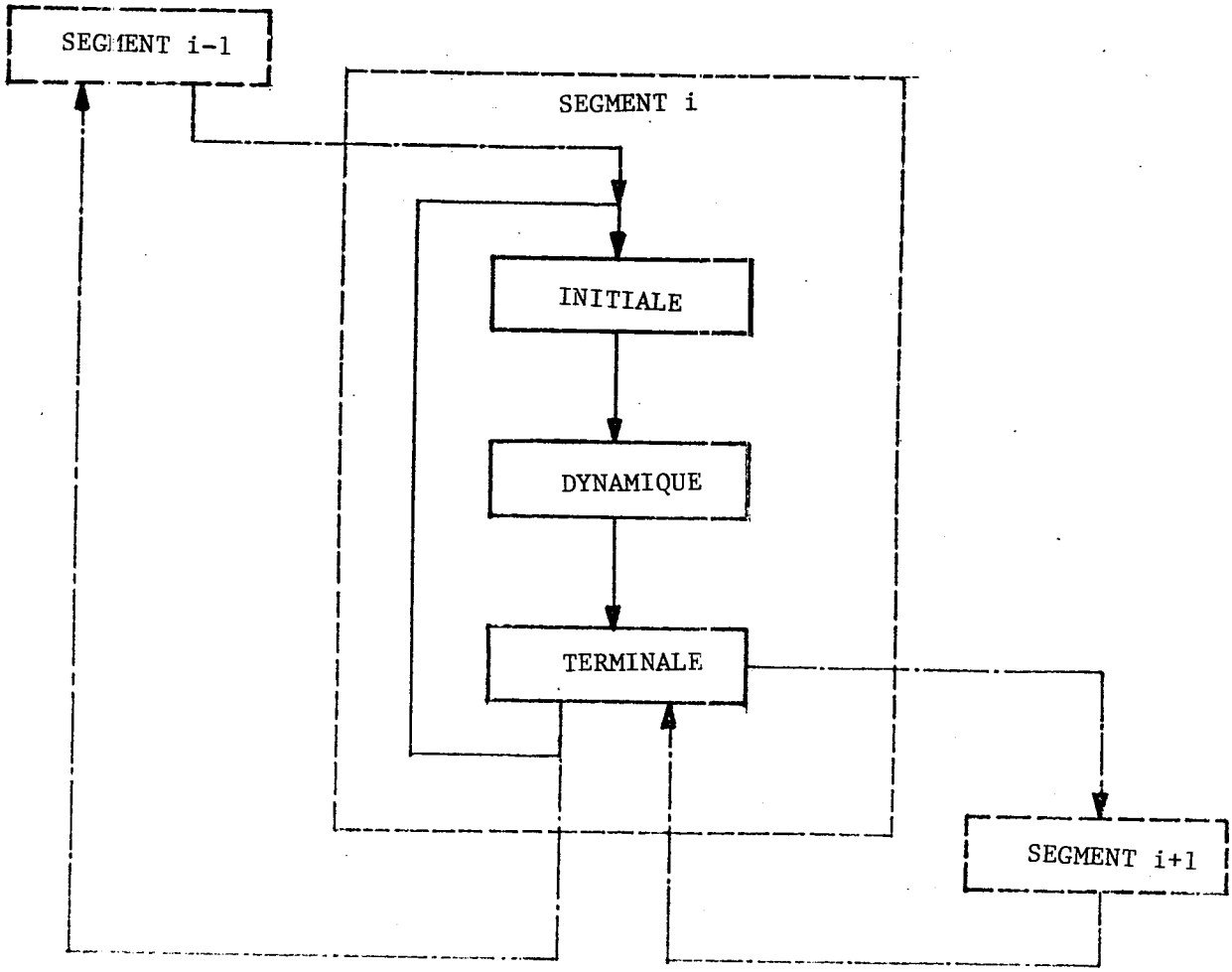
• L'**étape terminale** fait les opérations qui exigent que la simulation soit terminée (i.e. temps machine consommé, calcul de l'écart type, certaines sorties graphiques, etc...). Elle rend le contrôle à l'étape initiale soit pour une nouvelle simulation, soit pour sortir du système si l'étude est terminée.

Le rapport CSSL autorise l'impression des résultats dans n'importe quelle étape, mais les spécifications des sorties graphiques seulement dans l'étape initiale interprétative. Ainsi ACSL [41] en gardant toujours la structure CSSL, accepte exclusivement les instructions de visualisation dans l'étape terminale. D'autres non CSSL l'avaient déjà adopté [73], [74].

Une autre notion intéressante du rapport CSSL est la **segmentation** qui permet au processus de simulation d'être considéré comme une partie d'un processus plus large ; chacun de ces sous-processus est appelé un segment (voir figure I.5).

Le processus de segmentation permet de calculer par simulation les conditions initiales d'un modèle et de démarrer se simulation, il permet aussi de changer de variables indépendantes d'un modèle et de le réinitialiser, enfin d'avoir un ensemble séquentiel de modèles parallèles indépendantes. La segmentation est néanmoins utilisée quasi-exclusivement pour "optimiser" ou "contrôler" un modèle de simulation. Malheureusement elle n'a pas encore été employée pour les problèmes propres à la modélisation comme la hiérarchie des modèles ou leur restructuration (car celles-ci demanderaient une catégorie de "segments" non seulement connectable sous forme linéaire mais aussi sous forme de "réseaux", voir chapitre V).

S E G M E N T A T I O N



La structure de segmentation séquentielle est utilisée à l'insu de l'utilisateur par certains systèmes comme le moyen d'admettre des événements discrets que ces algorithmes d'intégration numérique traitent mal. Dans ces cas l'événement est effectué dans un segment précédent et on réinitialise des segments avec de nouvelles conditions initiales.

Enfin le rapport CSSL donne des recommandations techniques qui vont d'un mécanisme de macro-définition jusqu'à la description d'un modèle comme ensemble de blocs interconnectés, ou à la définition des fonctions par exemple celles qui correspondent à des retards discrets.
Prochainement un nouveau rapport CSSL sera fait.

BIBLIOGRAPHIE

[1] VON BERTALANFFY L.
"Théorie générale des systèmes"
Dunod, Paris, 1973

[2] BAREL Y.
"Prospective et Analyse de systèmes"
Travaux et recherches de prospective N° 14
La Documentation Française, 1971

[3] HITCH C.J., Mc KEAN R.N.
"The economics of Defense in the Nuclear age"
Harvard University Press, Cambridge, 1961
cité et traduit par [2] p. 81

[4] BAREL Y.
"L'idée de système dans les Sciences Sociales"
Journée A.F.C.E.T. "l'analyse de système : techniques, expériences, tendances" Paris, avril 1976

[5] FOULARD C., GENTIL S., SANDAZ J.P.
"Commande et régulation par calculateur numérique"
Eyrolles, Paris 1977

[6] BAREL Y.
"La reproduction sociale ; systèmes vivants, invariants et changement"
Anthropos, Paris 1973

[7] GRAWITZ M.
"Méthodes des Sciences Sociales"
Dalloz, Paris, 1974

[8] STRANCH P.G.
"Modeling in the Social Sciences : an approach to good theory and good policy"
dans Simulation Today, nb 39, SIMULATION, Vol. 26, nb 1 January 1976, pp. 153-156

[9] RIVERA E.

"L'utopie et les modèles de simulation en Prospective", rapport DEA, IPEPS-IREP, Université des Sciences Sociales de Grenoble, 1975

[10] RUYER R.

"L'utopie et les utopies", PUF, Paris 1970

[11] RIBEILL R.

"Modèles et Sciences Humaines", Metra Vol. XII, n° 2, 1973 pp. 271-303

[12] RECHENMANN F., RIVERA E., UVIETTA P.

"A case study on social responsability and impact" Proceedings of the 7th IFIP Conference on Optimization Techniques. Part 1, pp. 431-439, Springer Verlag 1976

[13] CROUSSE B.

"La méthode de scénarios" note interne IPEPS-IREP, 1972

[14] GRAS A.

"Clefs pour la Futurologie" Seghers, Paris 1976

[15] D.A.T.A.R.

"Scénarios d'aménagement du territoire", Travaux et Recherches de Prospective n° 12, La Documentation Française, 1971, Paris

[16] SAINT-PAUL R., GALZAIN M.

"La prévision technologique" (des scénarios) dans le Progrès Scientifique N° 156-157, Sept-Oct. 1972, pp. 27-31

[17] DECOUFLE A.C.

"La Prospective", PUF, Paris, 1972

[18] RABINS M.

"Modélisation des systèmes informationnels"
notes de cours E.N.S. d'Electrotechnique et de Génie Physique, 1975

[19] RECHENMANN F., RIVERA E., UVIETTA P.

"A modelling language for the Analysis of regional systems",
Proceedings of the International Symposium "Simulation'75",
Acta Press, Zürich, June 1975

[20] RECHENMANN F.

"Analyse et Modélisation descendantes des systèmes socio-économiques"
Thèse Docteur-Ingénieur E.N.S.I.M.A.G. - I.N.P.G.,
Juillet 1976

[21] NILSEN R.N., KARPLUS W.J.

"Continuous-System simulation languages : A state of the art survey"
Annales de l'AICA, n° 1, Janvier 1974, pp. 17-25

[22] BRENNAN R.D., LINEBARGER R.N.

"A survey of Digital Simulation : digital analog simulator programs",
Simulation Vol. 3, N° 6, Décembre 1964, pp.22-36

[23] BRENNAN R.D.

"Continuous system modeling programs : state of the art and Prospectives for development"
in Simulation Programming Languages Buxton J.N ed.
North Holland, 1968, pp.371-396

[24] EHRMANN R.

"Les Langages de Simulation"
Dunod - Université, Paris 1971

[25] CELLIER F.E.

"Continuous-system simulation by use of digital computers : A-state-of the art survey and prospectives for development",
Simulation'75, Zürich, june 1975

[26] National Bureau of Economic Research, Inc.
"Troll : An Introduction and demonstration"
December 1972

[27] National Bureau of Economic Research, Inc.
"Troll Premier"
October 1972

[28] OUDET B.A.
"Un système interactif d'aide à la décision dans un jeu d'entreprise"
Rapport de Recherche n° 38, IMAG - Sept. 1976

[29] Control Data, Corp.
"Mimic, Digital simulation language, Ref. Manual",
Sunnyvale Cal. 1968

[30] VERDE C.R., GONZALEZ J.Ch.
"Simulation language for dynamic and continuous systems"
Simulation'75, Zürich, june 1975

[31] HAY J.L., PEARCE J.G., NAROTAM M.D.
"Simulation Language implementation on minicomputers"
Proceedings of the Symposium on Simulation Languages for Dynamic Systems - Londres, Sept. 1975, pp. 1.1-1.10

[32] JACOLIN, LE FAOU C., PIMORT, VERAN,
"IMAG II, description du programme,
rapport interne IMAG, Juillet 1970

[33] LE FAOU C., SICOT J.P.
"Un programme général de simulation des circuits électroniques : IMAG 3",
Electronique, pp.39-43, 15 octobre 1974

[34] JUTILA S.T.
"Spatial macro-economic development (SMED)"
Papers of the Regional Association, Vol. 30, 1973
pp.39-57

[35] RECHENMANN F., RIVERA E., UVIETTA P.

"Introduction de la notion d'espace dans les modèles de simulation"
Colloque sur les méthodes mathématiques appliquées à la géographie, Besançon, octobre 1975

[36] RECHENMANN F.

"A continuous simulation language for dynamic space-distributed models"
UNESCO 2nd International Seminar on Trends in Mathematical Modeling.
Jablona, Pologne, décembre 1974

[37] PUGH A.L.

"Dynamo II, User's Manual"
M.I.T. Press, 1973

[38] I.B.M.

"System/360 Continuous System Modeling Program, User's Manual",
octobre 1969

[39] Controle Data Co.

"Continuous System Simulation Language Version 3, user's guide"
Sunnyvale, Cal. 1971

[40] I.B.M.

"Continuous System Modeling Program III and graphic Feature, General Information Manual",
Sept. 1971

[41] MITCHELL E.L., GAUTHIER J.S.

"Advanced Continuous Simulation Languages",
SIMULATION, March 1976, pp.72-78

[42] WATSON H.D., GOURLAY A.R.

"Implicit integration for CSMP III and the problem of stiffness"
SIMULATION, February 1976, pp.57-61

[43] MILLER J.H., BROSILOW C.B.

"An interface to link dynamic system models"
SIMULATION, July 1976, pp.33-38

[44] GOLDEN D.G., SCHOEFFLER J.D.

"GSL - a combined continuous and discrete simulation language",
SIMULATION, January 1973, pp.1-8

[45] GOLDEN D.G., SCHEOFFLER J.D.

"Problems in the implementation of combined continuous-discrete simulation language",
SIMULATION, February 1973, pp.49-52

[46] PRITSKER A.B., KIVAT P.J.

"Simulation with GASP II"
Prentice Hall inc., 1969

[47] PRITSKER A.B., HURST N.R.

"GASP-IV, A combined continuous-discrete FORTRAN based simulation language",
SIMULATION pp.65-70, Sept. 1973

[48] RIVERA F.E.

"Review of delays in dynamic system simulation models"
Proceedings of Informatica 76,
pp.5-104, Bled, Yugoslavie, Octobre 1976

[49] KNUTH D.

"The art of computer programming",
Vol. 1, Fundamental Algorithms,
p.258, Addison Wesley, 1968

[50] RECHENMANN F.

"Equation Sorting in Multilevel Structured Models"
Proceedings of the Symposium on Simulation Languages for Dynamic Systems, AICA, Londres, 8-10
pp.18.1-18.4

[51] JACOLIN,

"Optimisation de la simulation en langage CASSANDRE",
rapport DEA Informatique - U.S.M.G., Grenoble 1974

[52] JARSCH V.

"DYNASIS-A graphic version of DYNAMO"
Simulation'75, Zürich, June 1975

[53] DUSSAUCHOY A., JOURDAN P.

"Outil Graphique d'Aide à la Conception Interactive de Modèles de Simulation de Systèmes" (OGACIMSS)
Colloque IRIA, INFORSID, pp.299, Rocquencourt, Nov. 1976

[54] FORRESTER J.W.

"Principles of Systems",
Wright-Allen Press Cambridge, March 1968

[55] LUCAS J.J., WART J.V.

"DARE-P, A portable CSSL-Type simulation language"
SIMULATION pp.17-28, january 1975

[56] GALLIGANI I., MOLTEDO L.

"An interactive system for modeling"
in Optimisation Technique, Modeling and Optimization in the service of man. Part1, pp.794-807 - Springer-Verlag 1976

[57] ROACH J.R., CHOW E.P.

"An interactive digital simulation and optimization package : FORTRAN programs MINISM and OPTSIM"
SIMULATION, Octobre 1975, pp.115-120

[58] BEHRENS J.C., SØRENSEN P.E.

"COLTS, Continuous Long-Term Simulation"
in Simulation'75, Zürich, June 1975

[59] SOHNLE R.C., TARTAR J., SAMPSON J.R.

"Requirements for interactive simulation systems"
SIMULATION, Mai 1973, pp.145-152

[60] CROSBIE R.E.

"Interaction in continuous-system simulation languages"
Proceedings of the Symposium on Simulation Languages for Dynamic Systems. Londres, Sept. 1975, pp.28.1-28.4

[61] KISHLINE P.D., BUEHRING J.S.
"WISSIM-A Simulation Command language, Part I : User's guide"
University of Wisconsin, Décembre 1971

[62] MESAROVIC M., PESTEL E.
"Strategie pour demain",
Seuil, Paris 1974

[63] MESAROVIC M., PESTEL E.
"Regionalized and Adaptative model of the global world system"
internal report to the Club of Rome.
Sept. 1973

[64] MESAROVIC D., MACKO D. TAKAHARA Y.
"Theory of Multilevel Hierarchical Systems"
Academic Press, 1970

[65] MESAROVIC M., HUGHES B., SHOOK T., PESTEL R., GILLE P., YOSHU S.
"An interactive decision stratum for the multilevel world model"
FUTURES, Vol. 5, Num 4, Août 1973, pp.357-366

[66] VAN DER HIJDEN P., et al.
"Design for a multilevel interactive softaware"
Rapport SSRG 74-7, University of Nijmegen (Hollande) 1974

[67] VAN DER HIJDEN P.
"Some remarks on the design of interactive software for the espint-energy supply planning model",
Rapport SSRG 75-06, Nijmegen University (Pays-Bas) Avril 1975

[68] VAN DER HIJDEN P.
"Interactive software for the simulation of Social Systems",
Simulation'75, Zürich, Juin 1975

[69] CLERCK B., HEREDIA J.M., JANNET M.
"Modèles de Simulation Jouée",
Rapport de 3ème année ENSIMAG, Juin 1974

[70] RECHENMANN F., RIVERA E., UVIETTA P.

"World Models : A case study on social responsability and impact",
Proceedings of the 7th IFIP Conference on Optimization Techniques. Part 1, pp.431-439 - Springer-Verlag 1976

[71] RIBEILL G.

"Stratégie pour demain ou utopie sans lendemain ?"
LA RECHERCHE, Vol. 6, n° 54, Mars 1975, pp.285-286

[72] Simulation Software Committee,

"The Sci, Continuous System Simulation Language (CSSL)"
SIMULATION, Décembre 1967, pp.281-303

[73] RECHENMANN F., RIVERA E., UVIETTA P.

"A conversational environment for the construction and use of Dynamic Models"
Proceedings of the Symposium on Simulation Languages for Dynamic Systems. Londres, Sept. 1975, pp.27.1-27.4

[74] HILBORN R.

"A control system for FORTRAN simulation programming",
SIMULATION, Mai 1973, pp.172-175

CHAPITRE II

ARCHITECTURE D'UN SYSTÈME D'ANALYSE ET SIMULATION EN SCIENCES SOCIALES

TABLE DES MATIERES

1 . DEFINITION DES OBJECTIFS ET DES FONCTIONS DU SYSTEME

1.1 ORIENTATION GENERALE EN SCIENCES SOCALES

On a donc vu au premier chapitre que la simulation en Sciences Sociales, plus particulièrement dans le domaine socio-économique, est un phénomène complexe ; l'Informatique fournit une aide à deux niveaux, en proposant un langage de description (ou de modélisation) et un système pour leur simulation.
Ces deux niveaux sont différents et cela doit être clair à tout moment de la conception et de l'utilisation d'un modèle. De multiples erreurs peuvent être évitées **si la description du modèle est indépendante de sa simulation.** En effet, on pense parfois que le modèle doit être un programme au sens classique (séquence d'instructions que l'ordinateur doit exécuter) et non une **donnée** sur laquelle opère la séquence d'instruction ; cette idée a amené à des *modèles-programmes* appelant éventuellement de *véritables* programmes.

Ces ensembles hétérogènes ont une caractéristique commune : leur plus ou moins grande *illisibilité*, et donc le manque de compréhension du système ainsi modélisé.

Dans le cadre de la Prospective en Sciences Sociales, cette notion d'intelligibilité devient un préalable indispensable à toute réalisation informatique. Le langage de description doit se rapprocher le plus possible du langage dans lequel le système-problème est formalisé. Les contraintes syntaxiques et sémantiques du langage de description doivent être des guides dans la formalisation correcte du système.

Ceci a comme conséquence une nécessaire **Indépendance du langage de modélisation vis-à-vis du système de simulation.** Un seul langage modulaire de description a été défini en respectant cette indépendance. Ce langage, (LADESH) est spécialisé dans les systèmes socio-économiques régionaux hiérarchisés [1], [2].

Cette indépendance permet notamment :

- d'utiliser le **même système de simulation pour différents langages de description.** Chaque langage de description implique une sémantique particulière à une application en Sciences Sociales.

- que les améliorations apportées au système de simulation puissent se faire indépendamment des modifications des langages de modélisation (et de leur compilateur).

- Une **conception rationnelle** du processus de simulation qui permettra de planifier les expériences d'analyse et simulation.

- une **écriture plus rapide** du modèle, car celui-ci sera débarrassé de toute notion de simulation et pourra donc refléter au mieux le système modélisé.

En effet, les langages de description devront interdire des manipulations algorithmiques ne portant pas sur la description du modèle, ce qui permet d'éviter de lamentables erreurs :

- La fonction **retard** par exemple de CSMP [3]

$$Y = \text{delay}(n, p, x) \begin{cases} y(t) = x(t-p) & \text{si } t \geq p \\ y(t) = 0 & \text{sinon} \end{cases}$$

exige la connaissance au niveau **modèle** du paramètre 'n', où n est le nombre de points que la fonction devra sauvegarder entre t-p et t. Or 'n' est fonction d'une condition de la simulation : le pas d'intégration. L'utilisateur devra avoir soin de modifier la partie du modèle définissant 'n' chaque fois qu'il change de pas d'intégration, sinon de faux résultats à l'exécution seront obtenus.

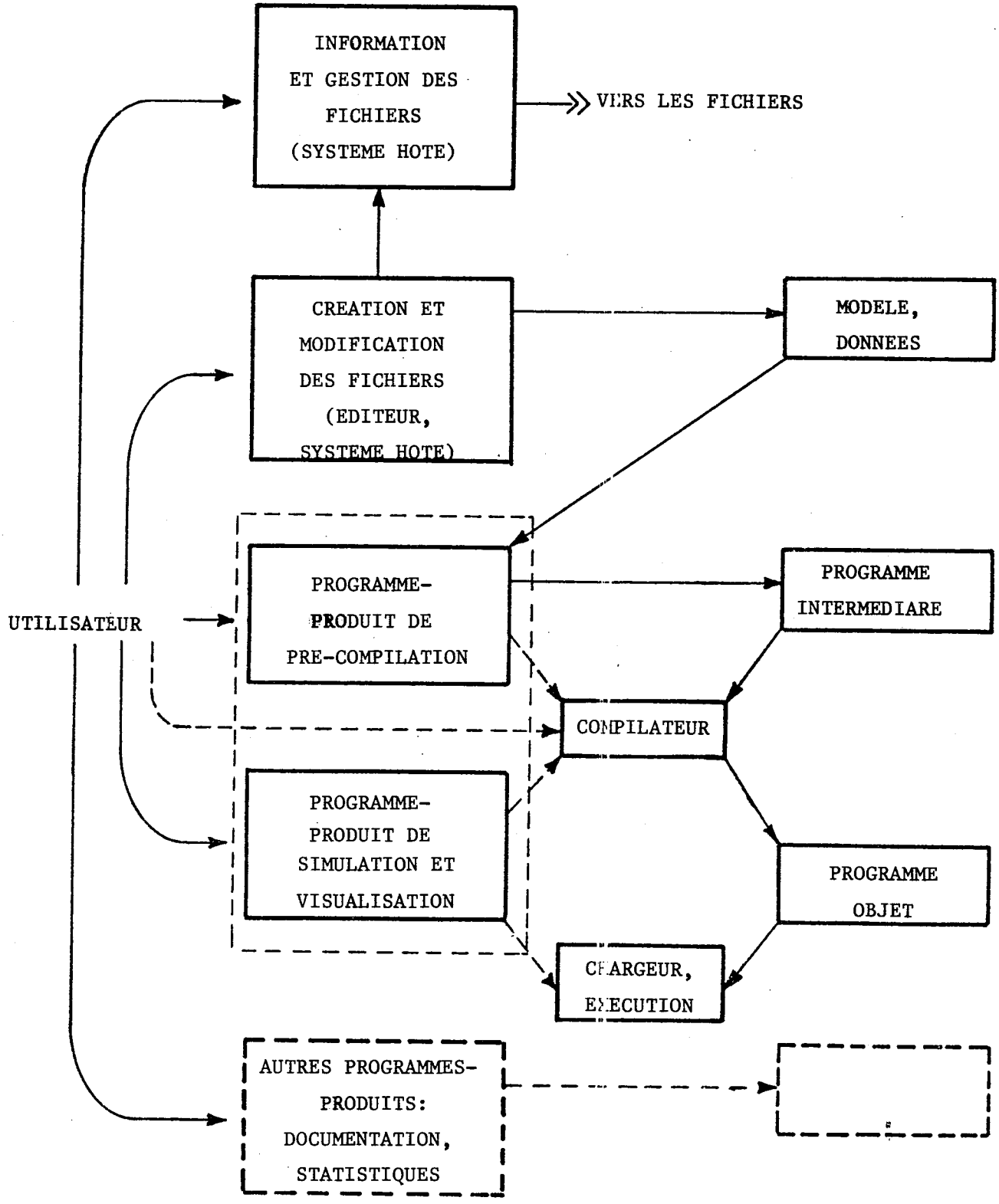
- Certaines fonctions du type événements discrets comme PULSE ou SAMPLE appartenant à deslangages de type DYNAMO [4] sont tels que leurs résultats dépendent aussi du pas d'itération choisi dans les cas où l'intervalle d'échantillonnage ne correspond pas à un nombre entier de fois le pas d'itération.

• On a constaté chez certains utilisateurs une mauvaise interprétation et donc une utilisation incorrecte du fait que, soit la méthode d'interprétation, soit le pas d'intégration, apparaissent explicitement dans des langages comme DYNAMO ; c'est ainsi que l'on peut parfois lire que les langages de type DYNAMO sont faits "pour traiter de modèles d'équations aux différences finies" [5].

La réalisation d'un système de simulation peut se faire de manières très différentes ; la façon certainement la plus simple pour un utilisateur non informaticien (l'utilisateur normal en Sciences Sociales) est de disposer d'un seul outil. Il s'avère toujours difficile de confectionner un bon programme de simulation ; on a besoin très souvent de la connaissance de plusieurs langages correspondant aux différentes interfaces entre l'homme et la machine (voir figure II.1). Cela complique la tâche déjà difficile de la simulation. C'est pourquoi nous est apparue insatisfaisante la solution de tous les systèmes de simulation continue qui est d'offrir soit des sous-programmes, soit un programme-produit (**package**). La meilleure solution est donc d'avoir une machine spécialisée en Simulation. On a donc comme seconde conséquence la nécessité d'avoir **un système informatique qui spécialise l'ordinateur en simulation.**

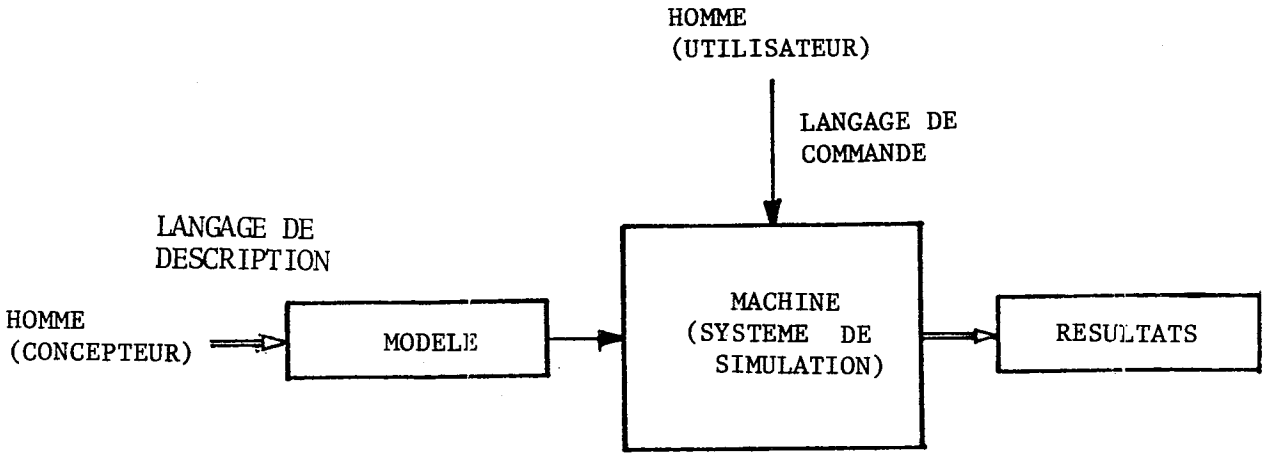
Différentes Interfaces généralement utilisées lors d'une étude en simulation avec les systèmes existants.

FIG. 2.1



La communication avec l'utilisateur est faite par un langage de commande. Le modèle est seulement une donnée sur lequel la machine travaille ; le travail consiste à interpréter et exécuter le langage de commande selon le schéma suivant :

Figure II.2



La longue phase de construction - test-analyse - mise au point des modèles, fait que les interactions homme-machine sont très fréquentes. Il est clair que toutes ces opérations forment un tout homogène et complètent le processus de simulation. L'incorporation de **facilités de dialogue homme-machine** dans le système est donc primordiale. Ces dernières ne justifient pas seulement la caractéristique **conversationnelle** d'un système de simulation mais en font un des atouts majeurs.

Le système de simulation doit être **simple à utiliser** ; ce qui impose un langage de commande de haut niveau, un nom pour chaque commande qui soit significatif pour l'utilisateur analyste de systèmes, un nombre réduit de commandes, une longueur limite de ces mêmes commandes, etc.

La **Clarté** implique que les étapes du travail d'un utilisateur soient bien définies. Le système de simulation fournit à l'utilisateur des repérages pour savoir à tout moment à quel stade il en est, et ce qu'il a déjà fait. Ce **guidage** constitue une véritable étape d'apprentissage.

La **Modularité** est importante, car le système doit être facilement modifiable.

La **Transportabilité** est un besoin particulièrement ressenti en Sciences Sociales, car aussi bien la description du modèle que les expériences dynamiques réalisées (simulation, analyse), doivent être répétables ailleurs pour pouvoir être l'objet d'une critique incessante. Il est donc très souhaitable que le système de simulation aussi bien que le langage de modélisation puissent être mis encore sur la plupart des matériels informatiques de taille comparable.

Enfin, et ce dernier point n'est pas le moindre, une économie considérable de temps machine peut être obtenue si **l'exploitation de résultats d'une simulation est indépendante de la description du modèle et de la simulation** (par exemple le choix d'une bonne échelle est difficile à prévoir d'avance).

1.2 DEFINITION DES FONCTIONS DU SYSTEME

Nous allons définir l'organisation d'un système informatique qui réponde aux orientations exposées.

La séparation "modélisation/simulation" apporte tout naturellement une séparation "description compilable/commande interprétable".

En effet, le système de simulation (par opposition aux grandes lignes de CSSL) doit permettre de faire **toutes** les opérations (ne concernant pas la description) qui sont nécessaires à une étude en simulation ; celles-ci devraient pouvoir être faites de façon **exclusivement conversationnelle.**

Une telle séparation a comme corollaire la séparation de deux entités complémentaires : le compilateur associé au langage de modélisation et l'interpréteur du langage de commande associé au système de simulation ; on n'insistera pas davantage sur les avantages de cette division.

Dans l'alternative proposée *"général/spécialisé"* nous avons choisi le deuxième terme mais au sein d'une classe suffisamment grande de modèles.

Le choix *"problème orienté"* qui a été fait dans un objectif de simplicité et clarté exposé précédemment, nous guidera pour concevoir un système dépourvu en particulier, de notions technologiques, pour rendre transparent à l'utilisateur des concepts comme "FICHIER", "COMPILATION", "CHARGEMENT", etc.

D'autres options confrontées aux objectifs de l'application perdent à nos yeux leur intérêt immédiat ; c'est le cas de la segmentation, car l'intégration automatique du processus de simulation dans d'autres processus rendrait délicate la maîtrise absolue du modèle, de son comportement et sa signification.

Les **choix d'algorithmes d'intégration**, de mécanismes **d'extension** et d'**analyse**, etc... ont été faits en fonction de notre expérience dans le domaine.

L'utilisation simultanée du système de simulation par plusieurs personnes agissant sur un même modèle peut être utile à des fins pédagogiques, ou pour tester des stratégies à plusieurs joueurs ; nous nous sommes volontairement restreints au cas le plus courant du concepteur qui cherche à étudier et à comprendre la dynamique de son modèle. Un système **mono-utilisateur** est, en l'état des connaissances, suffisant, d'autant que les systèmes d'exploitation évolués permettent de mettre à disposition des utilisateurs des copies d'un système ainsi, que d'établir la communication entre les utilisateurs.

Une étude en simulation ne doit pas se limiter à une série de simulations à un changement près des valeurs initiales (comme dans le rapport CSSL [6]), mais elle doit laisser la souplesse d'autres types de changement qui seront justifiés par la suite. Doivent pouvoir être modifiés :

- tout paramètre numérique (i.e. test de sensibilité)
- une partie de la description (voir paragraphe 5 "Création de nouveaux objets par les scénarios").

2 . SUPPORT LOGICIEL

Pour acquérir une expérience dans le domaine, établir un cahier de charges, connaître et évaluer les problèmes de la mise en oeuvre d'un système de simulation, nous avons défini un premier prototype de système de simulation [7].

Il est conversationnel et mis en oeuvre avec le mécanisme et langage d'extension EXEC du système d'exploitation CP-CMS (IBM/360) [8], [9].

Nous n'allons pas décrire ce premier essai, mais seulement signaler certaines conclusions qui nous ont permis en outre de faire le choix d'un système support de logiciel convenable :

- Les contraintes syntaxiques et lexicographiques sont parfois très gênantes ; par exemple : exclusion de certains caractères, utilisation obligatoire d'un certain séparateur, interdiction de certains types de paramètres, (comme les constantes réelles), nombre de restreint et longueur limitée des paramètres, etc.

- L'analyseur syntaxique doit être indépendant de la sémantique du langage de description et aussi souple que possible car il est à prévoir de multiples changements dûs à l'évolution et à la mise en cause fréquente de la grammaire.

- L'écriture des fonctions d'un système informatique est extrêmement compliquée par la non-existance, par exemple, d'opérateurs de multiplication ou de gestion de tableaux.

Le langage EXEC est bien adapté pour créer des commandes supplémentaires du système CP-CMS, mais pour la réalisation d'un système propre, il se révèle rudimentaire et lourd. Tous ces inconvénients ont fait que ce premier système a été abandonné.

Il faudrait pouvoir choisir un langage puissant et surtout facilement transportable pour décrire et programmer le système. Notre choix s'est porté sur FORTRAN IV parce que malgré ses limitations propres (allocation statique de la mémoire, absence de récursivité [13], manque de primitives pour la programmation structurée [14], etc...), il est commode [10], très répandu, permet une compilation séparée et rapide, mais surtout il est facilement transportable.

FORTRAN IV n'est pas unique, chaque constructeur a sa propre version [11] ; la transportabilité impose de se restreindre au noyau minimal défini par les normes internationales ISO [12].

Un des avantages de FORTRAN IV est qu'il permet facilement une programmation type modulaire [15], [16], [17]. Cela permet en outre de fournir un ensemble de programmes test de modules réalisés lors de la construction, et indispensable à la maintenance d'un produit logiciel.

Faisons la distinction entre le **langage intermédiaire**, le **langage associé** et le **langage support**. Le langage intermédiaire est le langage dans lequel le modèle est traduit grâce au pré-compilateur. Le langage associé est le langage avec lequel on peut décrire des extensions soit du langage de description, soit du système de simulation ; (comme c'est le cas des 'macros'). Le langage support est le langage qui sert à décrire l'application ; ainsi le système de simulation peut être décrit dans un langage support différent du langage support du pré-compilateur.

On n'a pas incorporé la possibilité d'un langage associé au système de simulation, car les extensions doivent être faites avec circonspection en assurant toujours une parfaite cohérence avec la partie existante.

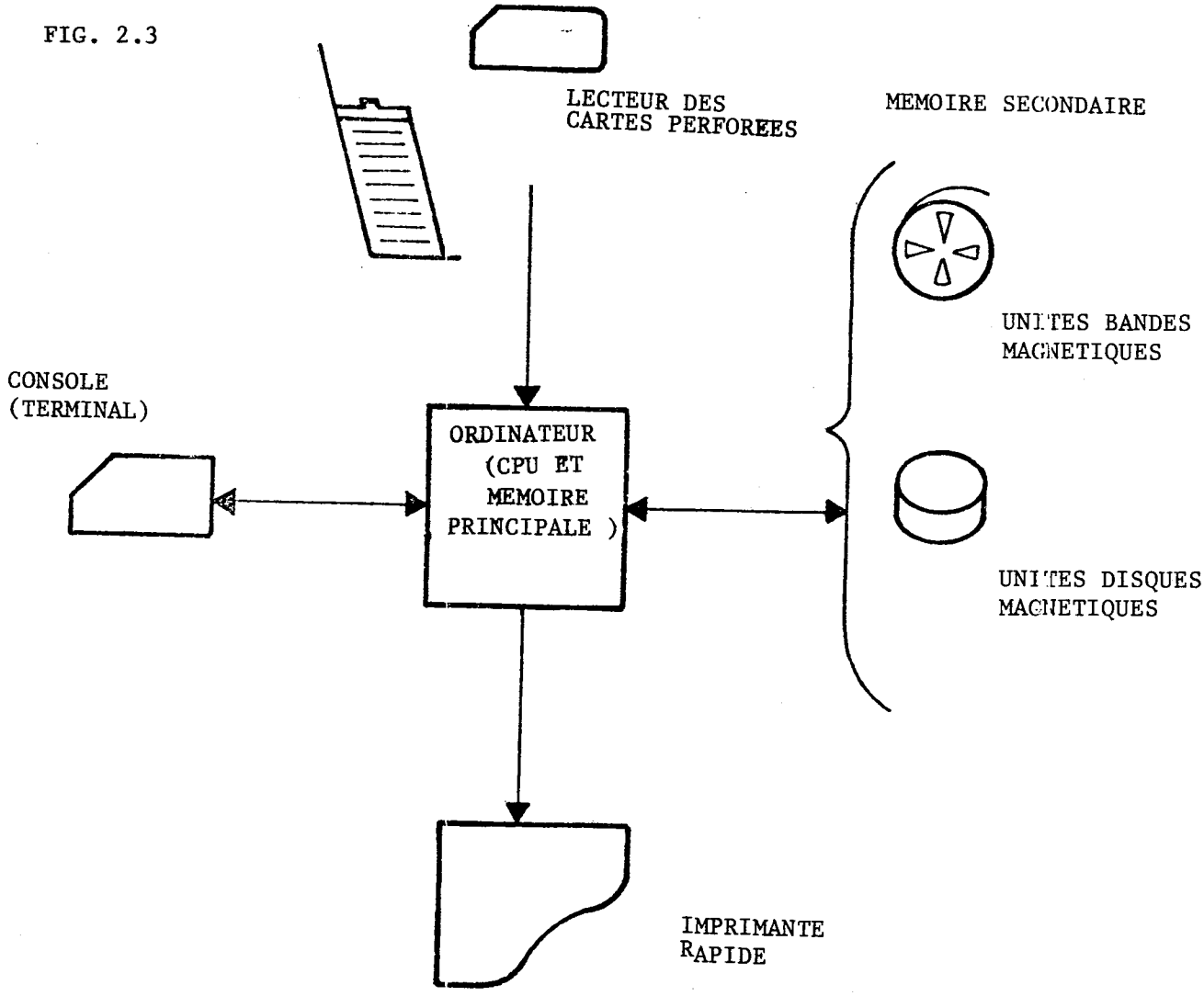
Le langage intermédiaire est lié à une pré-compilation et n'existerait pas s'il y avait un vrai compilateur qui génère du code (translatable) exécutable. En conséquence, le système de simulation ne doit pas permettre l'introduction de manipulations directes du modèle après la phase de pré-compilation (ou de compilation).

3 . SUPPORT ET CONFIGURATION MATERIELS

Un système informatique dit *évolué* ne doit pas incorporer dans son langage des notions liées à la technologie, mais ceci ne veut pas dire que l'on ignore les contraintes techniques de fonctionnement dûes principalement au matériel et à la façon dont il est interconnecté.

La configuration minimale du système conversationnel est classique comme le montre la figure II.3. La mémoire secondaire est néanmoins utilisée exclusivement en accès séquentiel du fait des contraintes de FORTRAN IV (ISO). L'entrée par lecteur de cartes perforées permet la création rapide des grands objets (par exemple un modèle). Les sorties par imprimante rapide, quoique non indispensable, sont fort appréciées des utilisateurs.

FIG. 2.3



C O N F I G U R A T I O N M A T E R I E L L E

M I N I M A L E S O U H A I T A B L E

4 . LES PRINCIPAUX OBJETS MANIPULES PAR LE SYSTEME DE SIMULATION

On va définir un **objet de simulation** comme un composant logiciel de base sur lequel le processus de simulation est effectué. Il y a parmi les objets principaux de simulation : le **modèle** et l'ensemble des paramètres numériques (appelé dorénavant **jeux de données**).

On peut définir d'autres **objets** comme **entités d'informations externes ou internes au système, qui transitent par lui et dont la durée de vie peut aller au delà d'une session.**

L'objet **produit d'une simulation** (ensemble de résultats numériques d'une simulation) qui reste transparent à l'utilisateur mais dont on doit prendre en compte l'existence en vue de la construction des **objets de visualisation** (liste d'impression numérique de résultats, graphes d'évolution, espace-phase, histogrammes, etc...).

Les objets portant l'information relative au modèle seront appelés génériquement **documentation.** Ils sont composés de l'identification et de la définition de chacune des variables intervenant dans la description d'un modèle, de ses unités et parfois aussi d'une description de la structure hiérarchique d'un modèle.

Le système de simulation aidant à la construction d'un modèle modulaire, doit permettre d'incorporer un module à la description d'un modèle, et de créer ou détruire ces objects **modules** contenus dans une bibliographie. On ne se soucie pas du contenu sémantique de ces modules ; ainsi on peut par exemple, dans le langage de modélisation LADESH faire correspondre ces modules aux **secteurs** [2].

On a évoqué au chapitre II - § 4, la communication unidirectionnelle au travers des cartes perforées : l'objet **lot** est un ensemble de cartes perforées (ou images de cartes perforées) créées par l'utilisateur et qui permettront au système de simulation de créer de nouveaux objets (modèle, jeux de données, documentation, etc...) ou rajouter un morceau à des objets déjà créés. Le contenu et la forme des objets-lots seront décrits au chapitre suivant.

Il existe aussi des objets internes créés par le système pour sa propre gestion (i.e. descripteurs) et indispensables à la bonne marche du système, mais qui n'ont aucun intérêt pour l'utilisateur. Enfin pour l'information en général il existe des objets dont le plus important est **l'état du système** ; il s'agit d'un objet qui contient les informations sur l'existence des principaux objets qui ne sont pas générés par le système (noms et nombre de modèles déjà créés, etc...).

Un système d'extension (s'il y en a un) du langage de commande n'est pas en soi un objet indépendant ; de même que l'analyseur syntaxique, il fait partie du système; par contre, une macro (la définition d'une macro) est un objet.

La figure II.4 montre un tableau récapitulatif des principaux objets.

Classification des principaux objets manipulés par le système.

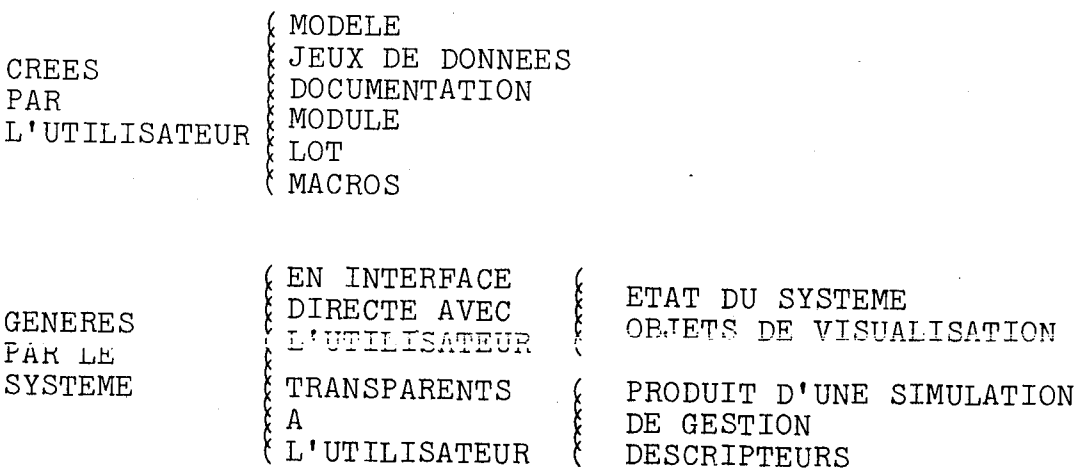


Figure II.4

On pourrait dire que les objets que manipule un système habillent le système *nu*. Dans un système qui n'a jamais été utilisé, il n'y a pas d'objets créés par l'utilisateur et les objets générés par le système sont vides. Les objets (qui sont donc particuliers à chaque utilisateur) forment le contexte du système et le mettent *en situation*.

5 . CREATION DE NOUVEAUX OBJETS PAR LES SCENARIOS

Une fois que le modèle du système social est testé (par exemple à travers un certain nombre de simulation), on est parfois amené à étudier la dynamique du système sous des hypothèses différentes de celles où on a réalisé le modèle fondamental ; dans ce cas on construit des scénarios qui seront étudiés et testés grâce à la Simulation.

Un Scénario est une description d'une séquence des états du système impliqués pour un ensemble d'hypothèses. Il est donc simulé par un modèle, mais un modèle dans une situation qui évolue (même structurellement). Au scénario est toujours associé un jeu de données.

Donc, avec le modèle de base du système analysé on peut générer différents scénarios si l'on change le jeu de données. Mais ceci n'est pas suffisant car les hypothèses quel'on peut faire ne sont pas seulement quantitatives et statiques (par exemple : changer la valeur du taux de croissance de la population) mais le plus souvent ce sont des hypothèses qualitatives et dynamiques (ou même structurelles). En effet, dans les relations qui constituent le modèle, il y a déjà des hypothèses de fonctionnement du système qu'on veut tester dans le cas général (par exemple changer le type de croissance de la population soit par l'introduction d'autres facteurs, soit par la forme de l'équation elle même, etc...).

Le fait que l'on ait deux scénarios d'un même système ne veut pas dire que l'un soit meilleur que l'autre. Par conséquent, il est souhaitable de les conserver et même de les comparer. En fait les modèles de deux scénarios d'un même système, sont-ils structurellement différents ? Cela dépend du type d'hypothèses ; en tout cas le système social restant le même, la plupart de ses relations, ainsi que la structure générale doivent être en principe les mêmes.

Il existe toujours un scénario **privilégié** de base qui **reflète** le système étudié ; par rapport à ce scénario de base, on peut classer les autres scénarios du point de vue des types fondamentaux de changements des hypothèses à effectuer, à partir du scénario de base pour construire d'autres scénarios :

- Le **scénario tendanciellement distinct,** est celui obtenu en gardant la structure du scénario de base (description) mais en faisant varier les intensités d'un ensemble de tendances ce qui se traduit par la modification de l'ensemble des valeurs numériques correspondantes.

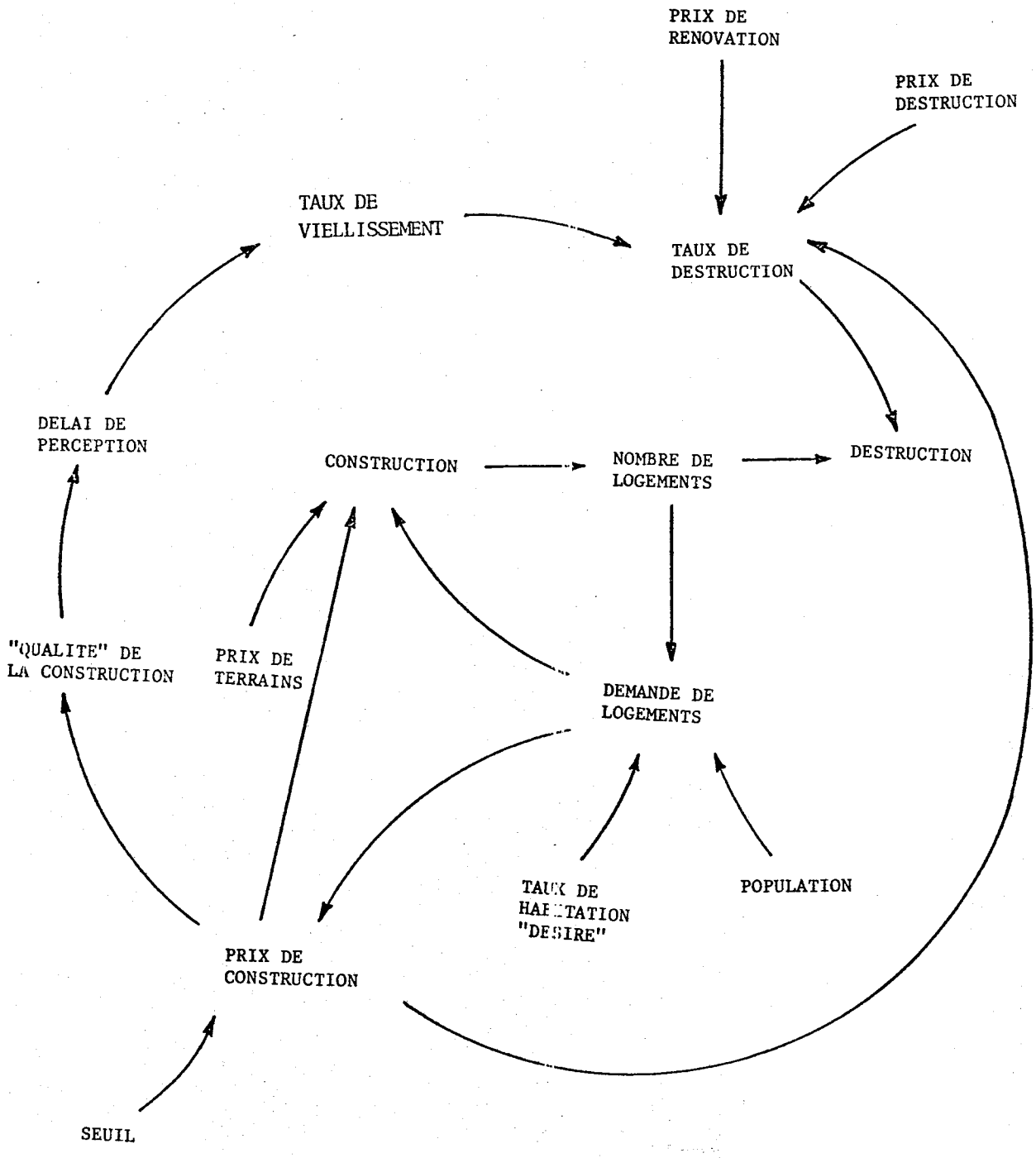
- Le **scénario tendanciellement opposé,** est comme son nom l'indique, obtenu en inversant une tendance, par exemple une croissance au lieu d'une décroissance. Ce changement est habituellement obtenu en modifiant la ou les relations correspondantes dans la description et éventuellement certaines valeurs numériques associées.

- Un **scénario structurellement distinct,** est un scénario qui se distingue du scénario de base par l'inclusion ou la suppression d'éléments structurels. Ceci est reflété par l'apparition ou la disparition dans le modèle des relations (équations) correspondantes ; cela implique une transformation **des relations où l'élément** en question intervient.

L'étude hypothétique du processus de construction/destruction dans le système du logement urbain, illustre ces différents types de scénarios.

Le scénario de base (voir figure II.5) montre que le système de logement est régulé de façon interne par la demande de logements, par le prix de construction et par le taux de destruction de logements. On peut espérer que le nombre de logements construits subira à moyen terme les mêmes variations que la population (variable exogène). Pourtant à long terme une relance de la construction peut se produire par le renouvellement plus ou moins rapide suivant la qualité de la construction.

Figure II.5



SCENARIO 1 (DE BASE)

Le scénario 2 (voir figure II.6) est du type **structurellement distinct.** En effet, comme on peut le constater sur la figure, on a introduit un nouvel élément structurel de description : la *spéculation foncière* qui est un facteur de régulation supplémentaire. Une nouvelle relations sera créée pour la définir, et la relation *prix des terrains* sera transformée en fonction de la spéculation. On peut espérer dans le comportement dynamique une stagnation à moyen terme de la construction des logements. Le niveau de stabilité du ratio construction/destruction sera vraisemblablement fonction du montant du prix (construction et terrain) et de la *voracité* des spéculateurs. Le fossé existant entre le taux réel d'habitation (concentration dans le même logement) et le taux *désiré* risque de se creuser.

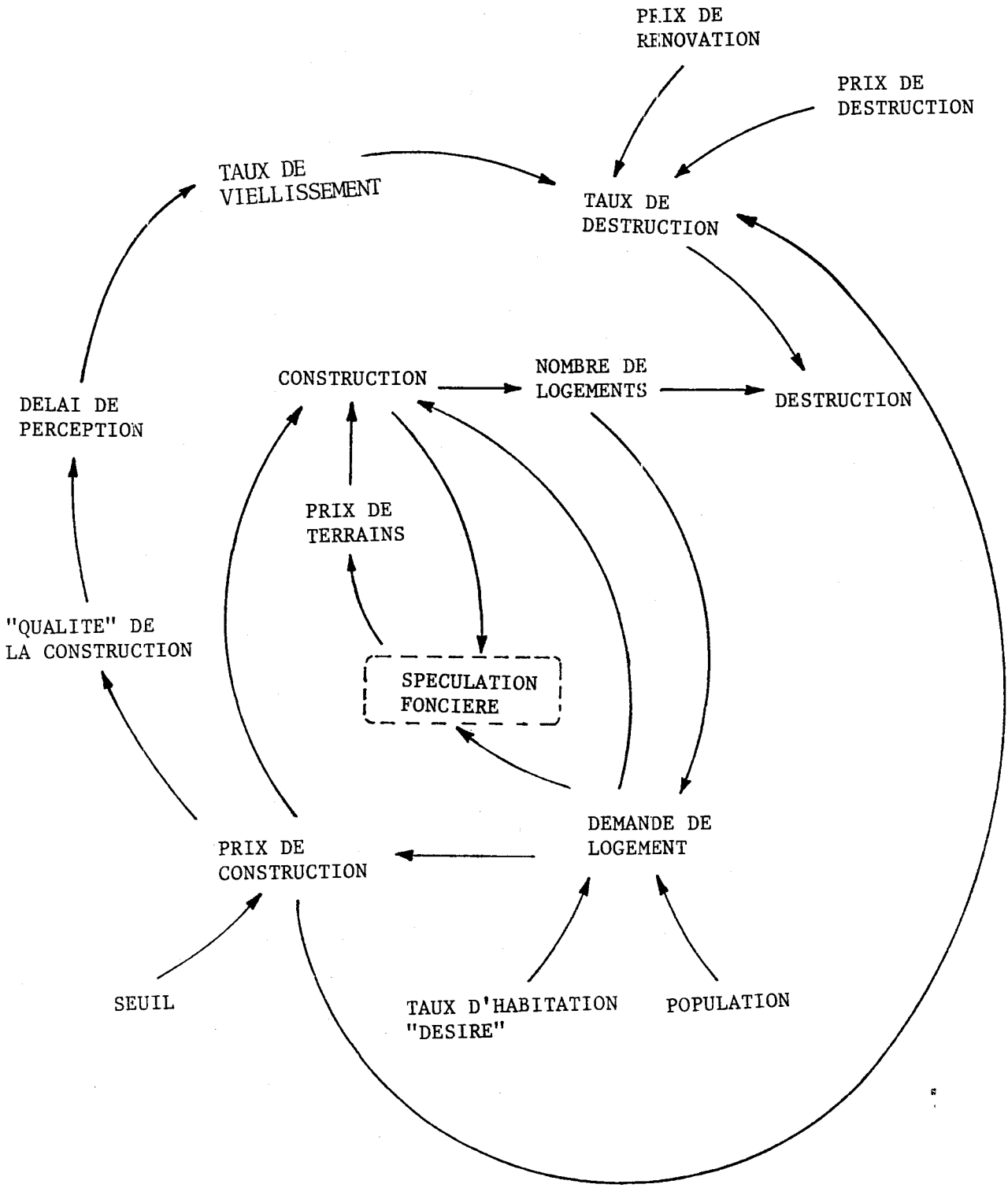
Le scénario 3 est du type tendanciellement opposé. Il a donc la même structure que le premier scénario et n'en diffère que par le fait que le prix de la construction n'intervient plus comme élément de régulation *positif* du système (a) mais comme élément *négatif* (b) dû a une autre forme de spéculation.

(a) demande de logement $\xrightarrow[-\quad -]{+\ \oplus\ +}$ Prix de la construction $\xrightarrow[-\quad -]{+\ \oplus\ +}$ Construction

(b) demande de logement $\xrightarrow[-\quad +]{+\ \ominus\ -}$ Prix de la construction $\xrightarrow[-\quad -]{+\ \oplus\ +}$ Construction

Cette hypothèse peut surement "bloquer" la destruction, car en augmentant les prix de la construction on peut espérer que d'une part le taux de vieillissement diminuera et que d'autre part, les prix de rénovation des logements deviendront plus compétitifs que ceux de la destruction et construction réunis.

Figure II.6


S C E N A R I O 2
PRIX DE RENOVATION
PRIX DE DESTRUCTION
TAUX DE VIELLISSEMENT
TAUX DE DESTRUCTION
CONSTRUCTION
NOMBRE DE LOGEMENTS
DESTRUCTION
DELAI DE PERCEPTION
PRIX DE TERRAINS
"QUALITE" DE LA CONSTRUCTION
SPECULATION FONCIERE
DEMANDE DE LOGEMENT
PRIX DE CONSTRUCTION
SEUIL
TAUX D'HABITATION "DESIRE"
POPULATION

Le Scénario 4 est du type tendanciellement distinct. En conséquence, il aura la même structure que le scénario de base et la différence, quantitative, est introduite au niveau de la *qualité de la construction*. En effet on peut espérer que grâce aux nouvelles techniques du bâtiment le rapport qualité/prix de construction s'améliorera. Le comportement dynamique du modèle sera sensiblement le même que pour le scénario de base à moyen terme, mais il se peut qu'à long terme la construction dépende quasi-exclusivement de l'augmentation de la population, car la qualité de construction allongera la vie des logements.

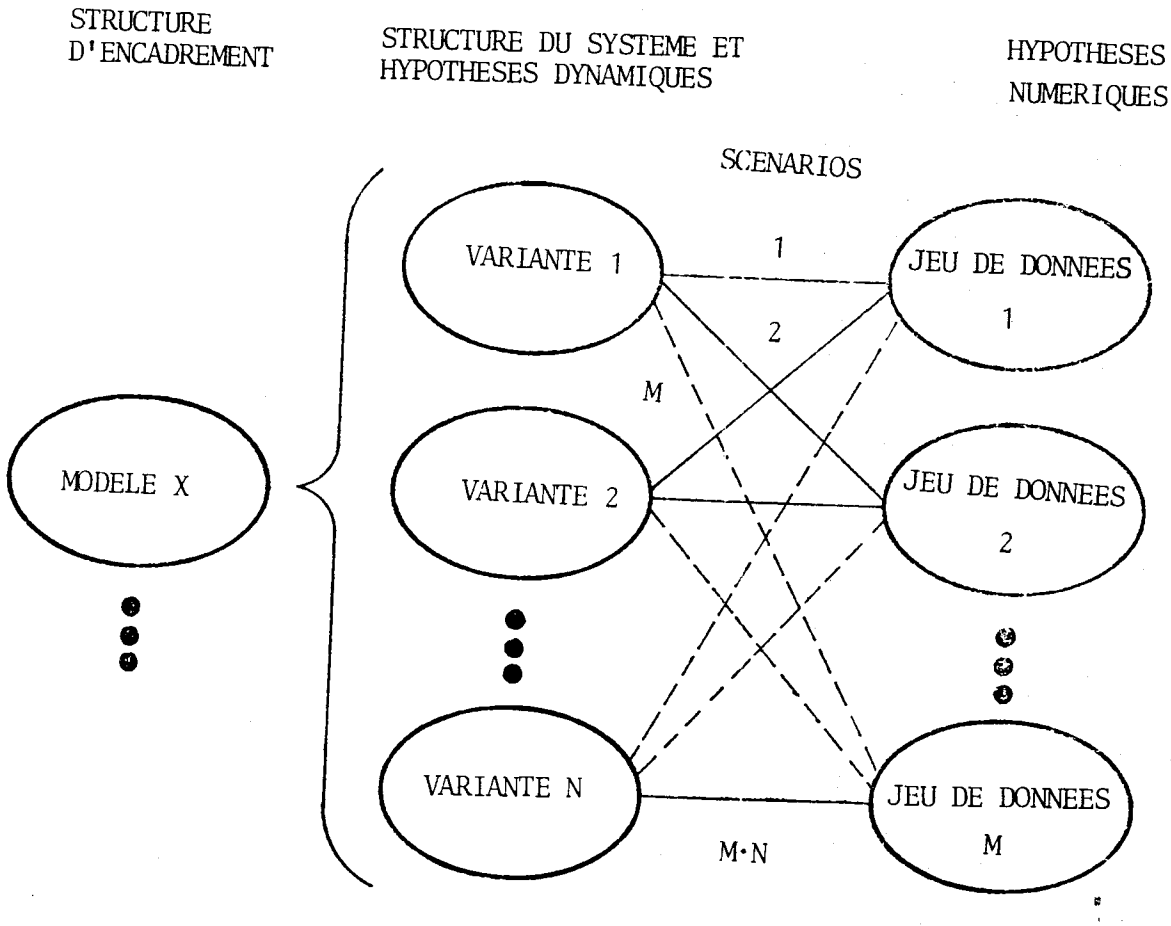
La simulation peut jouer un rôle important, car en explicitant toutes les relations et en les **expérimentant**, on peut mieux comprendre les *mécanismes* du système du logement et elle confirmera ou non la cohérence des scénarios compte-tenu des hypothèses formulées.

Ceci peut mettre en évidence certaines **hypothèses implicites** sur lesquelles il n'est pas opportun de s'étendre ici ; elles peuvent cependant révéler des faiblesses dans l'analyse. Citons par exemple le fait, optimiste, qu'en augmentant les prix de la construction on induit nécessairement une amélioration de la qualité de la construction. Ces considérations peuvent nous amener à reformuler le système modélisé.

L'informatique peut donc fournir une aide, si l'on évite à l'utilisateur de réécrire tout le modèle juste pour faire quelques changements. Le système en vue de la création de scénarios fournira à l'utilisateur des copies de son modèle ainsi que les outils nécessaires aux modifications. Ces copies (qui peuvent être des copies virtuelles, voir chapitre suivant) sont de nouveaux objets, appelés **VARIANTES**. Le modèle de base, comportant aussi des hypothèses, est la variante N° 1 à partir de laquelle on fait toutes les autres variantes [18].

La combinaison lors de la simulation, d'un jeu de données et d'une variante fait un scénario. Si chaque variante peut avoir un ensemble de jeux de données, on obtient ainsi une grande richesse de possibilités de scénarios. Le jeu de données, comme partie des hypothèses est partageable entre toutes les variantes d'un même modèle. Si on a n variantes et m jeux de données on pourra avoir m.n scénarios différents. Voir figure II.7.

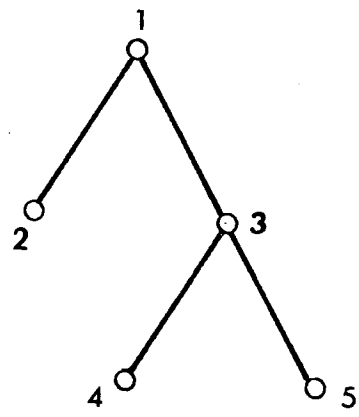
Figure II.7



Le nom d'un modèle doit donc être vu comme la désignation d'une classe qui englobe un certain nombre de variantes et c'est seulement par commodité que la description du modèle (structure fondamentale du système social) est la variante 1. Dans ce sens, deux modèles différents auront évidemment deux ensembles de jeux de données disjoints et non partageables.

Il existe plusieurs façons de créer des variantes dont la plus souple consiste en une modification soit sur le modèle (variante 1), soit sur une autre variante. Ceci implique une structure arborescente (voir figure II.8) qui entraîne des conséquences difficiles à prendre en compte même par un utilisateur averti (par exemple la modification d'une variante peut entraîner la modification de ses successeurs ; de même la destruction d'une variante implique la disparition des variantes successeurs).

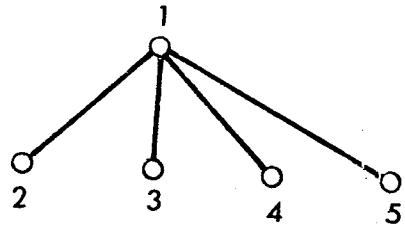
Figure II.8



C R E A T I O N A R B O R E S C E N T E

Mises à part les difficultés techniques de la mise en oeuvre de la structure arborescente (qui ont été surmontées), cette structure a été pour l'instant abandonnée car en analyse de systèmes on fait le plus souvent un seul scénario d'encadrement qui donne naissance à plusieurs scénarios tendanciels. La solution retenue est donc plus simple ; elle consiste à la création directe des variantes comme modifications d'une variante privilégiée (modèle ou variante 1). (Voir figure II.9).

Figure II.9



C R E A T I O N D I R E C T E

On pouvait penser que le même argument qui a joué pour la création des objets variantes à partir d'une variante privilégiée, joue aussi pour la création de jeux de données. Or ne c'est pas le cas, car le jeu de données a des fonctions multiples :

- Faire des études de validité, stabilité, comparaison, ajustements de paramètres.
- Faire des tests de sensibilité sur telle ou telle variable lors de la construction du modèle (variante 1).
- Faire des hypothèses quantitatives en vue de la création de nouveaux scénarios.

Avec les moyens traditionnels des langages de simulation exposés au chapitre I, on peut seulement changer les valeurs initiales d'un modèle entre une simulation et une autre ; dans des cas exceptionnels, on peut aussi changer toutes les constantes numériques. En faisant cela, on mélange tous les scénarios dans le même modèle, ce qui n'aide guère à la compréhension. Enfin, d'autres systèmes font le branchement entre un scénario et un autre par une intervention externe en cours de simulation, selon les résultats obtenus. La construction d'un scénario devrait précéder la modélisation. La simulation servant à une confrontation *hypothèses-résultats* typique de l'analyse de systèmes.

6 . ARCHITECTURE VISIBLE PAR L'UTILISATEUR

Etant donné que le but d'un système de simulation est de fournir tout type d'aides informatiques au développement de la simulation, il faut que l'organisation du système, vue de l'utilisateur, soit un guide et suive les étapes de la simulation. Par contre, l'organisation interne d'un système doit en assurer la gestion, en employant de la meilleure façon ses propres ressources. Et cette gestion des objets doit être transparente à l'utilisateur.

Il ne faut lui laisser qu'une partie minimale des spécifications des objets, de leur génération et des transformations qu'il doit effectuer.

Le dialogue homme-machine est établi à travers certains périphériques spécifiés dans les ressources matérielles : terminal, lecteur de cartes, imprimante rapide. Voir figure II.10.

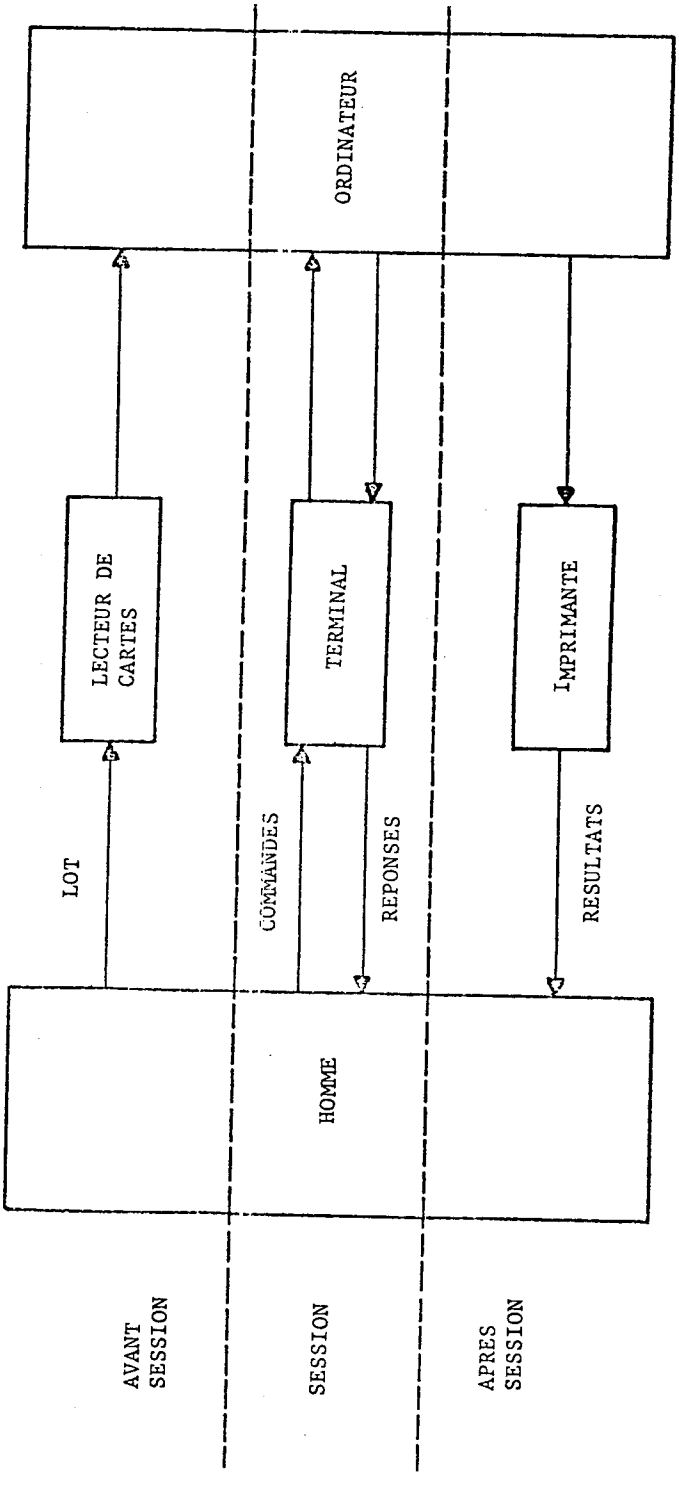


Figure II.10

Le système de simulation peut donc être vu comme une manipulation (au sens large) des objets : création, transfert, destruction, transformation. Voir figure II.11. Mais les opérations qui se font sur les objets sont parfois très compliquées à décrire sous forme de schémas montrant seulement des manipulations simples des objets ; d'autre part il y a des objets qui reçoivent de l'information (transfert) de la quasi totalité des autres (par exemple l'état du système) ; il y en a d'autres (comme une macro) qui, pratiquement, n'inter-agissant avec aucun objet mais directement avec le système.

L'architecture apparente doit prendre en compte ces flux entre les objets, de façon à trouver des chemins informatiques, étant bien entendu que c'est le langage de commande qui permet ces flux.

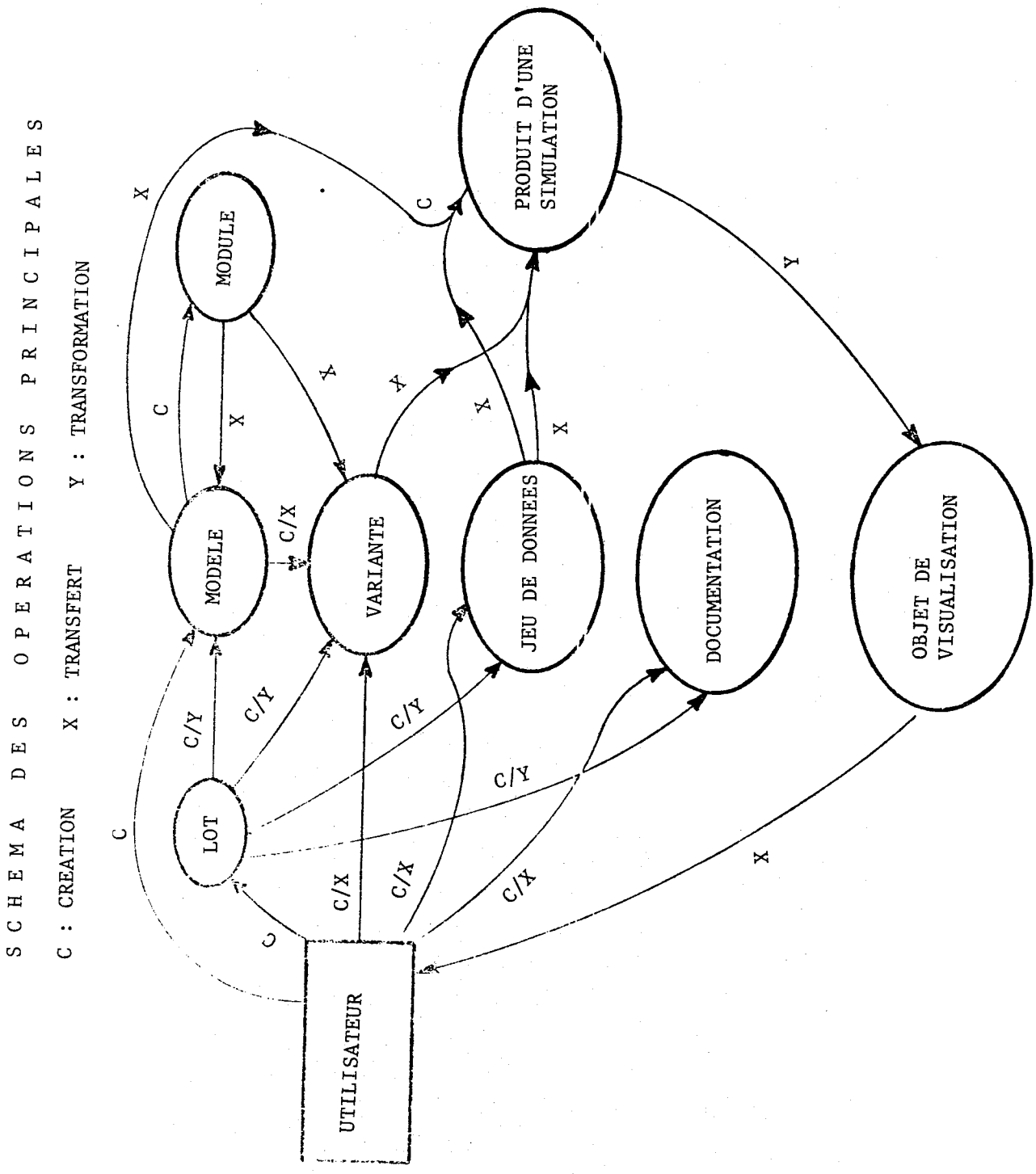
Un certain guide doit être donné à l'utilisateur pour utiliser facilement ce langage sans tomber dans le dialogue rigide (même structuré) de question-réponse [19], l'initiative du dialogue, et donc un degré assez large de liberté, doit être laissé à l'homme).

Une façon d'assister l'utilisateur est de regrouper les commandes dans des entités logiques appelés ENVIRONNEMENTS.
Un environnement est défini comme l'ensemble de commandes associé à un certain type d'opérations sur un ensemble réduit d'objets ; le découpage est donc à la fois logique (type d'opérations) et physique (ensemble d'objets).

Ce regroupement de commandes en environnements à été conçu de façon à :

1/ Respecter les grandes étapes du processus d'analyse et simulation, et
2/ Eviter certaines erreurs qui ne sont pas syntaxiques, mais dûes à la séquence dans laquelle les commandes peuvent être émises (par exemple le lancement de la simulation, sans avoir vérifié donc compilé, le modèle et le jeu de données).
En conséquence, il doit exister des contraintes pour le passage d'un environnement à un autre.

Figure II.11


SCHEMA DES OPERATIONS PRINCIPALES
C : CREATION X : TRANSFERT Y : TRANSFORMATION
PRODUIT D'UNE SIMULATION
MODULE
MODELE
VARIANTE
JEU DE DONNEES
DOCUMENTATION
OBJET DE VISUALISATION
LOT
UTILISATEUR
C
X
Y
C/X
C/Y

Les environnements retenus sont :

1/ Entrée
2/ Information
3/ Construction
4/ Mise à jour
5/ Vérification
6/ Simulation
7/ Visualisation.

• L'environnement d'**Entrée** est l'environnement initial où le système est mis à la disposition de l'utilisateur ; les fonctions de cet environnement sont de donner à l'utilisateur des renseignements très généraux sur l'état du système ainsi que de définir le contexte de travail : le modèle sur lequel on va travailler.

• L'environnement d'**Information** permet à l'utilisateur de se renseigner sur l'état du système, les objets existants dans le système : son contenu et sa sémantique (documentation !), certaines opérations de sauvegarde ou de destruction des objets seront incorporées de façon à agir, une fois connue, l'existence des objets ; en conséquence, il convient qu'il soit également, l'environnement de sortie du système.

• L'environnement de **Construction**, permet comme son nom l'indique, de construire certains objets, à savoir : le modèle, une variante, un jeu de données, la documantation, et des modules de la bibliothèque. La construction s'effectuant par blocs, elle se sert pour réaliser ces derniers objets, de l'objet "lot" (images des cartes perforées lues auparavant) et des objets de la bibliothèque : les modules.

• L'environnement de **Mise à jour** permet de faire des modifications (qui peuvent aller de la création, à la destruction, en passant par la transformation) des objets suivants : le modèle, une variante, un jeu de données, ou la documentation.

Cet environnement pour son usage propre (vue la grande probabilité de répétition de la même modification), permet de définir (créer, modifier et détruire) et utiliser des objets appelés **macros** qui permettent à l'utilisateur de faire des modifications plus aisées.

- L'environnement de **Vérification**, permet d'indiquer à l'utilisateur si son modèle, variante, et jeu de données sont prêts à être simulés, c'est-à-dire à indiquer si ses objets sont complets et s'ils sont exempts d'erreurs syntaxiques ; ces opérations que l'informaticien reconnaîtra font partie effectivement d'un processus de pré-compilation ou de compilation.

- L'environnement de **Simulation** sert à fournir tous les renseignements et spécifications d'une simulation (par exemple méthode d'intégration, demande de calcul de l'erreur, pas minimal et maximal de calcul, etc), ainsi que le lancement effectif de la simulation.

- L'environnement de **Visualisation** permet d'effectuer la sortie des résultats numériques et graphiques d'une simulation sous différentes formes (évolution temporelle, espace bidimensionnel, etc.) avec différentes spécifications (combinaisons d'échelles, effets zoomz, etc.), d'effectuer certains calculs comme la moyenne temporelle d'une variable, et finalement de faire des comparaisons, par exemple, entre les résultats des différentes simulations avec des moyens graphiques (par exemple histogramme).

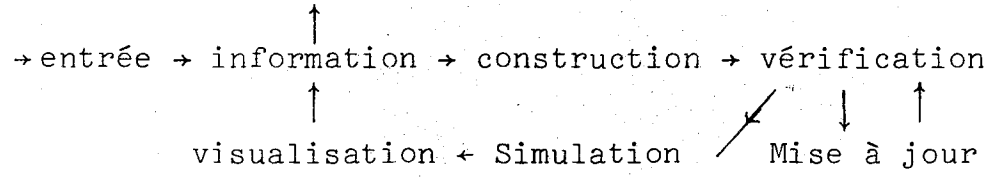
C'est grâce à l'existence de l'objet **produit d'une simulation** (objet absent dans les systèmes CSSL), que la visualisation des résultats pourra se faire *sur mesure* et en fonction du besoin d'interprétation des résultats. D'autre part des fonctions qui n'existent pas dans les autres systèmes comme la **comparaison** entre deux simulations ou le cadrage par **Zoom** pourront être effectuées.

Il y a donc **deux types de commandes : Interne et de Transfert.** Les commandes internes servent à effectuer les opérations propres à l'environnement et les commandes de transfert permettent de changer d'environnement.

Il n'existe pas un ordre imposé pour effectuer les commandes internes à un environnement, mais cela ne veut pas dire que l'ordre soit sans importance ; s'il y a deux commandes internes A et B, si B a besoin de renseignements de A, et si B est demandée avant A, une option de A est prise par défaut.

Les environnements avec les commandes de transfert forment un réseau [20] ou graphe orienté où les *noeuds* sont les environnements et les commandes de transfert les *branches normalement orientées*. Voir figure II.12. Donc l'ordre de passage entre deux environnements est limité par ces branches.

Il y aura donc des *chemins* très parcourus dans ce graphe, car ils correspondent au processus normal d'utilisation, comme par exemple :

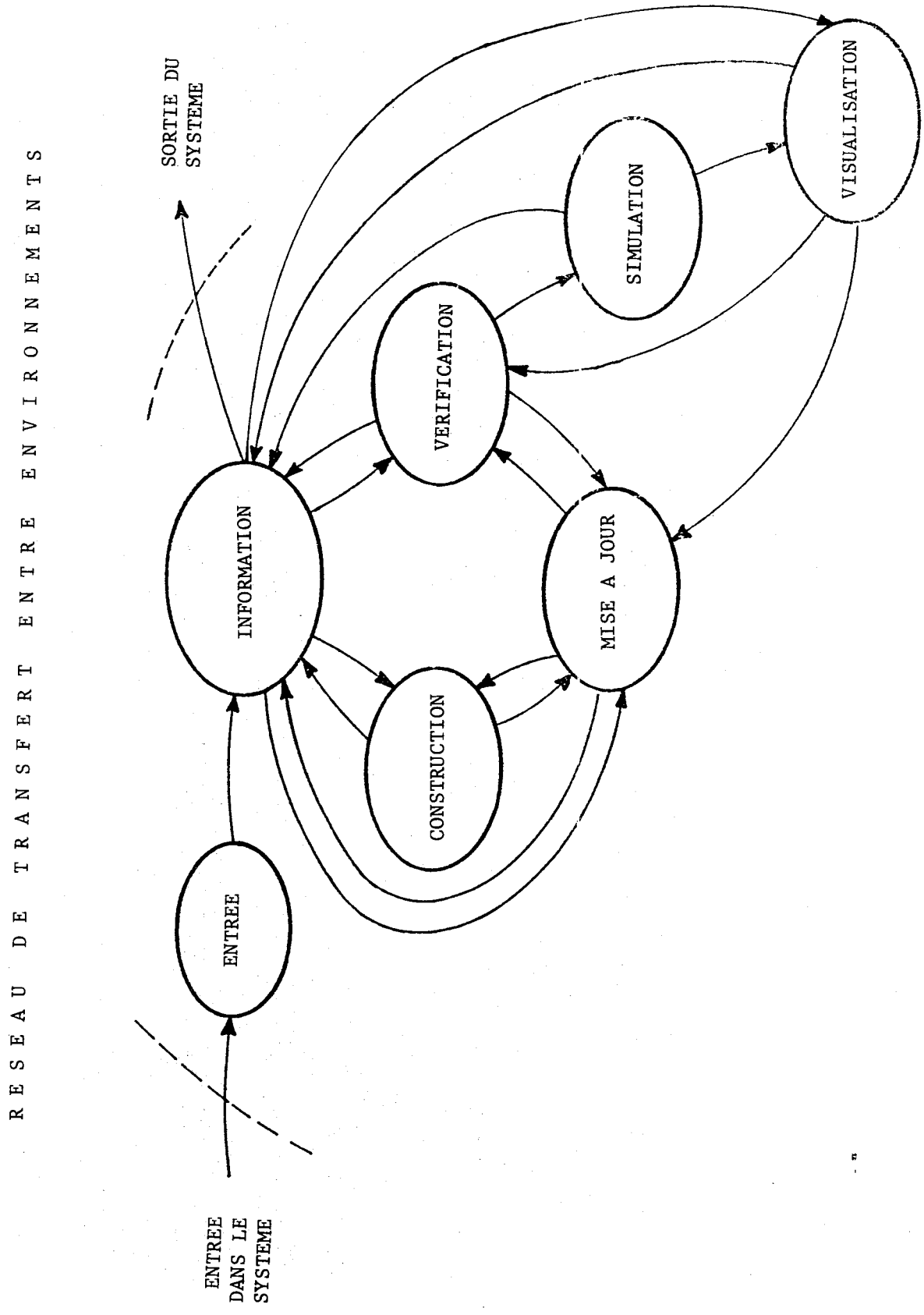


A cela on a rajouté des branches qui correspondent à des utilisations moins courantes mais tout à fait envisageables :

1/ Information → Visualisation
Si l'on veut visualiser des résultats des sessions précédentes.

2/ Information → Mise à jour
Si un modèle est partiellement construit à la session antérieure et que l'on veut compléter dans la session présente.

Figure II.12


RESEAU DE TRANSFERT ENTRE ENVIRONNEMENTS
ENTREE DANS LE SYSTEME
SORTIE DU SYSTEME
ENTREE
INFORMATION
VERIFICATION
CONSTRUCTION
MISE A JOUR
SIMULATION
VISUALISATION

3/ Visualisation → Vérification
Ce cas peut se produire, si après avoir simulé et visualisé un couple variante-jeu de données, on veut simuler (pour comparer) un autre couple qui n'a pas été vérifié auparavant.

4/ Information → Vérification
C'est le cas d'une démonstration d'un modèle-jeu de données. Ils ont déjà été testés dans des sessions antérieures ; on accède le plus vide possible à l'environnement de simulation, en transitant par sécurité par l'environnement vérification ; si la vérification existe on ne la refait pas !

5/ Construction $\overset{\rightarrow}{\leftarrow}$ Mise à jour
Si on veut faire des modifications sur un module de la bibliothèque, on accède au module dans l'environnement de construction, on fait les modifications dans l'environnement de mise à jour et on revient à l'environnement de construction pour le reintégrer ; il est néanmoins conseillé de le faire vérifier avant.

6/ Vérification → Construction
Soit si l'on veut construire un module de la bibliothèque après l'avoir vérifié ; soit si l'on veut créer des variantes une fois vérifié le modèle principal.

7/ '*' → Information
A partir de tous les environnements on doit pouvoir facilement sortir du système ou demander une information, c'est pour cela qu'on peut accèder à l'environnement d'information à partir de tous les autres environnements.

On a laissé pour l'instant certaines contraintes, par exemple :

1/ Entrée → Information
Le système de simulation exige toujours que l'on travaille sur un modèle auquel on fasse référence implicitement.

2/ Vérification → Simulation
On ne peut rien simuler, si l'on n'a pas vérifié avant ...

3/ Simulation → Visualisation

L'intérêt d'une simulation est de pouvoir visualiser ses résultats.

Exemples d'utilisation.

- *Premier exemple : le modèle n'existe pas.*

1 - *On entre dans le système. Le nom du modèle est donné et on passe à sa construction.*

2 - *S'il s'agit d'un grand modèle, on demande que la construction du modèle, l'entrée des jeux de données et la documentation soient effectuées à travers la lecture de cartes perforées.*

3 - *La construction d'un petit modèle (ou d'un module) peut se faire directement au terminal. La correction d'omissions ou d'erreurs de frappe etc... est aussi effectuée dans l'environnement de mise à jour.*

4 - *On demande la vérification (pré-compilation et compilation) du modèle (ou d'un module) et des jeux de données correspondants.*

5 - *S'il ny a pas d'erreurs, on passe à l'exécution (i.e. simulation), sinon on revient au 3e point.*

6 - *On demande la sortie des résultats désirés sous des formes diverses et l'on procède à l'analyse de ces résultats.*

7 - *On peut retourner à l'environnement de mise à jour pour effectuer une modification du modèle ou du jeu de données.*

8 - *Il est alors possible de re-simuler le modèle sous de nouvelles conditions, en ayant changé le jeu de données ou le modèle même. On obtient de nouveaux résultats ; on analyse, on compare aux résultats*
9 - *précédents etc... et on revient éventuellement au point 7.*

9 - *On retourne à l'environnement d'information de façon à sortir du système.*

- *Deuxième exemple* : *Le modèle existe partiellement, d'autre part, certains modules sont disponibles en bibliothèque.*

1 - *On entre dans le système.*

2 - *On identifie le modèle que l'on utilise.*

3 - *On demande l'information sur les parties existantes du modèle et les modules-bibliothèque.*

4 - *On construit d'autres unités du modèle soit en utilisant les modules déjà définis en bibliothèque soit en définissant de nouveaux éléments du modèle par lecture de cartes perforées ou directement au terminal.*

5 - *La correction est effectuée, si nécessaire.*

6 - *On procède à la vérification soit d'un nouveau module défini auparavant, soit du modèle avec les modifications incorporées, ainsi qu'à celle d'un ou plusieurs jeux de données pour le test.*

7/ La simulation est effectuée, on passe à la visualisation de résultats.

8/ On effectue les sorties de résultats et l'on va soit au point 5 pour correction, soit au point 7 pour simuler avec un autre jeu de données.

9/ On revient à l'environnement d'information, on sauvegarde les objets désirés et on sort du système.

L'avantage de séparer les fonctions en environnements permet aussi d'éviter la répétition de tâches que l'on fait lors de la simulation, ce qui fait une économie de temps d'utilisation de l'ordinateur. Par exemple :

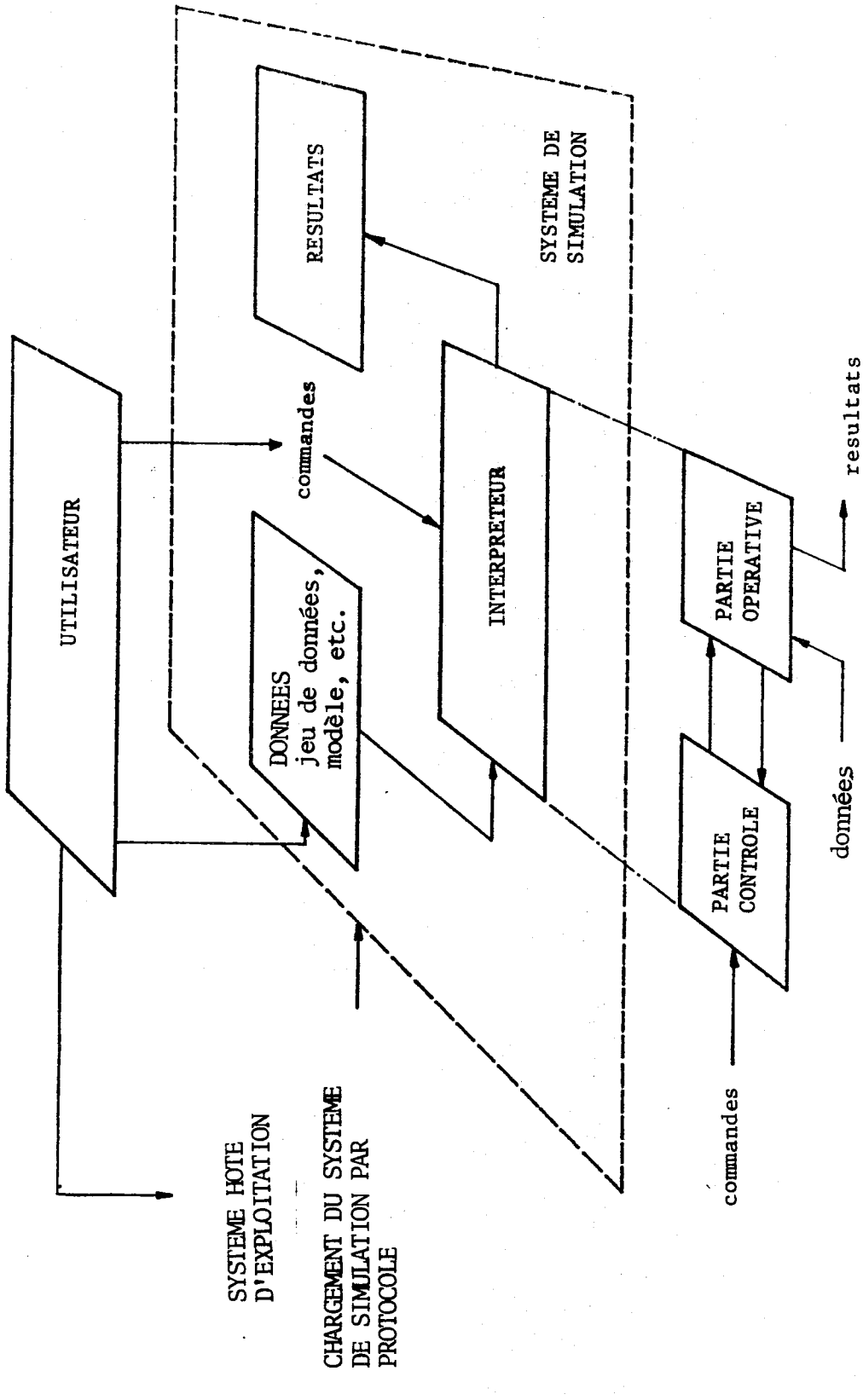
- Changer le pas d'intégration sans recompiler le modèle,
- Ajuster les échelles des sorties graphiques sans re-simuler le modèle, etc...

7 . ARCHITECTURE INTERNE

L'architecture interne du système ne doit pas obligatoirement suivre l'organisation telle qu'elle apparaît à l'utilisateur. En effet, une bonne gestion des ressources et une modularité de sa description impliquent un autre découpage.

De façon schématique, on pourrait décomposer le système comme un automate en une partie contrôle et une partie opérative (voir figure II.13). La partie contrôle se chargera du décodage des commandes et de la gestion au niveau général du système. La partie opérative se chargera d'exécuter les tâches demandées par la partie contrôle.

Figure II.13



VISION DESCENDANTE DU SYSTEME

L'architecture interne de la partie contrôle consiste (voir figure II.14) en un superviseur général de l'interpréteur qui coordonne les activités, un analyseur syntaxique qui fait une lecture et une analyse de la commande ; si la commande est correcte, les fonctions sémantiques qui sont appelées dans la partie opérative sont décodées, sinon l'analyseur sort un message d'erreur et se met en attente d'une nouvelle commande. Comme il y a plusieurs environnements et que chaque environnement à sa propre grammaire, on a besoin d'autant d'analyseurs syntaxiques ; c'est le superviseur qui les validera, en fonction de l'environnement où l'on se situe. Le superviseur général met à jour l'état du système et si l'on doit en sortir, fait les sauvegardes nécessaires à une nouvelle session.

La partie opérative est plus complexe. Un premier découpage fonctionnel est présenté dans la figure II.15. A partir de ce découpage on a défini les modules qui exécutent des tâches concrètes. Ces modules peuvent être appelés par les différentes parties du système, sans distinction d'environnement ; on évite ainsi la répétition de modules dans la description du système. Au chapitre suivant on évoquera plus en détail la façon dont quelques uns de ces modules ont été construits.

Figure II.14

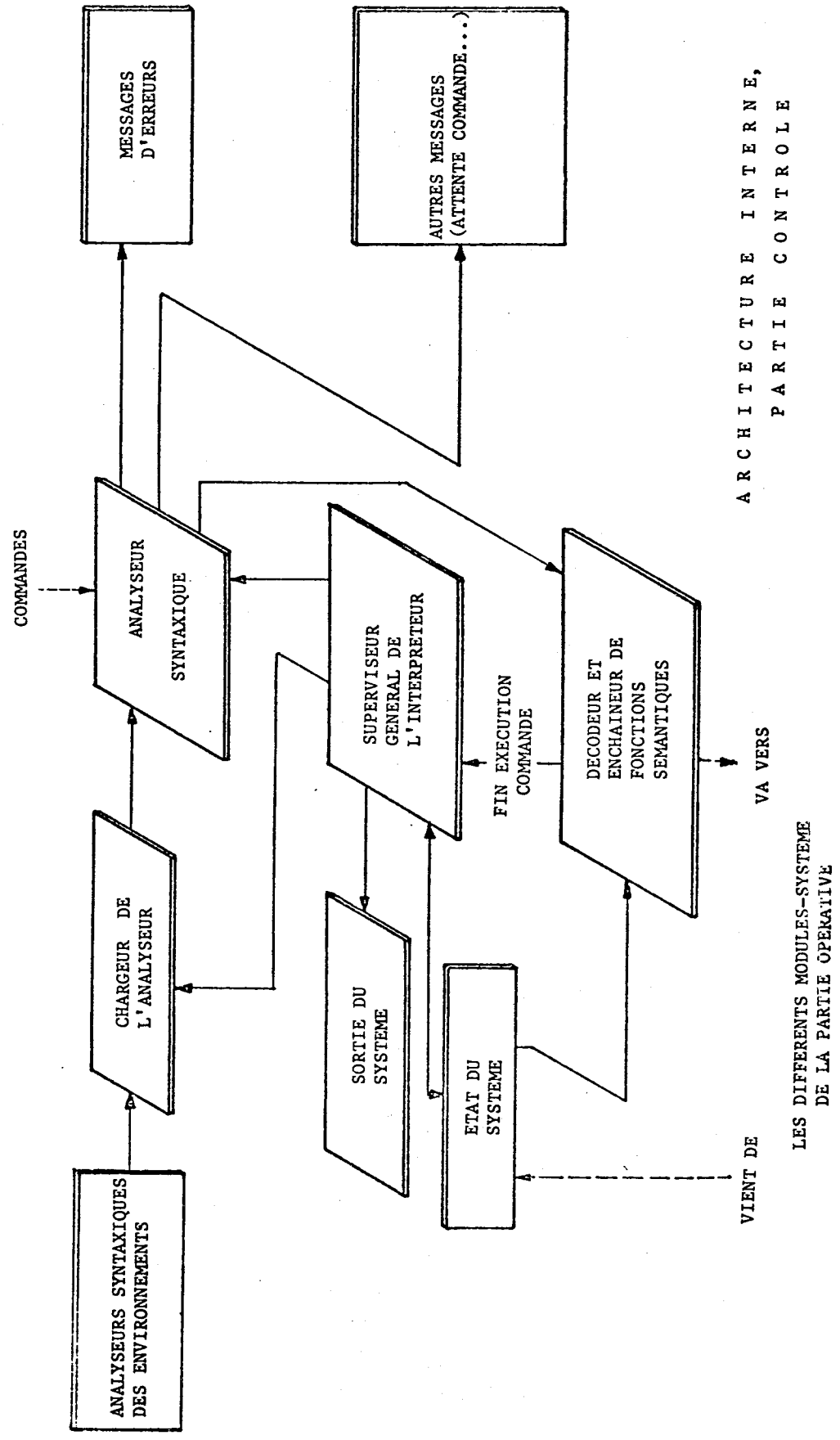
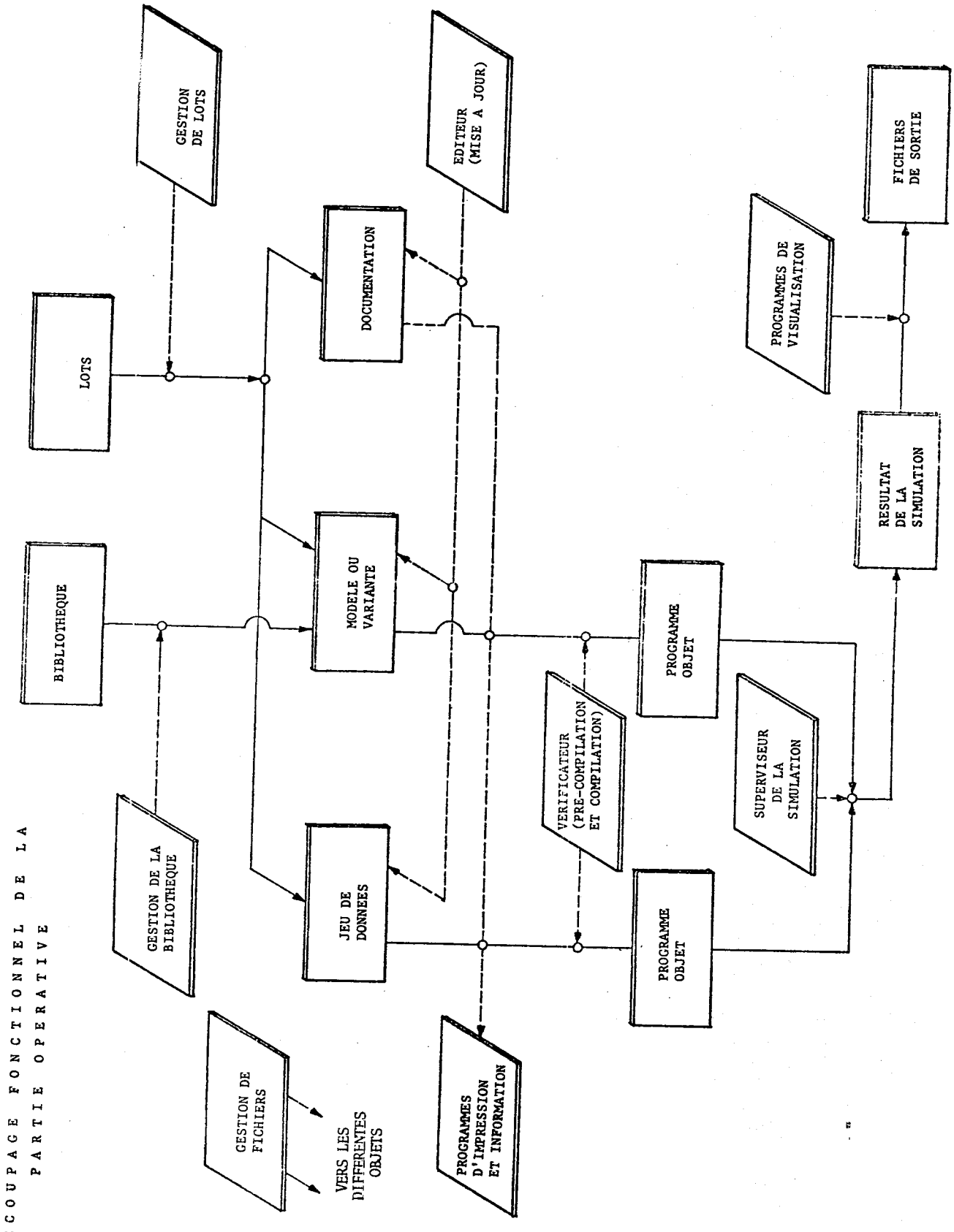

MESSAGES D'ERREURS
AUTRES MESSAGES (ATTENTE COMMANDE...)
COMMANDES
ANALYSEUR SYNTAXIQUE
SUPERVISEUR GENERAL DE L'INTERPRETEUR
DECODEUR ET ENCHAINEUR DE FONCTIONS SEMANTIQUES
FIN EXECUTION COMMANDE
VA VERS
CHARGEUR DE L'ANALYSEUR
SORTIE DU SYSTEME
ETAT DU SYSTEME
VIENT DE
ANALYSEURS SYNTAXIQUES DES ENVIRONNEMENTS
ARCHITECTURE INTERNE, PARTIE CONTROLE
LES DIFFERENTS MODULES-SYSTEME DE LA PARTIE OPERATIVE

Figure II.15


DECOUPAGE FONCTIONNEL DE LA PARTIE OPERATIVE
GESTION DE LOTS
EDITEUR (MISE A JOUR)
FICHIERS DE SORTIE
DOCUMENTATION
PROGRAMMES DE VISUALISATION
LOTS
BIBLIOTHEQUE
MODELE OU VARIANTE
RESULTAT DE LA SIMULATION
PROGRAMME OBJET
VERIFICATEUR (PRE-COMPILATION ET COMPILATION)
SUPERVISEUR DE LA SIMULATION
GESTION DE LA BIBLIOTHEQUE
JEU DE DONNEES
PROGRAMME OBJET
GESTION DE FICHIERS
VERS LES DIFFERENTES OBJETS
PROGRAMMES D'IMPRESSION ET INFORMATION

BIBLIOGRAPHIE

[1] RECHENMANN F., RIVERA E., UVIETTA P.

"A modelling language for the Analysis of Regional Systems". Proceedings of the International Symposium *Simulation'75*, Acta Press, Zürich, June 1975

[2] RECHENMANN F.,

"Analyse et modélisation descendantes des systèmes socio-économiques".
Thèse Docteur-Ingénieur - I.N.P.G. - Juin 1976

[3] I.B.M.

"System/360 Continuous System Modeling Program, User's Manual",
Octobre 1969

[4] RECHENMANN F., RIVERA E., UVIETTA P.

"Modélisation en Dynamique de Systèmes : DYNAMO",
à paraître - Rapport de Recherche ENSIMAG

[5] NAYLOR T., BULENTFY J., BURDICK D., KONG-CHU,

"Computer simulation techniques"
John Wiley & Sons, 1966, p.300

[6] Simulation Software Committee,

"The Sci, Continuous System Simulation Language (CSSL)"
SIMULATION, Décembre 1967, pp.281-303

[7] RIVERA E.

"Le système de simulation sur EXEC"
Note interne à l'équipe CAO. 1975

[8] CREVEUIL M.

"Macro-mécanisme pour le langage de commande d'un système conversationnel"
Thèse 3e Cycle - USMG, 1974

[9] Equipe CP/CMS

"Extension du langage de commande de CMS, Commande EXEC"
Note Technique T13 - C.I.C.G., 1974
voir aussi IBM "CP-67/CMS, Version 3.1 - User's Guide, 1972

[10] COURTIN J., VOIRON J.
"Traduction des schémas de programmes en FORTRAN, Application aux structures de données", IUT-B Informatique Grenoble 1974-75

[11] VEILLON G.
"Algorithmique"
Laboratoire d'Informatique, USMG-INPG, Grenoble, 1974

[12] TENNY T.
"Structured Programming in FORTRAN"
DATAMATION, July 1974, pp.110-115

[13] C.I.I.
"Normes de Programmation FORTRAN IV CII"
Manuel d'Utilisation, Janvier 1973

[14] WRIGHT D.L.
"A comparison of the FORTRAN Language Implementation for Several Computers"
C. ACM, Vol. 9 / Nb 2 / February 1966, pp.77-79

[15] YOURDON E.
"Techniques of program structure and design"
Prentice Hall 1975

[16] PARNAS D.L.
"A Technique for Software Module Specification with Examples"
C. ACM, May 1972, pp.330-336

[17] PARNAS D.L.
"On the criteria to be used in decomposing systems into modules"
C. ACM, December 1972, pp.1053-1058.

[18] RIVERA E.
"Interactive software for the simulation of scenarios"
Proceedings of Informatica 76, p.1.104, Bled (Yugoslavie), Octobre 1976

[19] MEADOW C.T.

"Man-Machine Communication",
John Wiley & Sons - 1970

[20] KUNTZMANN J.

"Théorie des Réseaux",
Dunod - 1972

CHAPITRE III

UN EXEMPLE D'UTILISATION

TABLE DES MATIERES

1 . PRESENTATION GENERALE D'UN MODELE

C'est à travers un exemple type qu'il semble le plus parlant de mettre en évidence les apports soit au niveau de la méthode d'analyse soit au niveau des possibilités techniques qu'offre le système de simulation.

Le modèle avec lequel on travaille dans ce chapitre fait partie d'un projet de modélisation régionale que l'on est en train de développer au sein de l'équipe [1]. Evidemment il en est encore au stade d'étude. En effet sur les 23 agglomérations retenues, ce premier effort s'est concentré seulement sur cinq, en gardant une complexité similaire ; il doit donc être vu comme une maquette servant à illustrer l'utilisation et non pas comme une présentation de résultats. Il présente néanmoins l'intérêt de ne pas être un exemple *purement académique*.

Le but du modèle est l'étude et la compréhension de la dynamique migratoire (non-pendulaire) entre les principales agglomérations de la région Rhône-Alpes. La région étant le **système** et les agglomérations les **sous-systèmes**. La compréhension des transferts dynamiques a été fondamentalement liée à l'emploi et secondairement au logement [2], [3]. Un schéma du modèle de base au niveau global est montré dans la figure III.1 ; dans la figure III.2 on montre plus en détail les relations entre éléments retenus au niveau hiérarchique le plus bas.

Une exposition plus détaillée de l'étude menée, et du modèle se trouve dans [4], [5].

Figure III.1

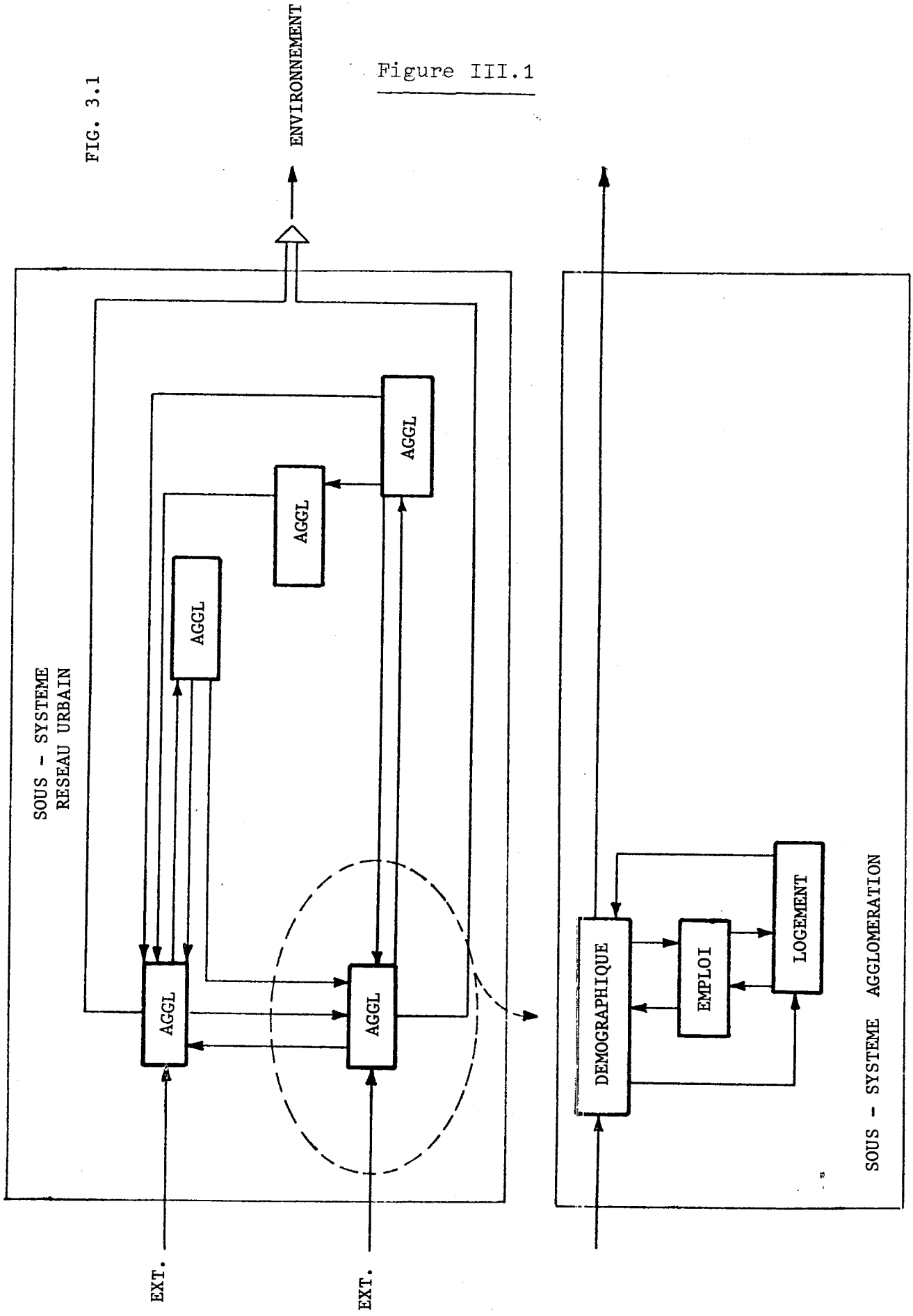


FIG. 3.1
ENVIRONNEMENT
SOUS - SYSTEME
RESEAU URBAIN
AGGL
AGGL
AGGL
AGGL
AGGL
EXT.
EXT.
DEMOGRAPHIQUE
EMPLOI
LOGEMENT
SOUS - SYSTEME AGGLOMERATION

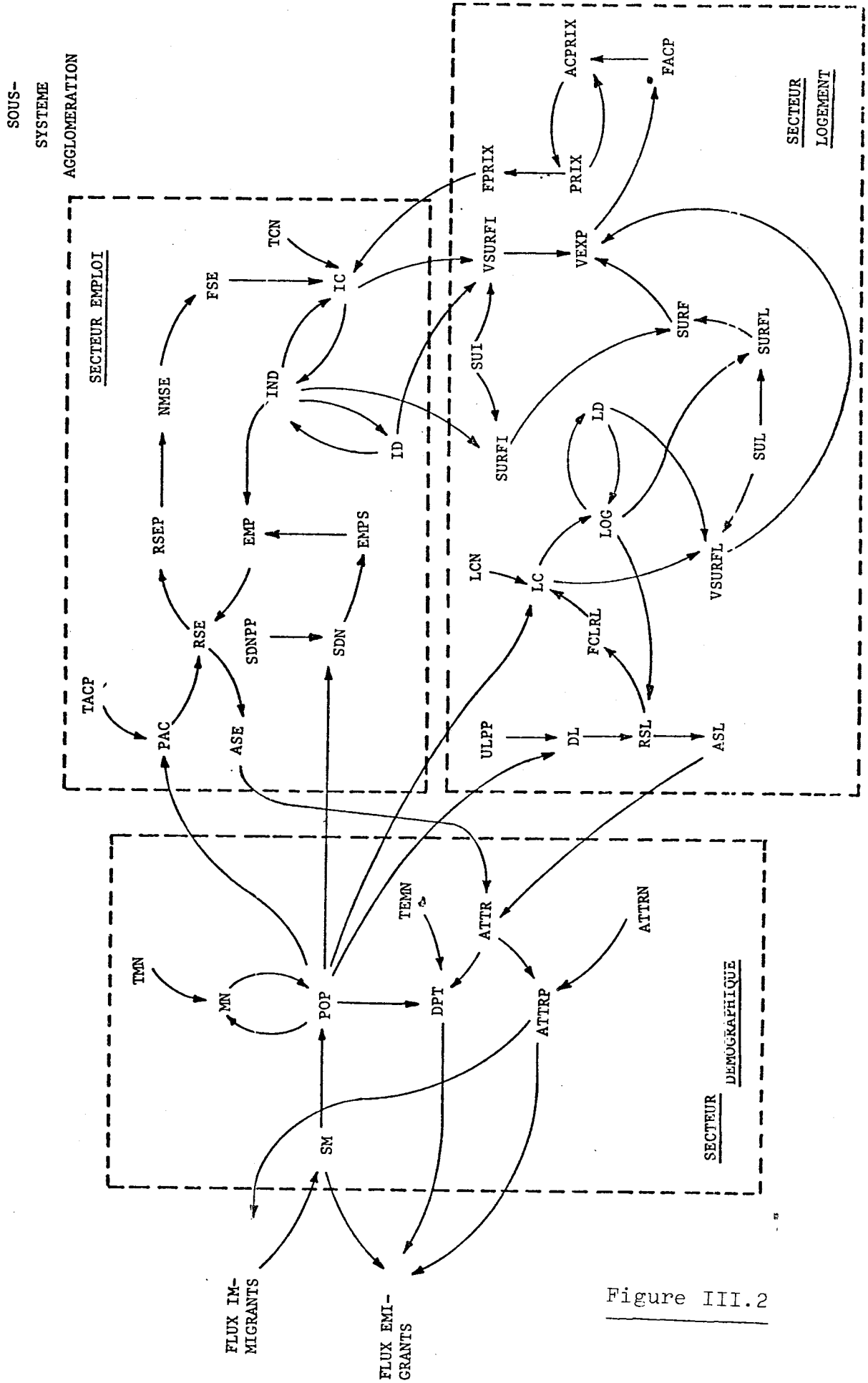

SOUS-
SYSTEME
AGGLOMERATION
SECTEUR EMPLOI
SECTEUR
LOGEMENT
SECTEUR DEMOGRAPHIQUE
TACP
PAC
RSE
SDNPP
SDN
EMPS
EMP
RSEP
NMSE
FSE
TCN
IC
IND
ID
SUI
VSURFI
FPRIX
ACPRIX
FACP
PRIX
VEXP
SURF
SURFL
SURFI
LD
LOG
SUL
VSURFL
LCN
LC
FCLRL
ULPP
DL
RSL
ASL
ASE
TMN
MN
POP
DPT
TEMN
ATTR
ATTRN
ATTRP
SM
FLUX IM-
MIGRANTS
FLUX EMI-
GRANTS

Figure III.2

On montrera ensuite la méthode d'analyse suivie dans la simulation. Nous pensons que ceci présente de l'intérêt car ce processus n'est jamais décrit ni par les réalisateurs de systèmes de simulation, ni par les concepteurs de modèles (ceux-ci se contentent de présenter un modèle tout fait, sans décrire son processus de construction, analyse, test et validation, qui permet de mieux comprendre et critiquer la dynamique du système modélisé). Cependant un premier effort dans ce sens a été fait par l'équipe de D. Meadows qui explique plus en détail [6] le modèle, certains tests et développements faits sur son modèle, pour le Club de Rome [7].

On ne va présenter ni le modèle ni sa phase de conception mais on se bornera à illustrer certaines des démarches représentatives du processus de simulation et d'analyse que l'on a fait ; comme on l'avait déjà indiqué de multiples retours en arrière et certaines remises en cause sont nécessaires.

On rappelle que la modularité d'un modèle ne permet pas seulement une simplification de la description, mais aussi la possibilité de simuler et d'étudier une partie clef du système. La démarche descendante [8] guide et assure une cohérence à chaque niveau d'agrégation dans la conception de la description ; la simulation ne suit pas nécessairement la même démarche à cause de la difficulté de simuler l'interface du modèle avec un niveau d'agrégation plus bas. Ce qui dépend bien évidemment du modèle, et de la connaissance du comportement ; ayant une idée du comportement modulaire du niveau inférieur, on peut tester le niveau supérieur, pour ensuite revenir au niveau inférieur. C'est ainsi que l'on présente d'abord un module et puis les scénarios.

2 . CONSTRUCTION ET TEST D'UN MODULE

Le module retenu contient une partie de la description du modèle de base : l'agglomération. La création du modèle est faite par lots et on lui associe deux jeux de données pour le test.

Notation : ← commandes émises par l'utilisateur
→ réponses du système.

```
← Protocole de chargement et accès au système de simulation

→ * WELCOME TO SAISYE                    ---------passage automatique à
→ * OK :                                           l'environnement d'entrée
←                    (retour chariot)
→ * THE ENVIRONMENT IS INPUT
← ?                                      ---------demande de la liste
→ * THE POSSIBLE COMMANDS ARE :                    des commandes
→ HELP ; | INFO ; | NAMES ; | MODEL <name> ;
→ * OK :
← MODEL AGGLO ;                          ---------passage à l'environnement
                                                   d'information et déclara-
                                                   tion du modèle.
→ * NEW MODEL                            ---------Il s'agit d'un nouveau
→ * OK :                                           modèle.
← HEP ;
→     |
→ ** SYNTAXIC ERROR, UNKNOWN COMMAND **---erreur
← HELP ;                                 ---------demande de la liste des
                                                   commandes
→ * THE POSSIBLE COMMANDS ARE :
→ HELP ; | INFO ; | NAMES ; | OUT ; | BYE ; | MODEL <name> ; | CHECK ; |
→ UPD <objet>; | STRUCT ; | REPORT ; | KEEP RUN <identif>; | LIST <objects>; |
→ PRINT <objects> ; | OPRINT <objects> ; | BUILD ; | ERASE <object> ; |
→ DISPLAY RUN <identif> ;
→ * OK :
← NAMES MODELS ;
→ * THE AVAILABLE MODELS ARE :
→ REGIONAL, AGGLO
```

```
→ * OK :
← BUILD ;                          --------- passage à l'environnement
                                             de construction.
→ * OK :
← BATCH ;                          --------- construction par lot.
→ * BUILDING VARIANT 1             --------- l'identification de la
                                             variante et du jeux de
                                             données a été allouée
                                             par défaut.
→ * BUILDING DATASET 1
→ * OK :
← DATASET 3 ;                      --------- identification d'un nouveau
                                             jeu de données
→ * NEW DATASET
→ * OK :
← BATCH ;                          --------- construction par lots
→ * BUILDING DATASET 3 ;           --------- construction de ce jeu
                                             de données
→ * OK :
← BATCH ;
→ ** EMPTY BATCH READER **         --------- l'objet lot est vide
← CHECK ;                          --------- passage à l'environnement
                                             de vérification
→ * OK :
← MODEL ;                          --------- demande de vérification
                                             (et compilation) du modèle,
                                             des messages provenant du
                                             compilateur (ou du pré-
                                             compilateur) peuvent apparaître.
→ * OK :
← VARIANT 3 ;                      --------- demande de vérification
→ ** ERROR VARIANT DOES NOT EXIST **
← DATASET 3 ; DATASET 1 ;          --------- on peut faire plusieurs com-
                                             mandes sur la même ligne et
                                             même continuer une commande
                                             sur la ligne suivante.
→ * OK :                           --------- compilation du jeu n° 3,
                                             correcte
→ * OK :                           --------- compilation du jeu n° 4,
                                             correcte
```

Evidemment si une erreur de compilation se produit, on doit aller à l'environnement de mise-à-jour afin de la corriger, pour revenir ensuite à l'environnement de vérification.

On suppose maintenant que l'on va tester le module. Il s'agit d'une procédure délicate et extrêmement longue pour laquelle on ne peut pas donner de *recettes* générales. On peut d'abord essayer d'éliminer les *aléas* de la simulation, c'est-à-dire des erreurs dûes principalement à la méthode d'intégration et au pas.

Il ne faut pas essayer tout de suite les méthodes compliquées ou d'ordre élevé, mais commencer par la plus simple, celle d'Euler ; qui en général devrait être suffisante ; le pas d'intégration sera déterminé par la trace que l'on veut obtenir ; s'il n'y a pas de contrainte on peut laisser au système le choix du pas par défaut. S'il y a contraintes (types délai), des critères supplémentaires pour le choix du pas sont nécessaires (voir [9], [10], [11]). Pour l'exemple retenu on s'est fixé une étude sur 50 ans ; il parait raisonnable de considérer que la connaissance des résultats avec une période d'un an est suffisante pour connaître l'évolution du système. Comme les constantes de temps du modèle sont exprimées en années alors un pas de 1 est retenu

```
← GO MODEL DATASET 1 ;        --------- passage à l'environnement
→ * OK :                                de simulation pour le modèle
                                        (variante 1) et le premier
                                        jeu de données.
```

comme la méthode *par défaut* du système est celle d'Euler on peut ne pas la spécifier

```
← DT = 1.0 ;                  --------- spécification du pas
→ * OK :                                d'intégration
```

← FROM 1977 TO 2025 RUN ; --------- lancement de la simulation, le numéro de la simulation est pris par défaut. On passe automatiquement à l'environnement de visualisation.

→ * RUN NOMBER 1 --------- l'identification du run et le temps de simulation sont imprimés

→ * SIMULATION TIME : 0.86930 seg

→ * OK :

On identifie par un titre le scénario qui sera imprimé dans l'en-tête de sortie précédant chaque graphique.

← TITLE $ TEST MODULAIRE : LOG = L, POP = P, EMP = E $;

→ * OK :

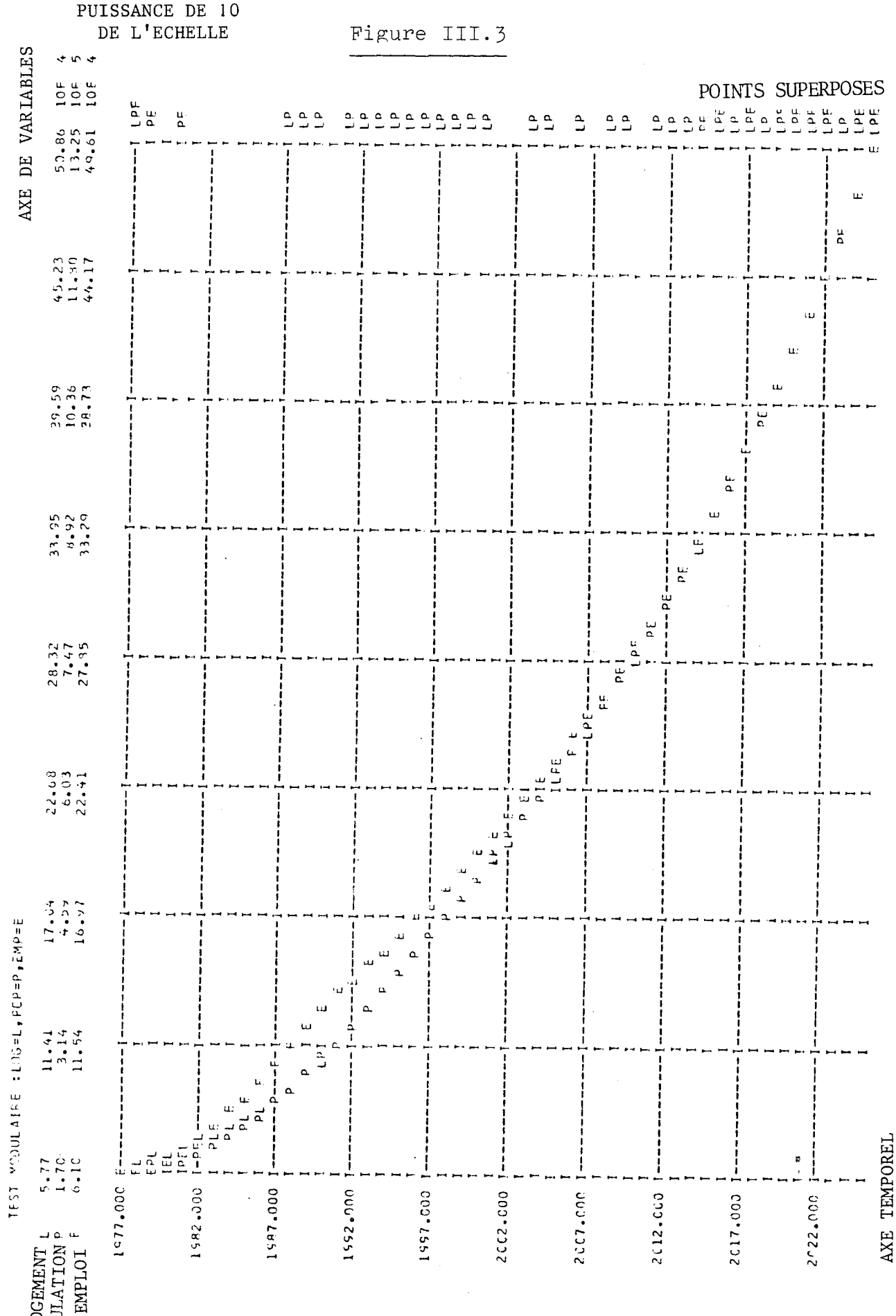
Ce titre sera valable jusqu'à ce que l'on redéfinisse le titre, ou que l'on sorte de l'environnement.

Il faut donc obtenir la trace d'un certain nombre de variables significatives (variables d'état et autres dont la signification est importante) et il est recommandé pour ce test, de laisser les variables *s'exprimer* au maximum, c'est-à-dire que chaque variable ait son échelle déterminée par ses valeurs minimales et maximale obtenues au cours de la simulation. La commande est alors simplifiée, car lorsque les échelles ne sont pas indiquées, chaque variable a sa propre échelle déterminée par son minium et maximum. Si on ne spécifie pas les dates du début et fin de simulation, alors par défaut ce sera itération par itération que l'on va visualiser les résultats de la simulation.

← OPLOT LOG = L, POP = P, EMP = E ;

C'est une demande de visualisation du LOGEMENT représenté par L, de l'EMPLOI représenté par E et de la population par P.
(voir figure III.3).

Figure III.3


PUISSANCE DE 10
DE L'ECHELLE
AXE DE VARIABLES
POINTS SUPERPOSES
AXE TEMPOREL
LOGEMENT L 5.77 11.41 17.04 22.68 28.32 33.95 39.59 45.23 50.86 10F 4
POPULATION P 1.70 3.14 4.59 6.03 7.47 8.92 10.36 11.80 13.25 10F 5
EMPLOI E 6.10 11.54 16.97 22.41 27.85 33.29 38.73 44.17 49.61 10F 4
1977.000
1982.000
1987.000
1992.000
1997.000
2002.000
2007.000
2012.000
2017.000
2022.000

Si l'on veut des sorties sur imprimante rapide on prend les options *OFFLINE*, c'est-à-dire les commandes précédées par un "O" et les visualisations sont faites sur fichier avec un format plus large.

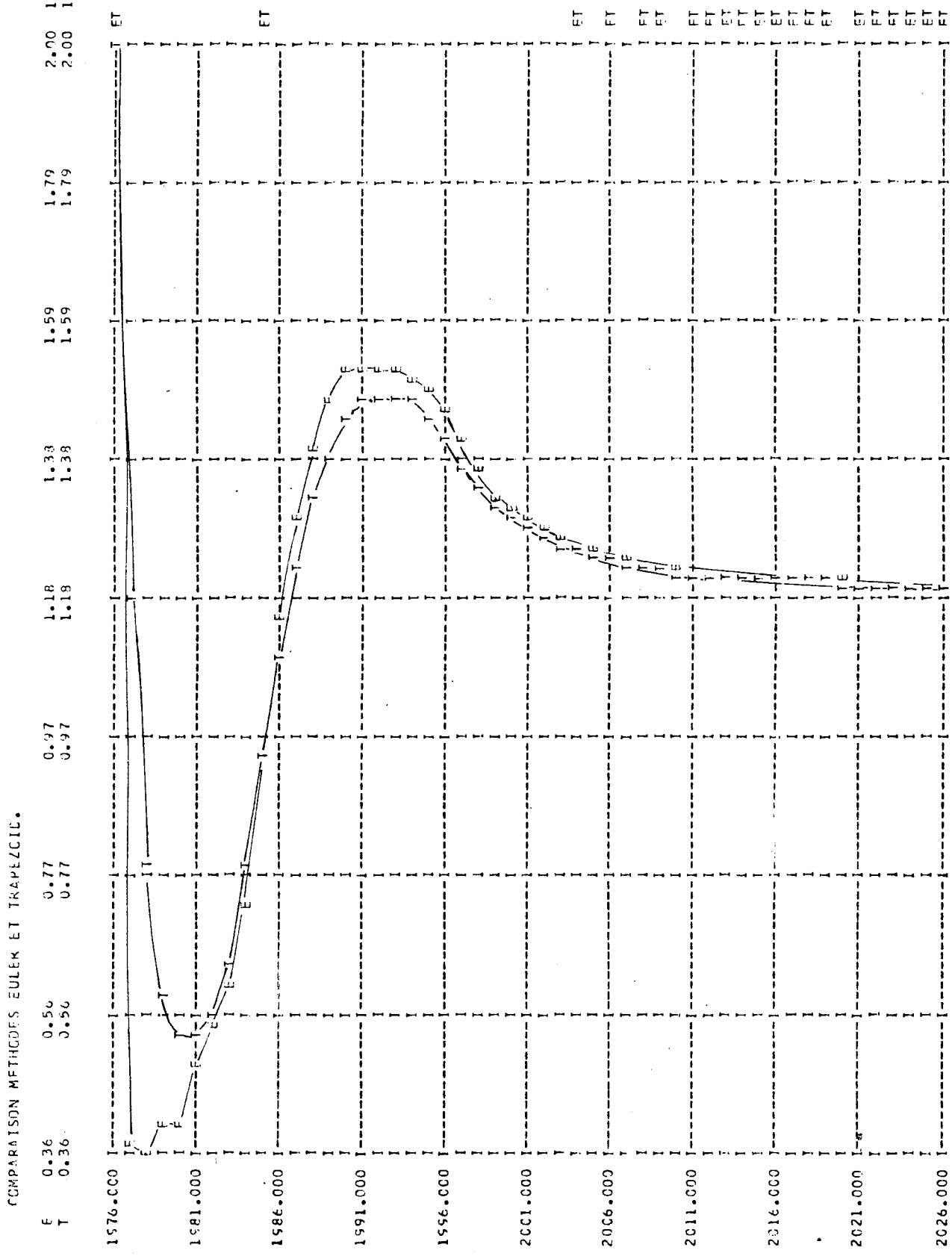
Même si l'on ne rencontre pas des instabilités *apparentes* au cours de la visualisation, il faut toujours essayer de faire au moins une nouvelle simulation avec un pas réduit de moitié. S'il y a encore des variations, on peut demander d'estimer l'erreur et éventuellement ajuster le pas par rapport à une précision donnée ou utiliser la méthode exponentielle (voir Annexe IV).

Dans la figure III.4 on montre des instabilités *typiques* de la méthode d'intégration d'Euler pour la variable ATTRP (attractivité), on peut les éliminer bien sûr, en réduisant d'avantage le pas d'intégration ; en général il est plus économique d'utiliser une méthode d'ordre plus élevé.

Figure III.4

PAS D'INTEGRATION = 1.0

EULER = E, TRAPEZOIDALE = T

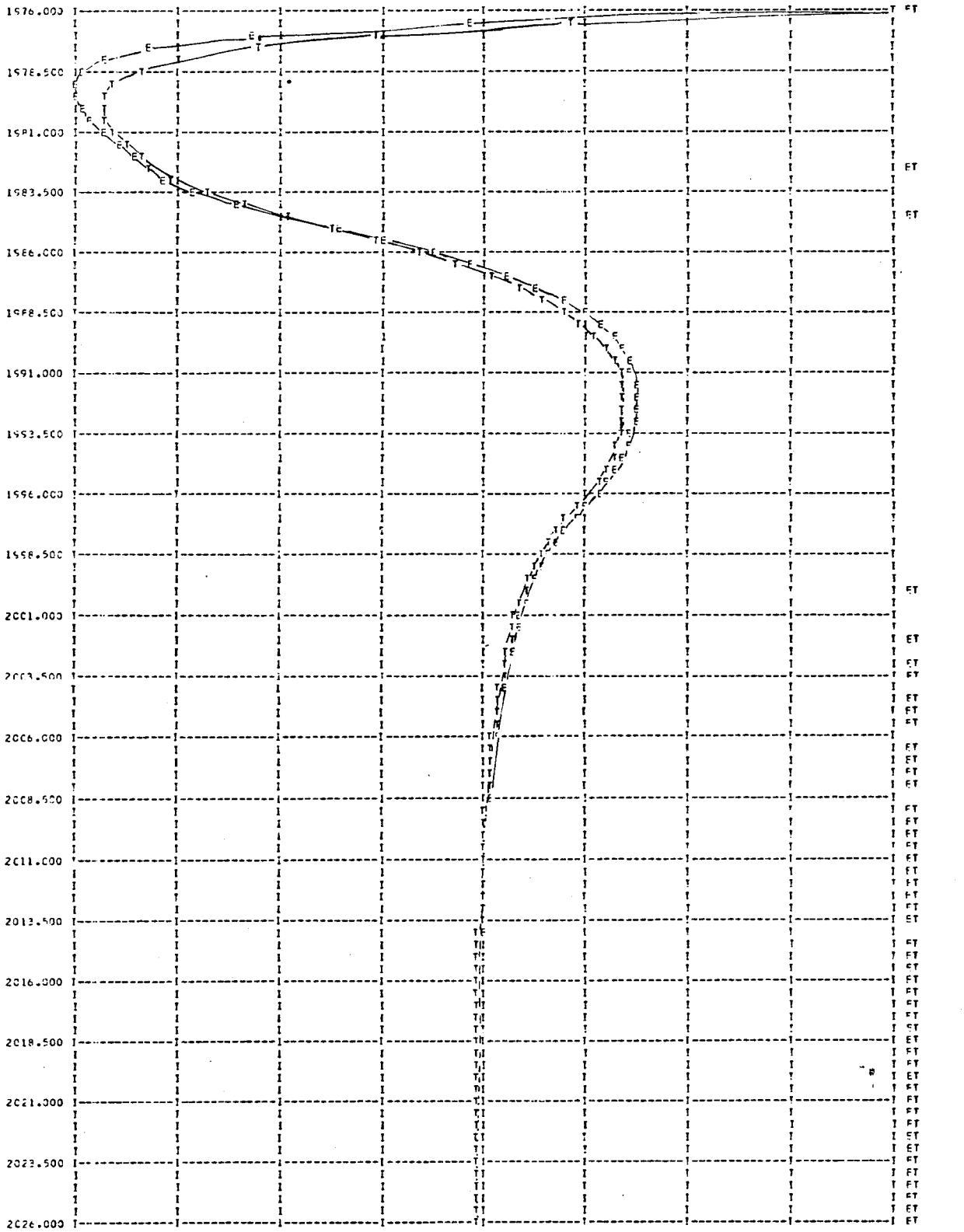


Au cours de plusieurs simulations, on a observé que la méthode TRAPEZOIDALE donne de bons résultats précision/performances (stabilité, temps de simulation).

Voir figure III.5 (Attrap).

PAS D'INTEGRATION = 0.5

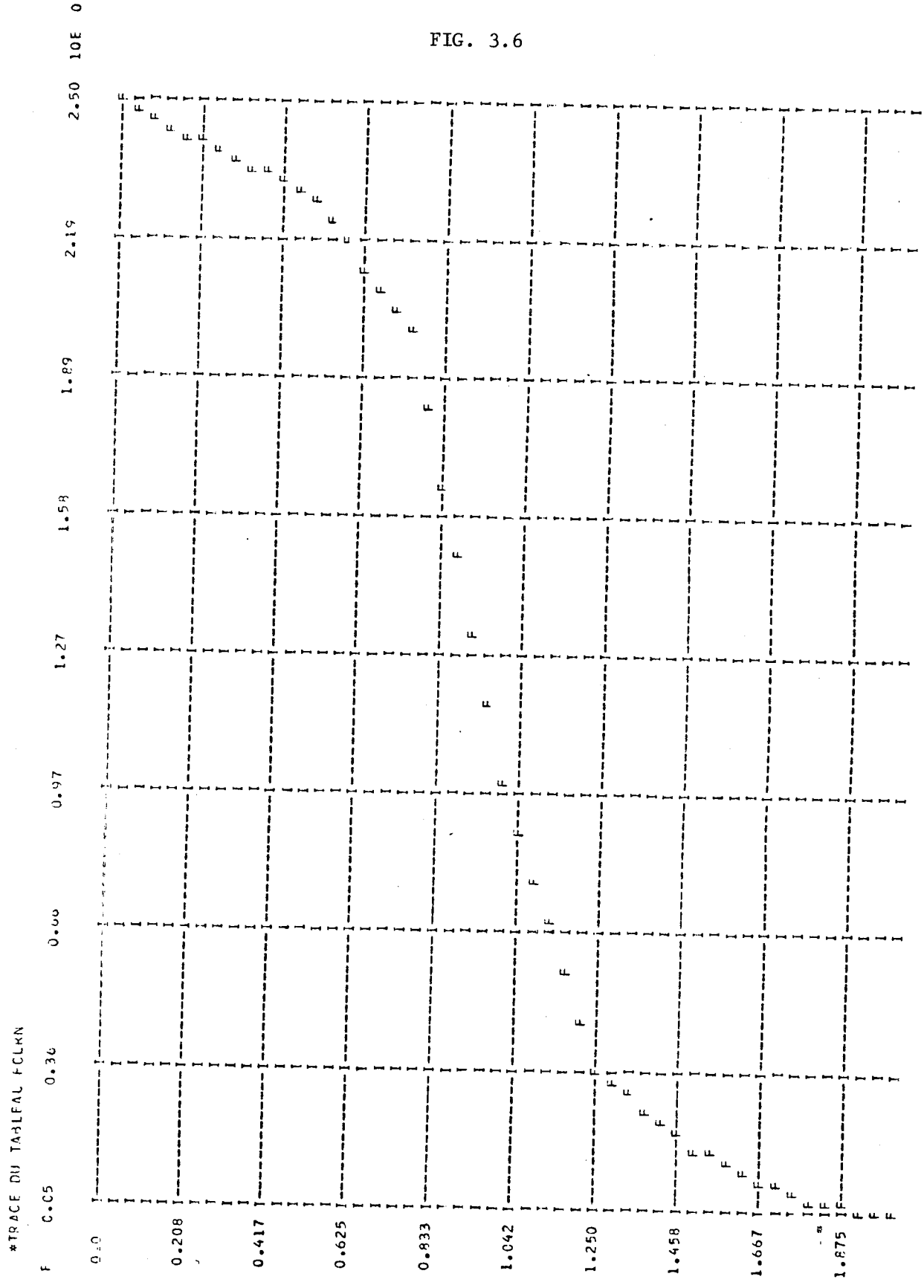
FIG. 3.5 EULER = E, TRAPEZOIDALE = T



C'est à ce moment, que l'on peut commencer à faire d'autres tests. Des tests que l'on va appeler *correcteurs* ou *normaux*, c'est-à-dire des tests où l'on essaie de corriger la valeur de certains coefficients intervenant dans les relations et des tests que l'on va appeler *critiques*, c'est-à-dire des tests où l'on change les valeurs de coefficients *exogènes* au système en essayant d'abord de les perturber peu, pour ensuite donner à ces coefficients des valeurs que l'on juge limites ou critiques ; c'est de cette dernière façon que l'on va réaliser ce que l'on appelle *test de sensibilité* ; en effet traditionnellement le test de sensibilité ou bien n'est pas fait, ou bien est réalisé par méthodes pseudo-aléatoires. Les méthodes pseudo-aléatoires, en tout cas, ne peuvent pas être interprétées aisément, car ce n'est pas nécessairement la stabilité du système modélisé que l'on cherche ; il est donc préférable vu le temps et le coût des essais aléatoires de faire des tests déterministes plus ponctuels dont l'interprétation est plus évidente.

Une partie importante des tests *normaux* consiste en la vérification des variables définies par des tableaux. Un tableau correspond à une définition fonctionnelle, entre deux variables, obtenue à partir de son graphe en divisant en segments de longueur égale le domaine de la variable indépendante et en attribuant les valeurs correspondantes à la variable dépendante.
La fonction TABLEAU fait par exemple une interpolation linéaire entre chaque couple de points du graphe ainsi défini. Dans l'exemple on montre le tableau FCLRN (facteur de construction lié au sous-logement) défini en fonction de la variable RSL (Ratio de Sous-Logement). (Voir figure III.6.)

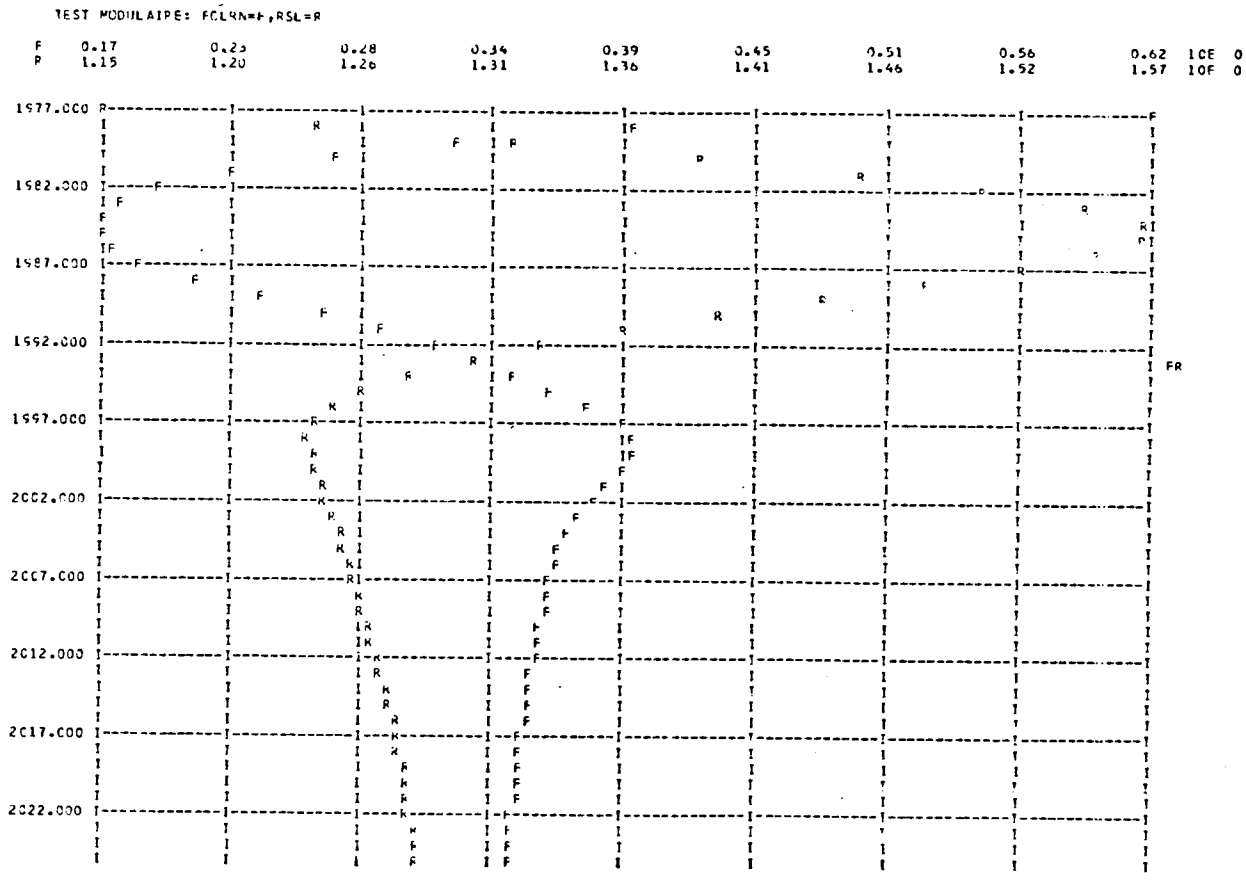
FIG. 3.6


*TRACE DU TABLEAU FCLKN
F C.C5 0.36 0.00 0.97 1.27 1.58 1.89 2.19 2.50 10E 0
0.0
0.208
0.417
0.625
0.833
1.042
1.250
1.458
1.667
1.875

Or au cours de la simulation la variable indépendante RSL n'atteint pas tout le domaine prévu. L'appréciation de ce phénomène n'est pas facile dans une courbe par rapport au temps (Figure III.7), mais plutôt à travers le BISPACE (Figure III.8).

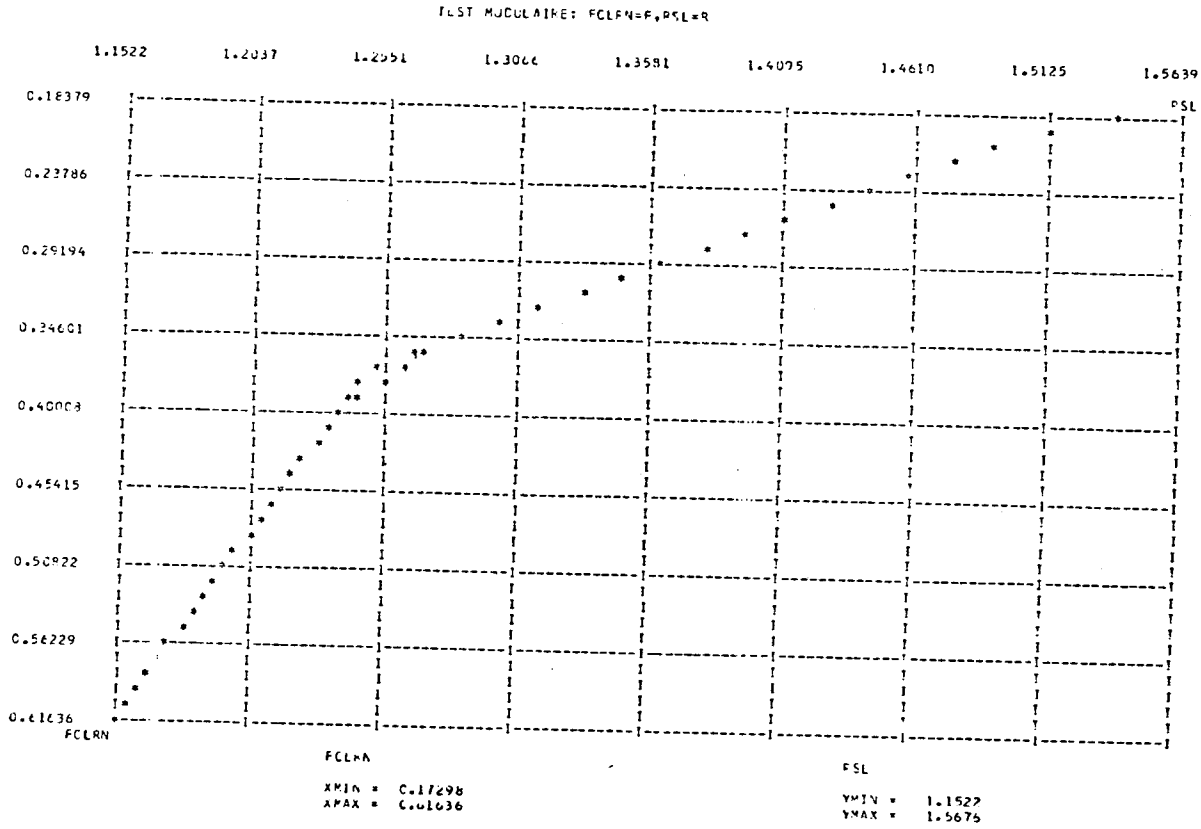
← OPLOT FCLRN = F,RSL = R ;
→ * OK : (Figure III.7)

Figure III.7



← OBISPACE FCLRN, RSL ;
→ * OK : (Figure III.8)

Figure III.8

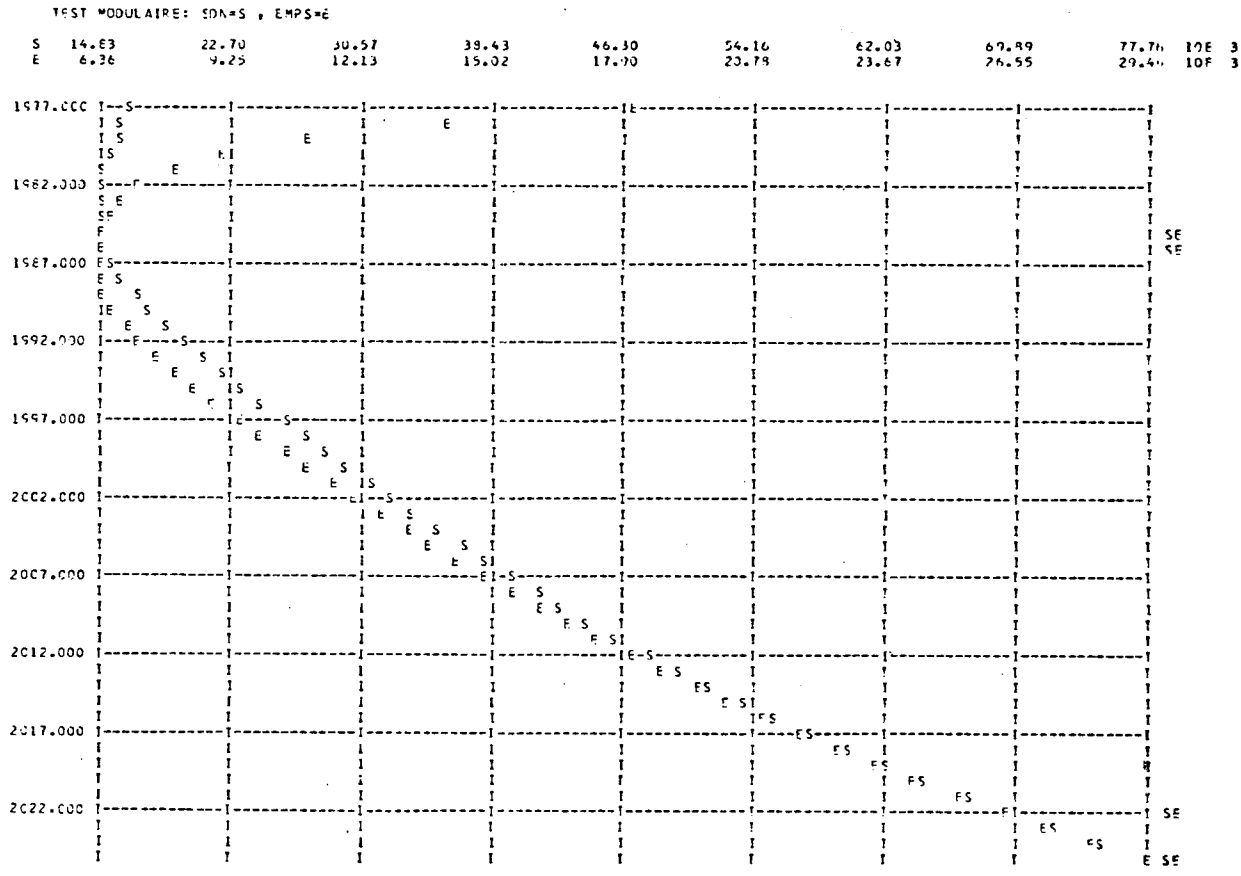


Comme on l'observe (dans la figure III.8), c'est seulement dans une partie restreinte de la plage de fonctionnement de la fonction tableau que l'on travaille ; ceci est extrêmement important à saisir surtout si l'on est près de la limite de définition de la fonction.

Parmi les tests normaux les plus indispensables à faire restent ceux relatifs aux fonctions *délais continus* non pas seulement pour le choix du type, ordre et magnitude du délai mais aussi pour les répercutions dans l'ensemble du modèle. Certains critères pour le test de délai se trouvent dans [5] et [6]. Dans l'exemple on a supposé qu'entre le nombre d'emplois dans les services (EMPS) et le nombre désiré d'emplois dans les services il y a un délai de type information de 3ème ordre, la magnitude entière du délai a été supposée de 2 ans (voir figure III.9).

← OPLOT SDN = S, EMPS = E ;
→ * OK :
(Voir figure III.9).

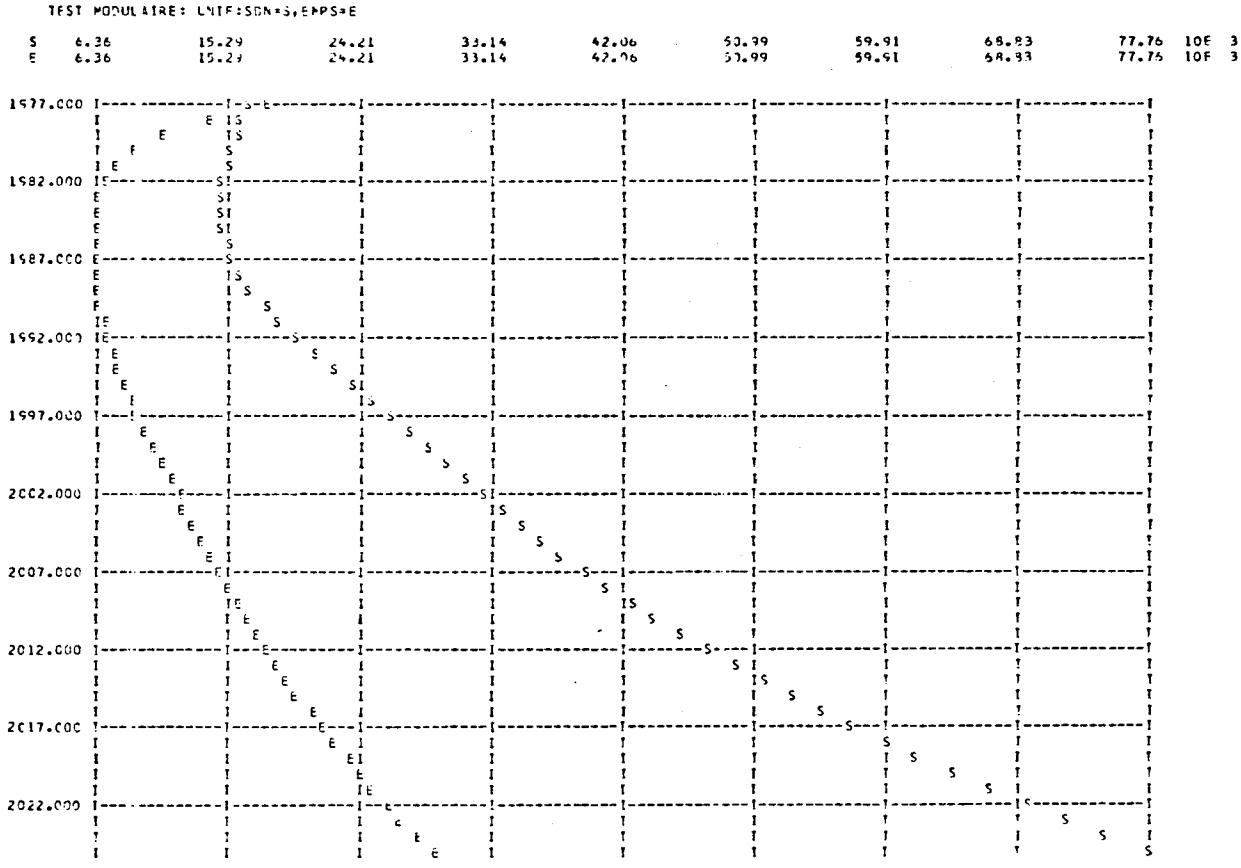
Figure III. 9



L'option UNIF pour comparer les valeurs des variables à la même échelle peut être donc nécessaire.

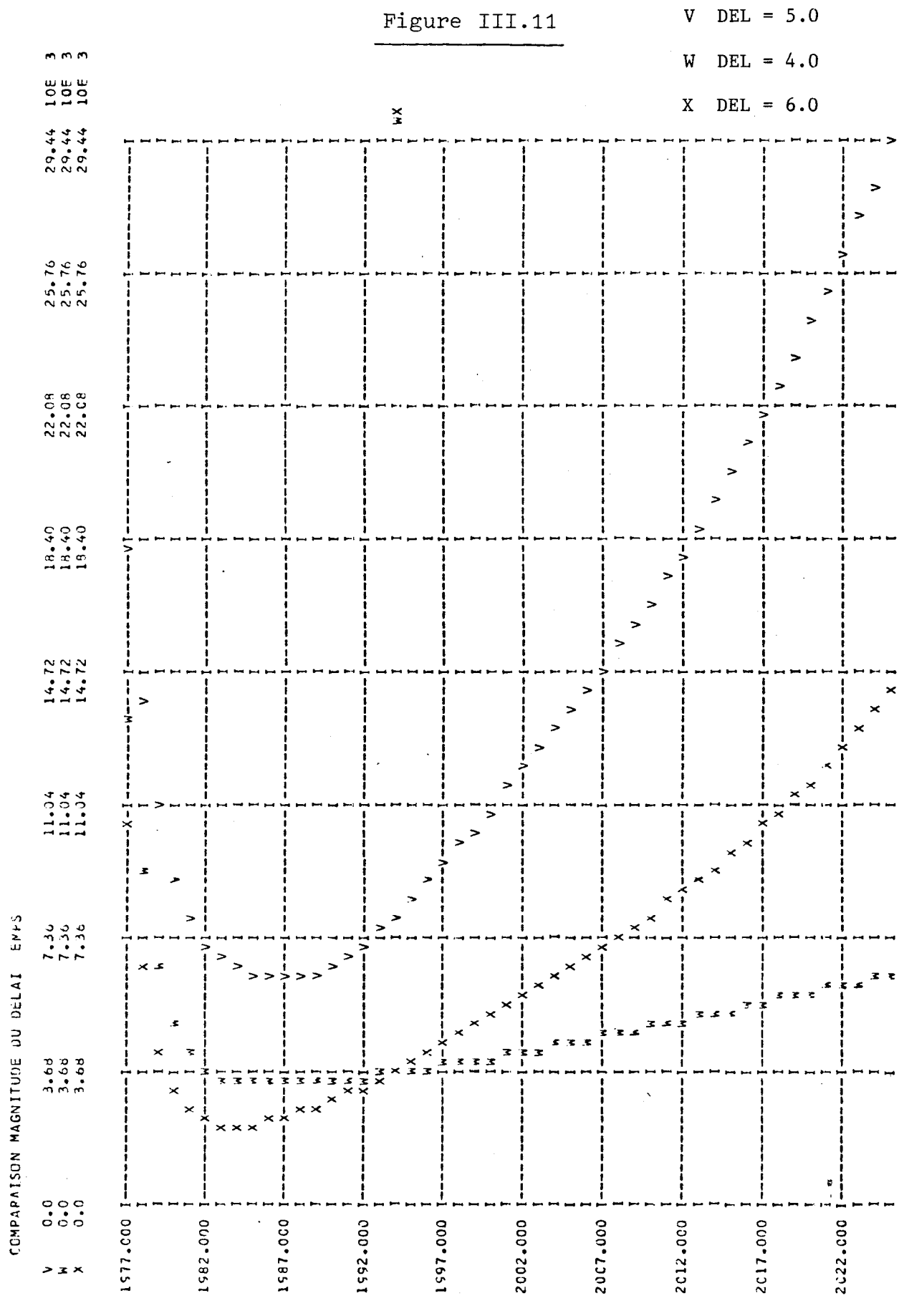
← OPLOT UNIF SDN = S, EMPS = E ;
→ * OK :
(voir figure III.10).

Figure III.10



La magnitude d'un délai est parfois difficile à déterminer, en particulier si celle-ci dépend du niveau de la *perception qualitative* d'une valeur que l'on doit satisfaire, comme c'est le cas entre SDN et EMPS. Des corrections de ces valeurs devront être faites pour chaque agglomération. (Figure III.11).

Figure III.11



En ce qui concerne les tests critiques, on pense que ce n'est pas la peine de créer de nouveaux jeux de données, car ils ne seront plus utilisables d'une part, et d'autre part, c'est seulement la valeur d'un seul coefficient qui en général change. Si le nombre de changements est plus important il peut être intéressant de créer une macro pour changer la valeur du coefficient testé (voir Annexe III). Dans l'exemple on a testé la valeur du coefficient exogène *Taux normal d'immigration* TIMN. On passe donc de l'environnement de visualisation à celui de mise à jour.

```
← UPD DATASET 1 ;          --------- Passage à l'environnement de
                                     mise à jour
→ * OK :
← L 'TIMN/0.05/' ;         --------- localisation de la chaîne
→ * DVMI/50./,TACP/0.4/,TIMN/0.05/,ATTRN/1./,
→ * OK :
← C '0.05','0.02' ;        --------- remplacement d'une chaîne par
                                     une autre
→      * DVMI/50./,...,TIM/0.02/,...
→ * OK :
← LI ;
→ 0023                     --------- numéro de la ligne
→ * OK :
← CHECK ;                  --------- passage à l'environnement de
                                     vérification
→ * OK :
← DATASET 1 ;              --------- vérification premier jeu de
                                     données
→ * OK :
← GO MODEL DATASET 1 ;     --------- passage à l'environnement de
                                     simulation
→ * OK :
← TRAPEZ ;                 --------- méthode trapézoïdale
→ * OK :
← DT = 0.5 ;               --------- indication du pas d'intégration
→ * OK :
```

← FROM 1977 TO 2025 RUN 2 ; --------- simulation entre 1977 et 2025 ayant comme identif. le n°2

→ * OK : passage à l'environnement de visualisation

les mêmes opérations sont répétées pour les valeurs TIMN=0.01(RUN3),TIMN=0.03(RUN4),TIMN-0.07(RUN5)

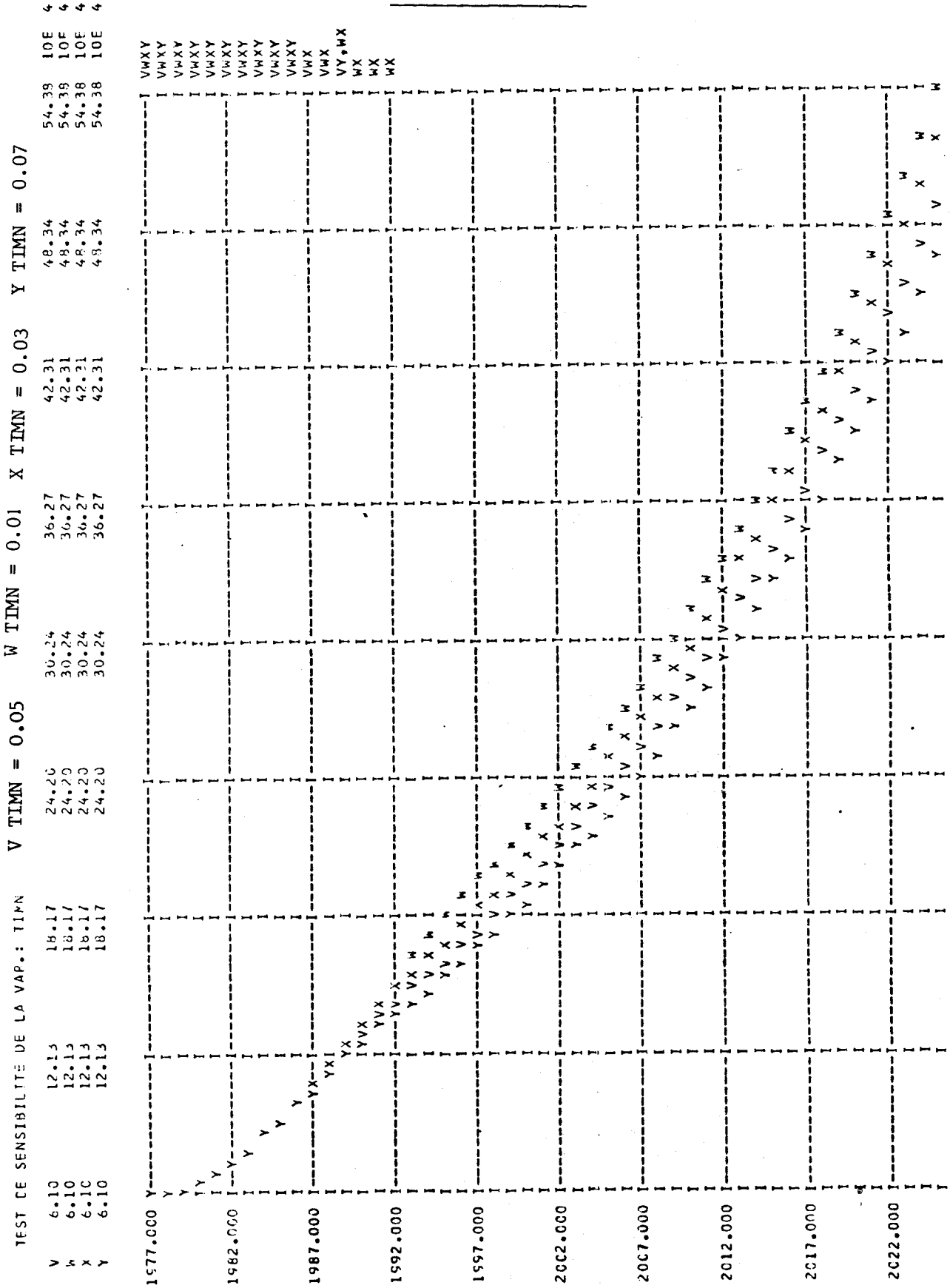
Cela permet de faire une comparaison entre ces tests de sensibilité pour la variable Emploi

← OCOMPAR EMP PLOT RUNS 2=V,3=W,4=X,5=Y ;

→ * OK :

(voir Figure III.12)

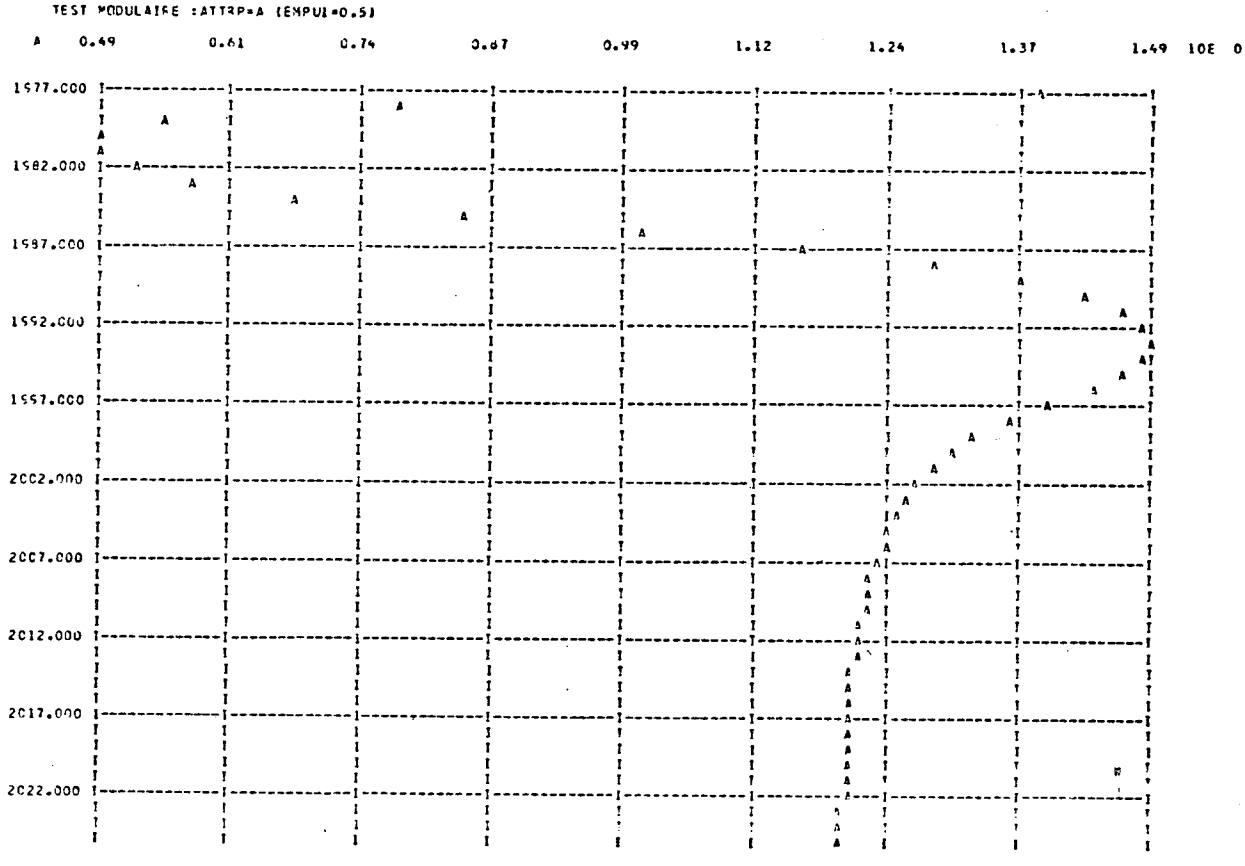
Figure III.12



Or chaque test de sensibilité n'a pas les mêmes variables *indicateurs*, car celles-ci sont retenues en vue des objectifs de l'étude et de l'imbrication de la variable dans le modèle. Il est fréquent d'avoir des surprises dans le comportement du modèle lors d'un changement, mais un effort de compréhension minutieux doit être fait avant de *s'aventurer* à une modification définitive du modèle de base.

Ainsi une légère variation dans la pondération du nombre d'emplois industriels EMPUI (par exemple par l'arrivée d'une nouvelle technologie qui permet une automatisation plus grande sur l'industrie principale) aura non seulement de grandes répercutions sur l'emploi mais aussi sur l'attractivité. Comparez la figure III.4/III.5 (EMPUI=1) et la figure III.13 (EMPUI=0.5).

Figure III.13



Une fois le module suffisamment testé, celui-ci peut être stocké dans la bibliothèque pour être inséré par la suite dans le modèle régional de base. Il ne faut pas oublier d'effacer les objets dont on ne se servira plus, par exemple certaines simulations.

```
← !                          --------- retour à l'environnement
                                       d'information
→ * INFORMATION ENVIRONNEMENT
→ * OK :
← ERASE RUN 2 ;              --------- effacement du produit d'une
                                       simulation (le n°2)
→ * OK :
← ERASE RUN 3 ;              --------- effacement simulation 3
→ * OK :
← ERASE RUN 4 ;              --------- effacement simulation 4
→ * OK :
← BUILD ;                    --------- passage à l'environnement de
                                       construction
→ * OK :
← LIBRARY PUT 2 ;            --------- le modèle courant devient le
                                       module 2 de la bibliothèque
→ * OK :
← INFO ;                     --------- passage à l'environnement
                                       d'information
→ * OK :
← BYE ;                      --------- sortie du système SAISYE
→ * RETURN TO CMS ;
```

3 . SIMULATION ET ANALYSE DE DIVERS SCENARIOS

On va étudier un modèle régional comportant seulement cinq agglomérations du type de celle testée au paragraphe précédent avec de légères variations ; trois types d'agglomérations sont représentés : *Métropole* (où il existe tous les types d'activité économique et de services), *Sous-métropole* (où le taux de services est plus réduit, mais où ses activités sont assez diversifiées), et *Ville spécialisée* (où il existe une prédominance d'une industrie ou activité employant une grande partie de la population).

On s'est intéressé dans cette étude préliminaire fondamentalement aux flux migratoires entre les agglomérations, car celles-ci représentent une des phases les plus difficiles à mettre en oeuvre : **la connexion entre sous-modèles.**

Les scénarios portent sur l'existence d'une agglomération *relais*, c'est-à-dire d'une agglomération qui a une dynamique propre, mais limitée, où les émigrations se font exclusivement vers une autre agglomération plus attrayante (laquelle en général est une métropole et relativement voisine de l'agglomération *relais*.

On peut définir formellement une agglomération *relais* de la façon suivante :

Si A, B, X, Y sont des agglomérations de la région R et si $A \to B$ indique l'existence d'un flux migratoire de A vers B.

A est une agglomération *relais* si
$\forall$ agglomération X et Y $\in$ R , $x \neq A$ et $y \neq A$
on a les trois conditions suivantes :

1/ Il existe **au moins une** X tel que $X \to A$
2/ $\forall$ X tel que $X \to A$ alors / $A \to X$
3/ Il existe **un seul** y tel que $A \to y$

L'agglomération *relais* doit être conçue plutôt comme une zone *tampon* que comme une zone de *transit*, car ce dernier concept impliquerait l'existence d'une population enracinée et d'une population flottante. Or ce n'est pas le cas.

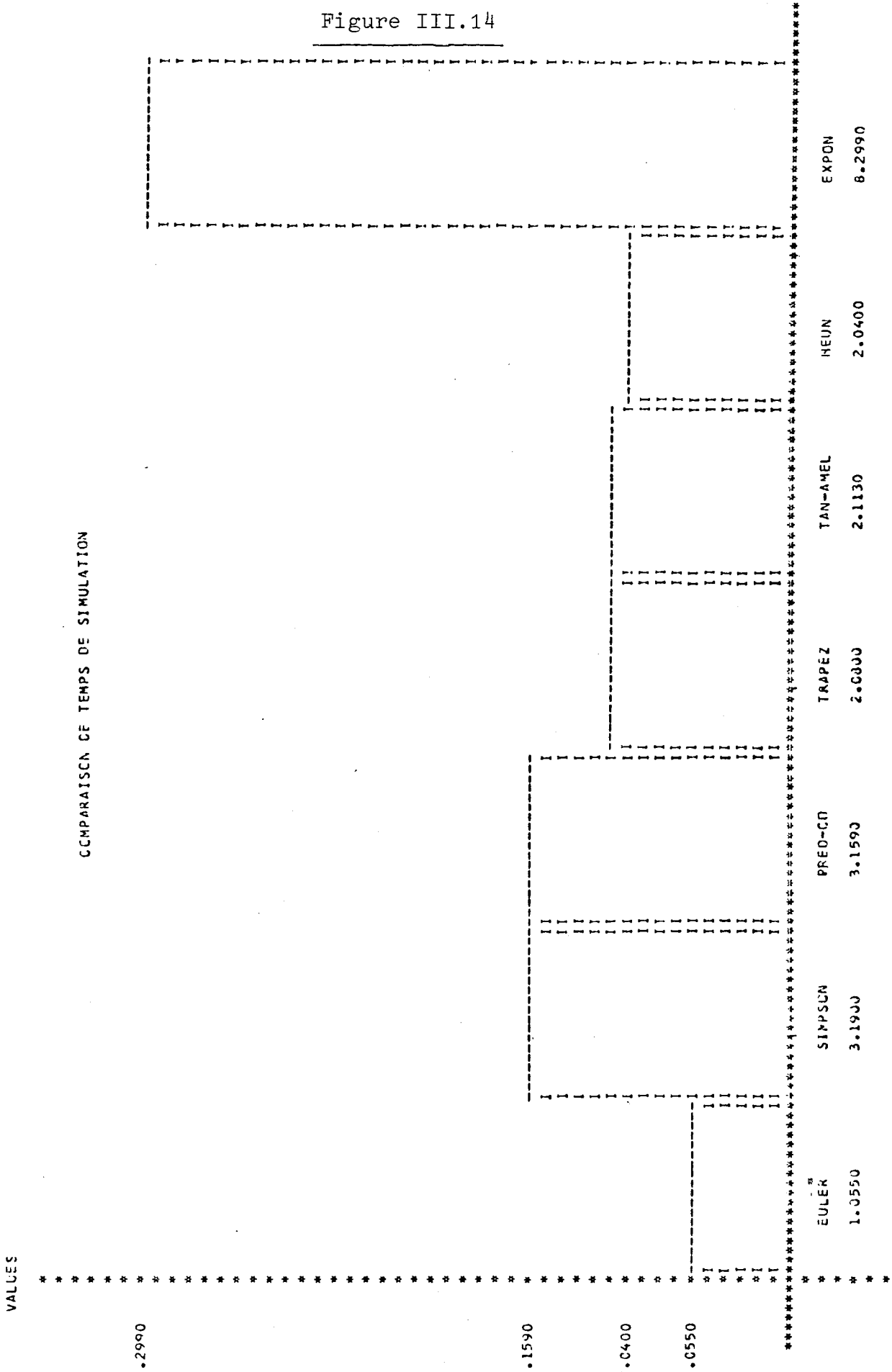
Il y aura deux types de changements qui engendreront les scénarios, un **conjoncturel** et **tendantiellement distinct** (le phénomène de saturation d'une agglomération) et l'autre **structurellement distinct** (le développement des nouveaux flux migratoires), ces deux changements peuvent se combiner. Le fait qu'une agglomération commence à être saturée induit un excédent de population qui normalement émigre vers les autres agglomérations du réseau, à ce moment, alors, de nouveaux flux migratoires peuvent se créer.

Le modèle est de taille moyenne (45 variables d'état, et en développant les régions et secteurs environ 500 équations), ce qui fait par exemple sur une simulation à moyen terme (50 ans) et un pas d'intégration de 0.5 an, 4500 intégrales. Les temps de simulation (voir figure III.14) obtenus, dépendent donc de l'ordre de la méthode d'intégration ; la méthode exponentielle étant une méthode à pas variable, ne peut pas être comparée directement aux autres méthodes.

La construction des modèles représentant les scénarios est analogue à celle montrée lors de la construction d'un module.
On peut s'en servir, du fait que l'on a créé un module en bibliothèque contenant la description d'une agglomération type, pour amener ce module et l'ajouter au nouveau modèle REGIONAL qui est le scénario de base. Dans l'environnement de **construction** on fera

```
← LIBRARY GET 2 ;          --------- apporter le module 2 de la
                                     bibliothèque et l'incorporer
→ * OK :                             au modèle
```

Figure III.14



Le reste de la construction du scénario de base, ainsi que les changements pour l'obtention d'autres scénarios pourront être faits dans l'environnement de **mise à jour**.
Ainsi par exemple :

```
← UPD VARIANT 2 ;          --------- passage à l'environnement
                                     de mise à jour pour cons-
                                     truire une nouvelle variante
→ * NEW VARIANT
→ * OK :
```

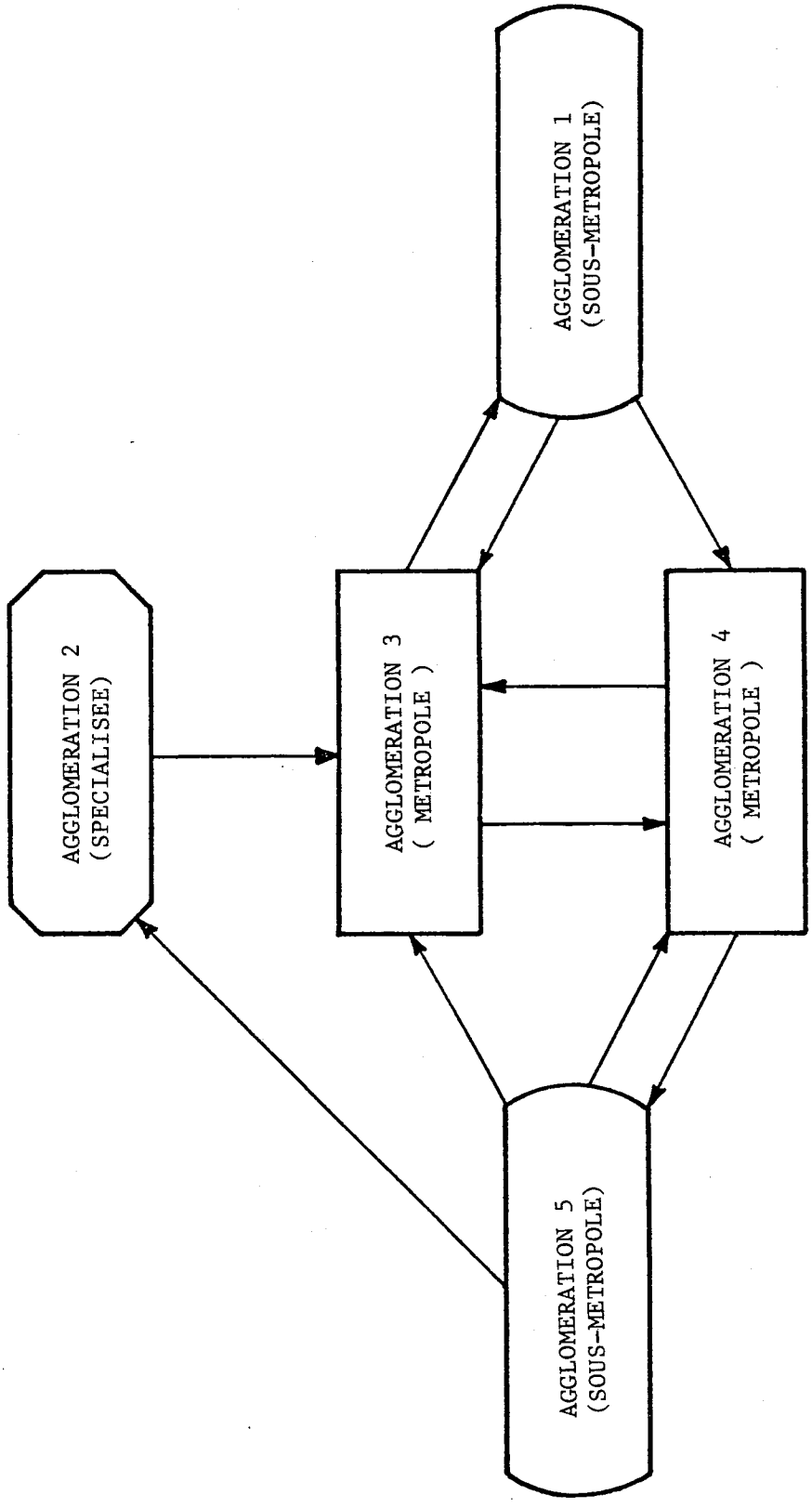
demande la création d'une variante du modèle de base, à partir duquel on fera les changements voulus

```
← L 'ATT2='                --------- localisation d'une chaîne
→ ATT2 = ATTRP2 + ATTRP3
→ * OK :
```

Après les modifications, on vérifiera (compilation) et on passera à la simulation pour chacune des variables de la même façon que l'on a opéré pour le test du module.

Le scénario de base (variante 1) représentera l'existence d'une ville *relais* (agglomération 2) dans le réseau inter-urbain (figure III.15), les émigrations vont vers une métropole (agglomération 3) qui a une attractivité plus forte que celle de la ville relais . En conséquence on peut espérer que la population de la ville *relais*, au contraire des autres, descend jusqu'à un certain seuil, malgré une certaine hausse de son attractivité (ATTRP) ; d'autres variables suivront avec un décalage cette tendance, comme par exemple les emplois industriels (IND) ou le nombre de logements (LOG). En effet le nombre de logements doit plutôt se stabiliser, car les logements ne sont détruits qu'au bout d'une longue existence (et encore !) et dans la plupart des cas on les renouvelle, donc la baisse de la population ralentira la construction de logements.

Figure III.15



STRUCTURE DU RESEAU INTER-URBAIN
SELON LES SCENARIOS 1 ET 2

L'attractivité se distingue de la notion classique des modèles gravitaires inter-urbains ; ici l'attractivité rend compte des différents aspects qui forment *l'image perçue de la ville*, notamment à travers l'emploi et le logement. Par contre aucune, prise en compte explicite de la distance n'est faite.

La simulation confirme le scénario.
Passons directement à l'environnement de **visualisation** :

← TITLE $ EXISTENCE DE LA VILLE RELAI $;
→ * OK :
← OZOOM -2 POP2 = W, POP3 = X, POP4 = Y ;
→ * OK :
(figure III.16).

Si l'on veut visualiser la population au temps initial (1977) pour les 5 agglomérations on peut obtenir un histogramme par la commande :

← OHISTO 1977 POP1, POP2, POP3, POP4, POP5 ;
→ * OK :
(figure III.17).

Le logement de l'agglomération relais non seulement se stabilise du fait de la décroissance de la population, mais par la suite il suit le mouvement de la population.

← OZOOM-2 LOG1=V, LOG2=W, LOG3=X, LOG4=Y, LOG5=Z ;
→ * OK :
(figure III.18).

Figure III.16

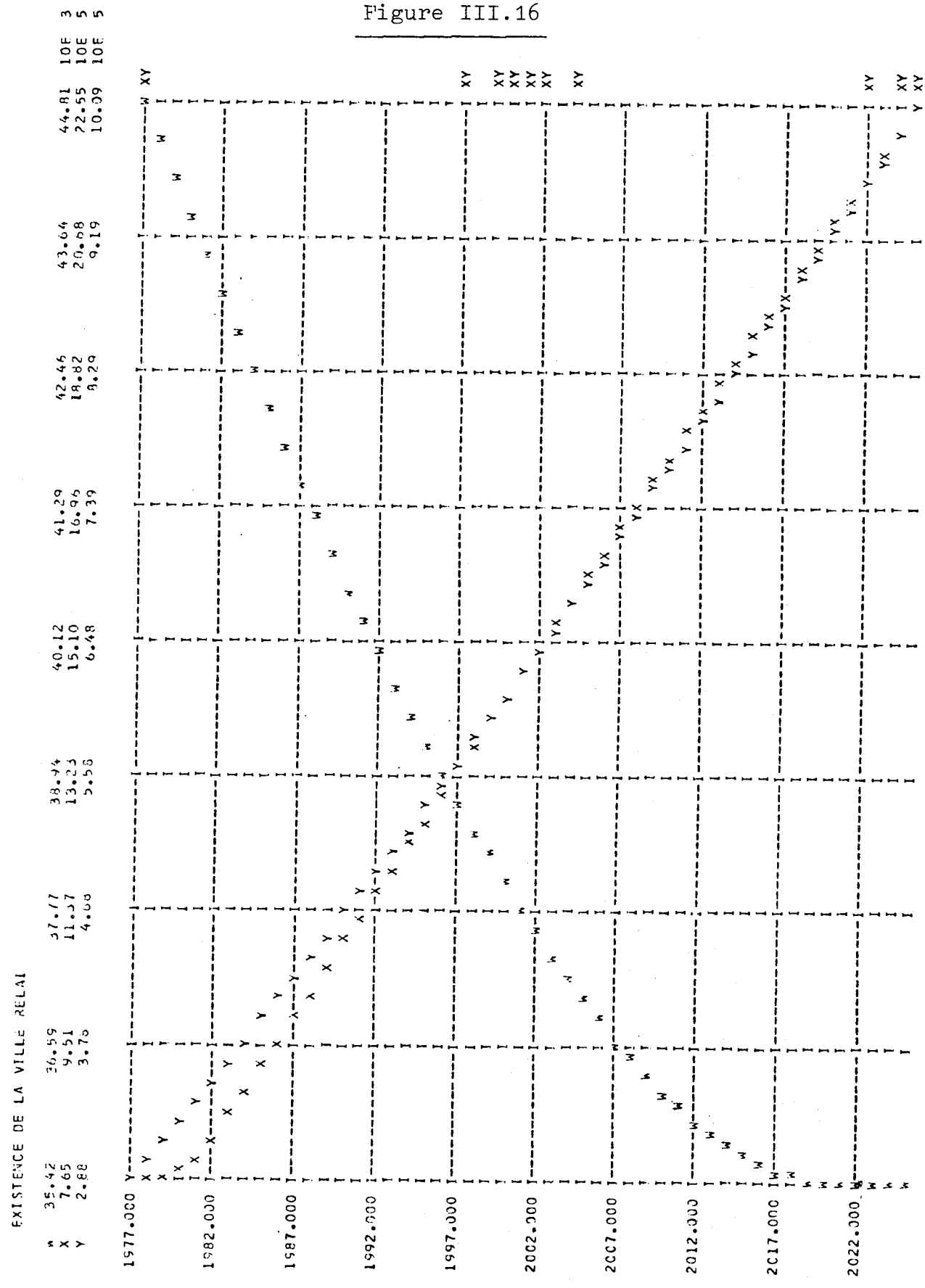


Figure III.17

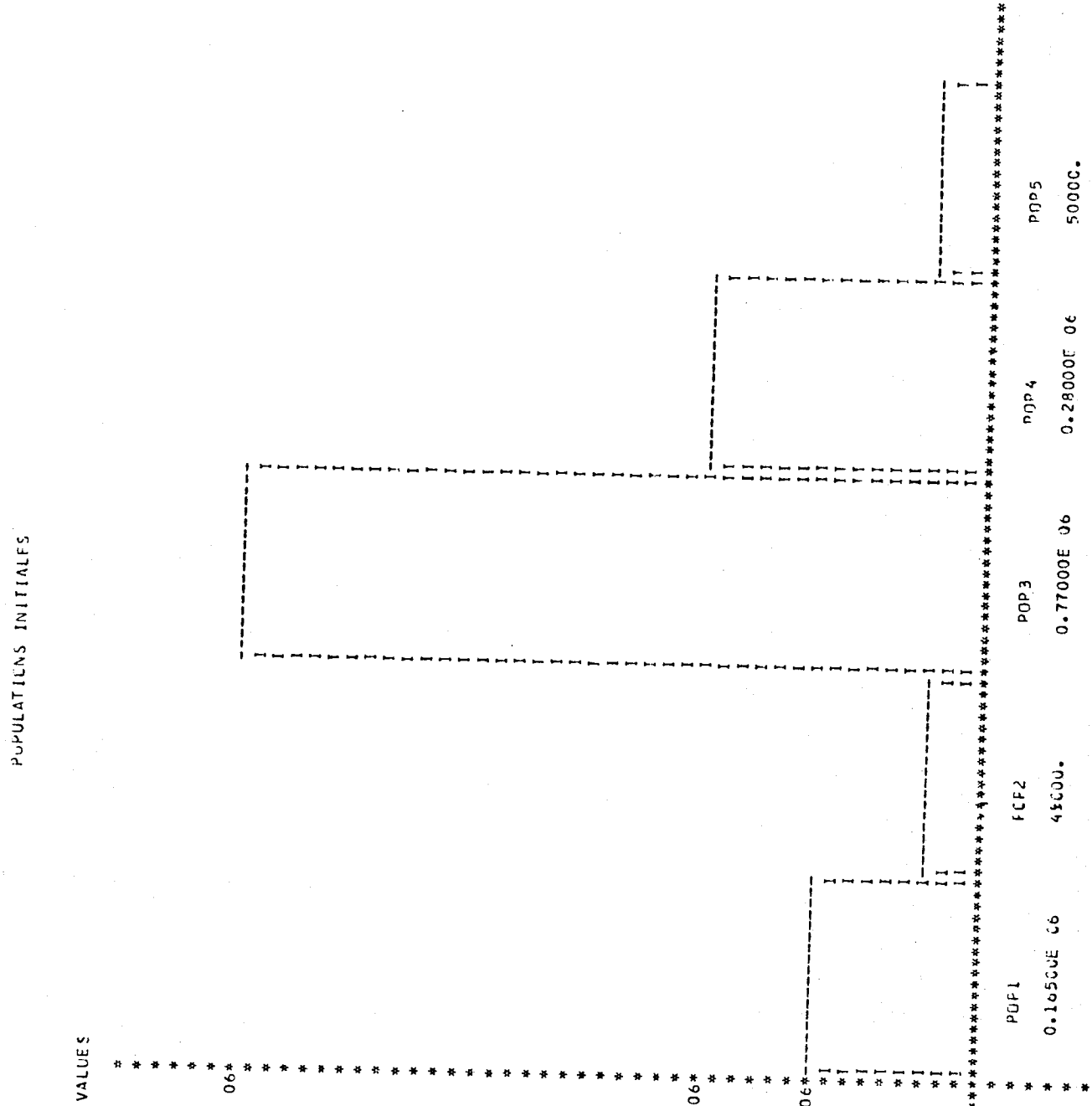
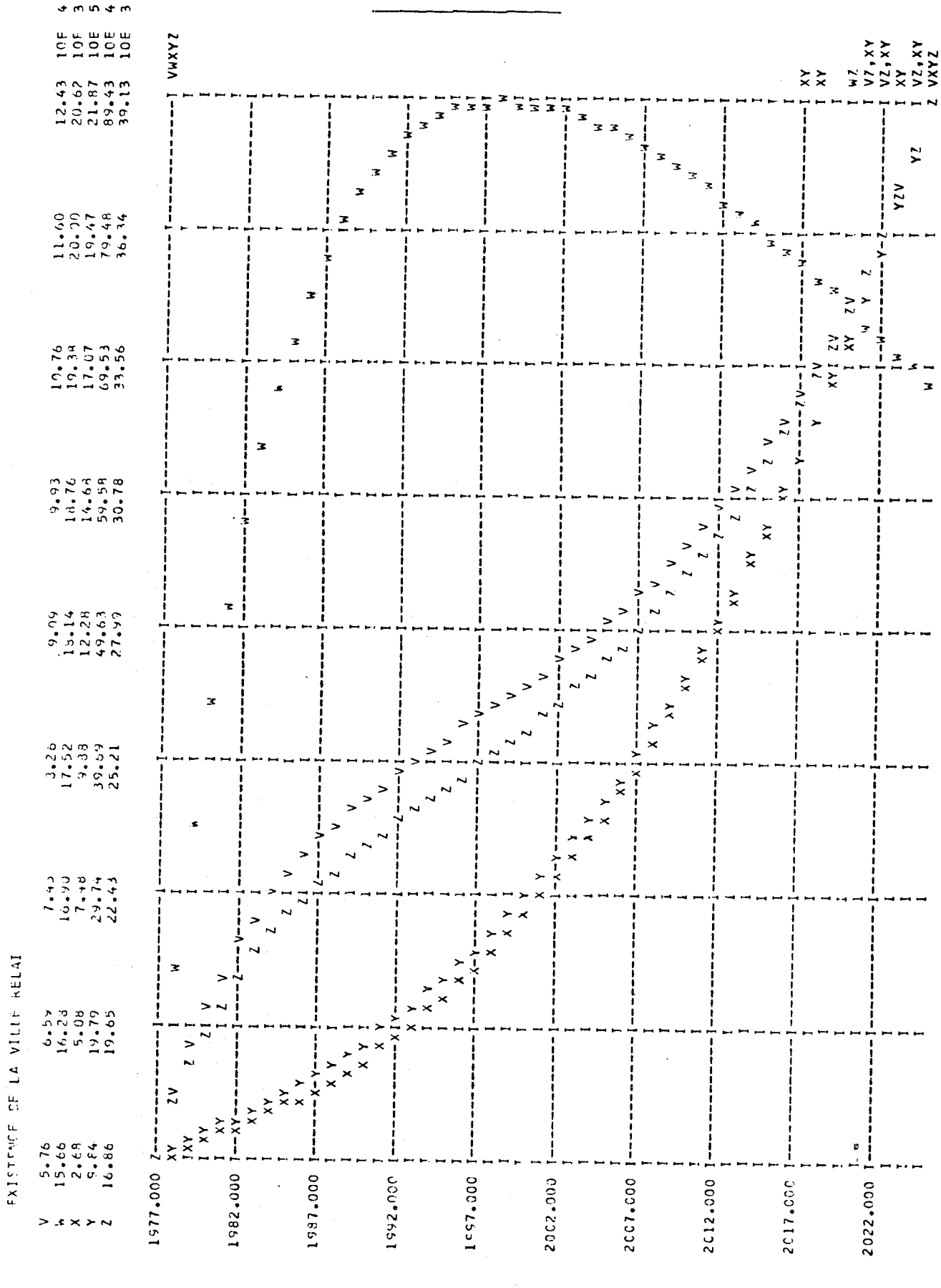


Figure III.18

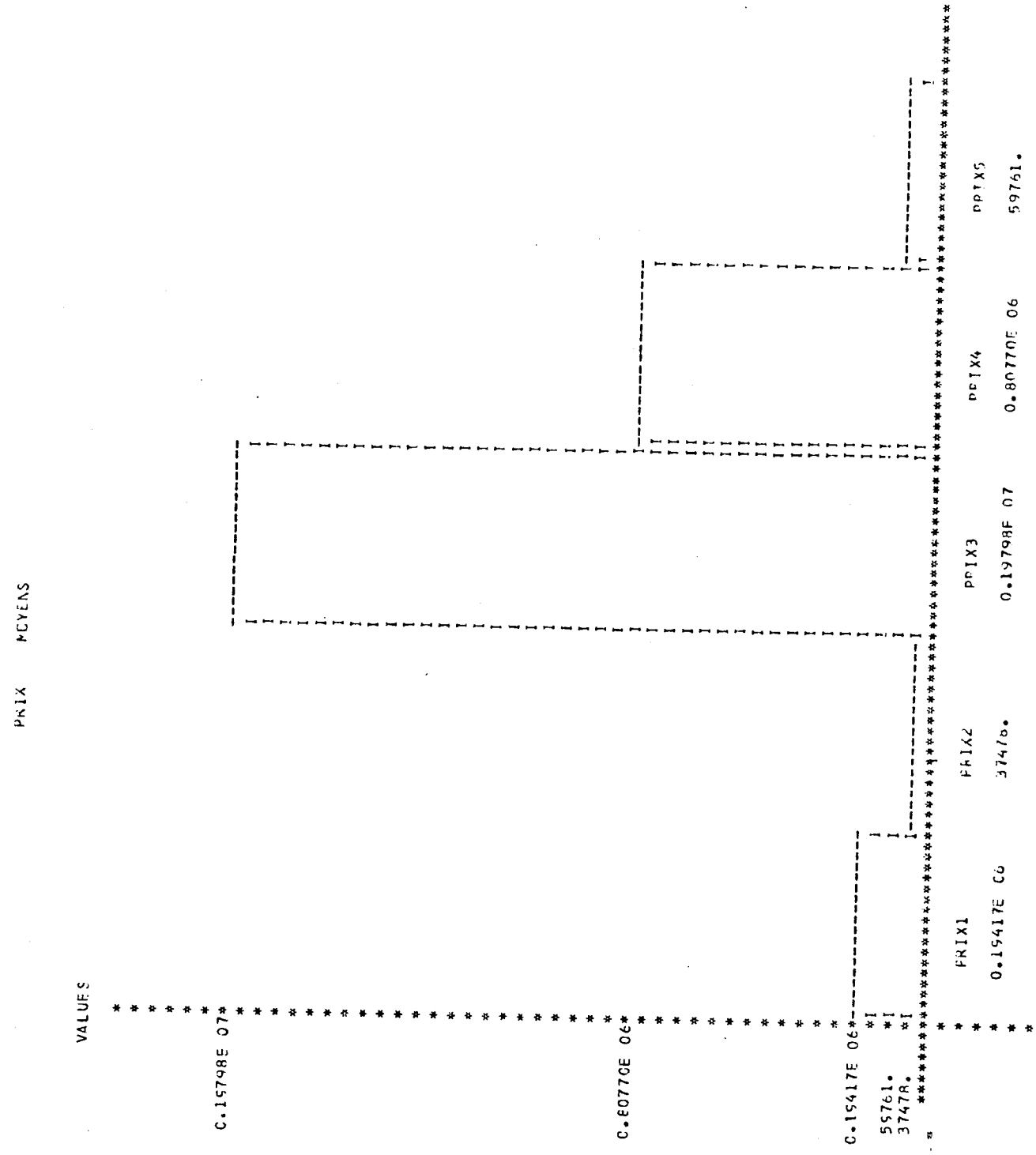


Une information intéressante peut être obtenue par exemple en visualisant les valeurs extrêmes prises par une variable (commande MINMAX) au cours de sa simulation ou bien la moyenne temporelle, numériquement (commande MEAN) ou sur graphique (commande HISTO MEAN) ; une illustration en est donnée ci-dessous pour le prix des terrains dans chacune des agglomérations

```
← MINMAX PRIX1, PRIX2, PRIX3, PRIX4, PRIX5 ;
→

XMIN : PRIX1 =  0.1650CE 06     XMAX : PRIX1 =  0.2767CE C6
XMIN : PRIX2 =   32143.         XMAX : PRIX2 =   45CCC.
XMIN : PRIX3 =  0.77CCCE 06     XMAX : PRIX3 =  0.4525CE C7
XMIN : PRIX4 =  0.28CCCE 06     XMAX : PRIX4 =  0.19C59E 07
XMIN : PRIX5 =   50CCC.         XMAX : PRIX5 =   87455.

→ * OK :
← MEAN PRIX1, PRIX2, PRIX3, PRIX4, PRIX5 ;

                XMEAN : PRIX1 =  0.1941?E 06
                XMEAN : PRIX2 =   37478.
                XMEAN : PRIX3 =  0.1979BE 07
                XMEAN : PRIX4 =  0.8C77DE 06
                XMEAN : PRIX5 =   59761.

→ * OK :
← OHISTO MEAN PRIX1, PRIX2, PRIX3, PRIX4, PRIX5 ;
→ * OK :
```

(figure III.19).

Figure III.19

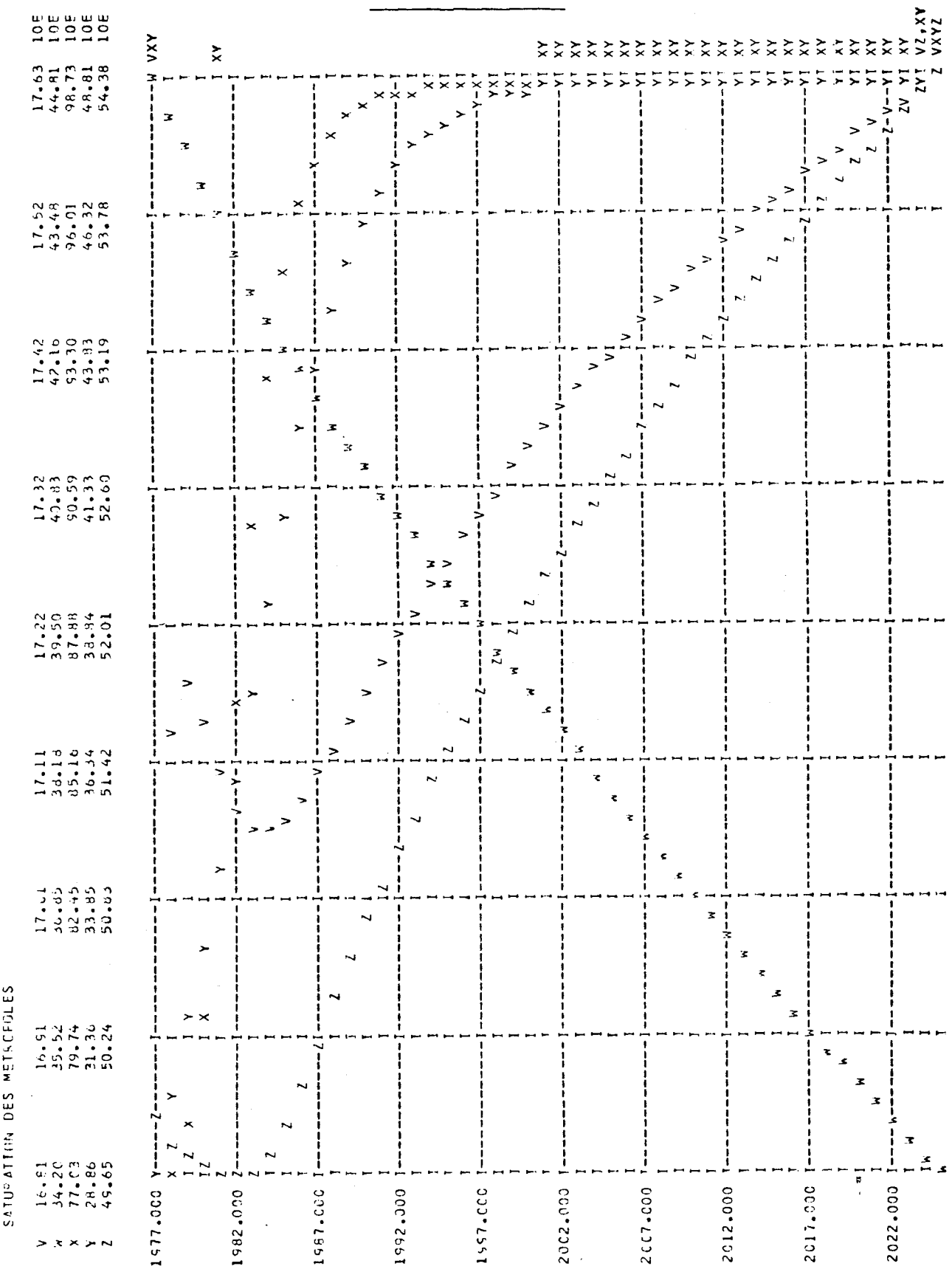


Le scénario 2 décrit l'intervention d'un phénomène de saturation démographique pour les métropoles ; la saturation est faite graduellement à partir de 100 000 habitants en-dessous du seuil théorique de saturation (1 million d'habitants pour l'agglomération 3, un demi-million pour l'agglomération 4). On a modélisé la saturation exclusivement à travers l'attractivité, c'est-à-dire en limitant la perception de l'attractivité vis-à-vis des migrants potentiels. On peut donc attendre la saturation de la population des deux métropoles et un ralentissement de la baisse de la population de la ville relais ; par contre dès que la saturation intervient les deux sous-métropoles retrouvent une forte croissance de la population

```
← OZOOM-2 POP1=V, POP2=W, POP3=X, POP4=Y, POP5=Z ;
→  * OK :
```

(figure III.20)

Figure III.20



Le scénario 3 montre un processus de *satellisation* de la ville relais. Le phénomène de saturation intervient comme dans le scénario précédent, mais contrairement au cas antérieur la régulation du système inter-urbain ne se fait pas au niveau du réseau, mais à travers le développement d'une liaison privilégiée entre la métropole saturée et la ville relais (voir figure III.21). En effet l'on suppose que lorsque la métropole se sature, l'*excès* de son attractivité vis-à-vis de l'extérieur est rapporté vers sa ville satellite qui bénéficie pour ainsi dire du *prestige* de la métropole, l'équilibre démographique de la ville relais peut être donc atteint au fur et à mesure qu'elle se satellise ; on peut même espérer que la ville satellite atteindra une croissance démographique type métropole.

STRUCTURE DU RESEAU INTER-URBAIN
SCENARIO 3 : SATELLISATION

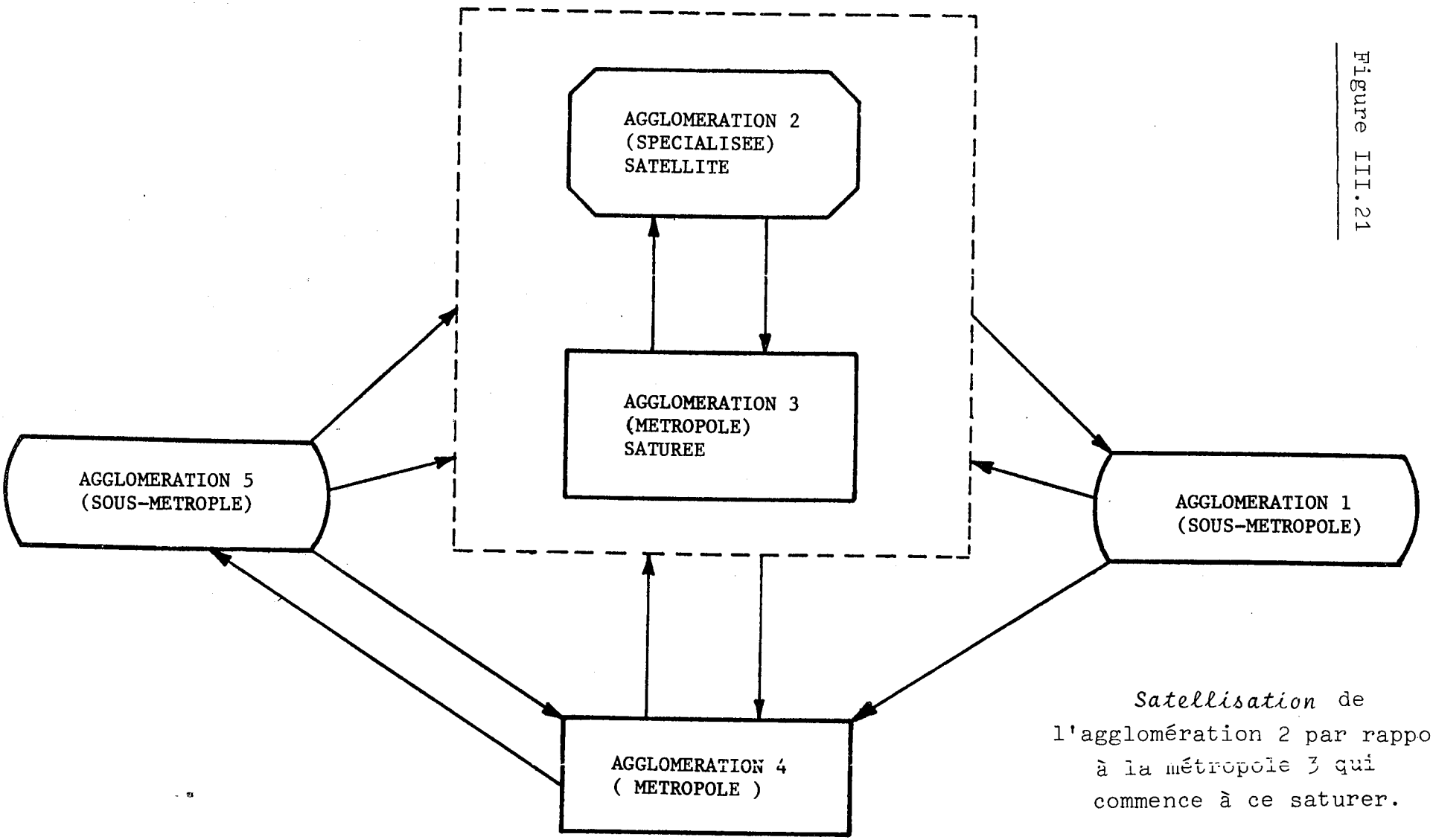


Figure III.21

Satellisation de l'agglomération 2 par rapport à la métropole 3 qui commence à ce saturer.

Dans l'environnement de visualisation, après l'avoir simulé

```
← TITLE $ SATELLISATION DE LA VILLE RELAI $
→ * OK :
← OZOOM-2 POP1=V, POP2=W, POP3=X, POP4=Y, POP5=Z ;
→ * OK :
```

(figure III.22)

```
← OZOOM-2 ATTRP1=A(0.,4.), ATTRP2=B(0.,4.0), ATTRP3=C(0.,4.),
          ATTRP4=D(0.,4.), ATTRP5=E(0.,4.) ;
→ * OK :
```

(figure III.23)

On peut revenir sur le scénario précédent pour regarder l'attractivité des différentes agglomérations

```
← OTHRUN 2 ;
→ * OK :
← TITLE $ SATURATION DES METROPOLES $ ;
→ * OK :
  OZOOM-2 UNIF ATTRP1=A, ATTRP2=B, ATTRP3=C, ATTRP4=D, ATTRP5=E ;
    OK :
```

(figure III.24)

Figure III.22

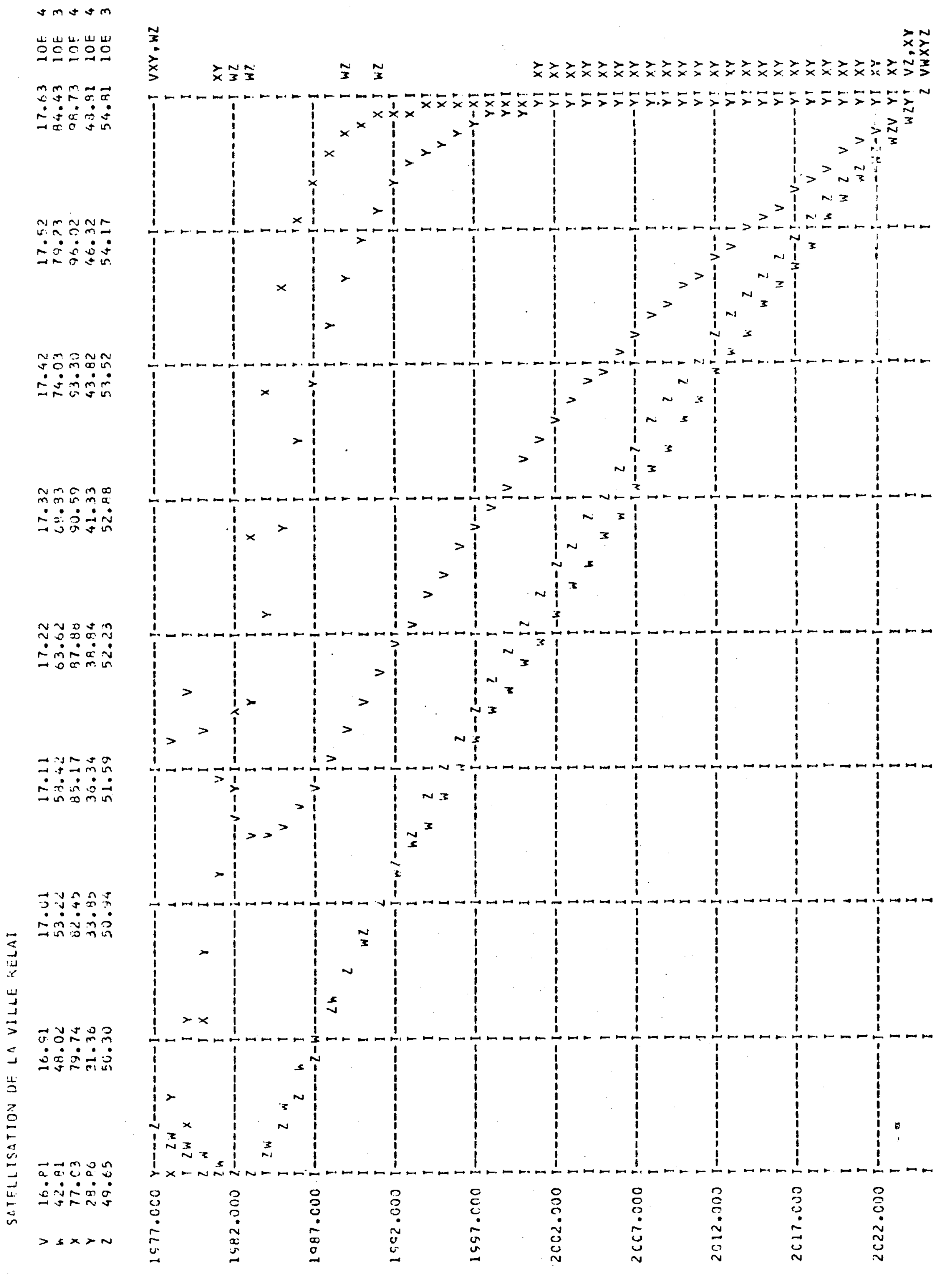


Figure III.23

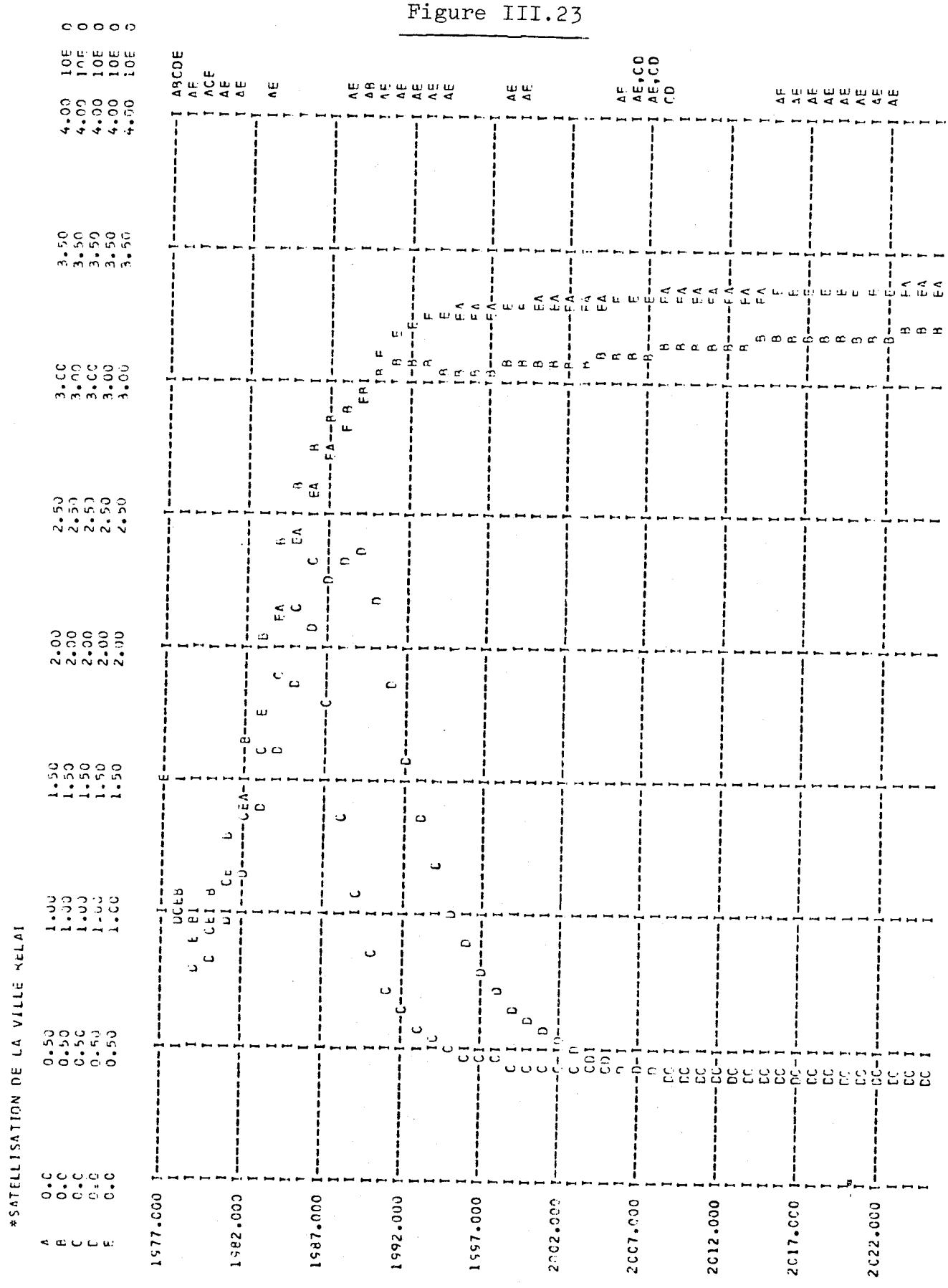
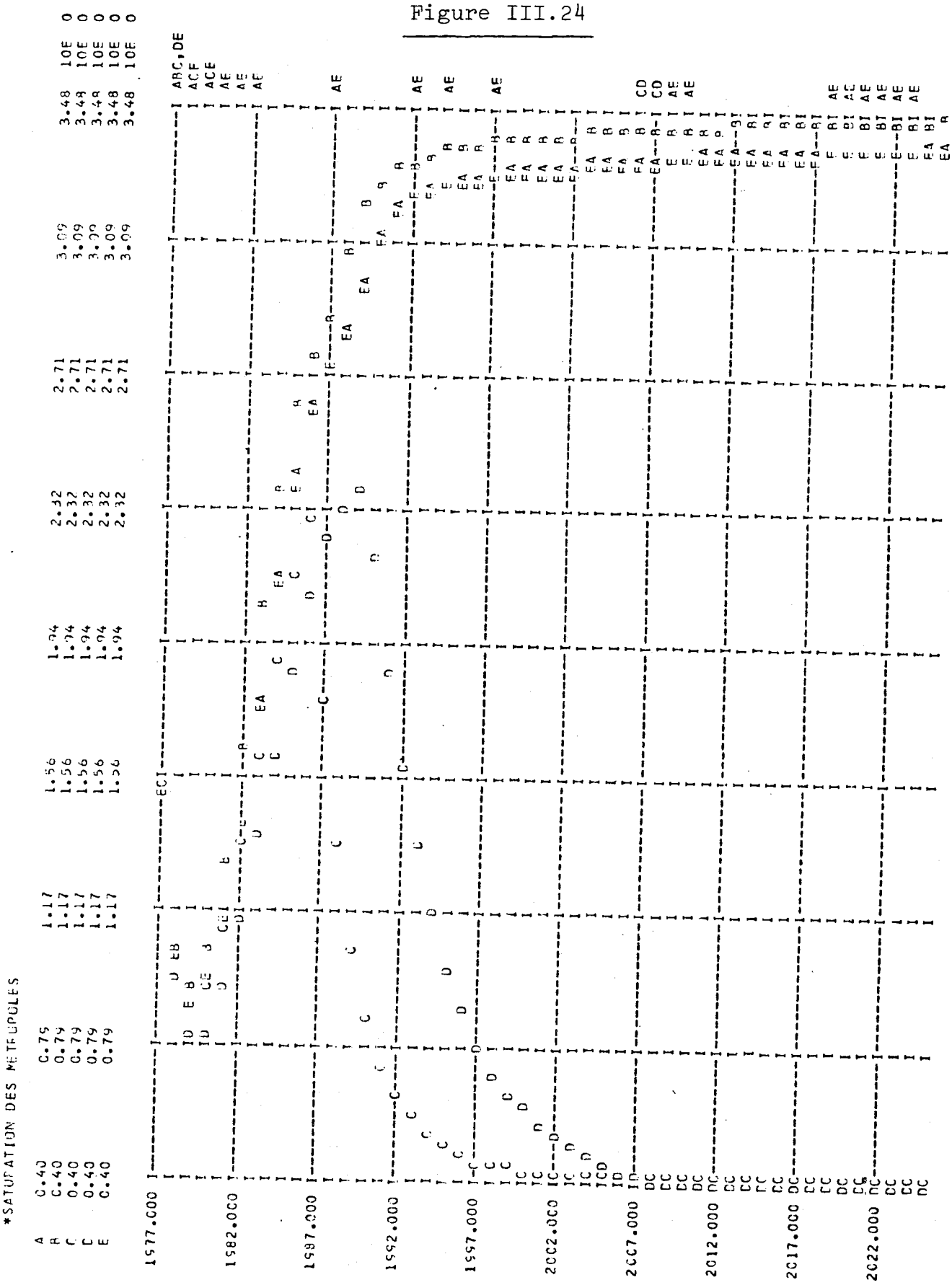


Figure III.24



Le quatrième scénario joue exclusivement sur le développement de nouveaux flux migratoires vers l'agglomération relais (figure III.29), ce qui peut correspondre à certaines politiques d'aménagement par la création directe de *pôles d'attraction* (par exemple : la création d'un centre universitaire). Ceci ouvre la voie à de nouveaux flux migratoires, car on permet à la dynamique interne de l'agglomération de se développer.

```
← TITLE $ EMERGENCE DE LA VILLE RELAI $ ;
→ * OK :
← FROM 1977 TO 1989 OZOOM-2, POP1=V, POP2=W, POP3=X, POP4=Y,
       POP5=Z ;
→ * OK : (figure III.25)
← FROM 1977 TO 1989 OPLOT  POP1=V, POP2=W, POP3=X, POP4=Y,
       POP5=Z ;
→ * OK :  (figure III.26)
← FROM 1977 TO 1989 OZOOM+2 POP1=V, POP2=W, POP3=X, POP4=Y,
       POP5=Z ;
→ * OK :  (figure III.27)
← FROM 1980 TO 2010 ZOOM-2 ATTRP1=A, ATTRP2=B, ATTRP3=C,
       ATTRP4=D, ATTRP5=E ;
→ * OK :  (figure III.28).
```

Comparaison des scénarios

```
← OCOMPAR ATTRP2  PLOT RUNS 1=V, 2=W, 3=X, 4=Y ;
→ * OK :  (figure III.30)


← OCOMPAR POP2 PLOT RUNS 1=V, 2=W, 3=X, 4=Y ;
→ * OK :  (figure III.31)
```

Figure III.25

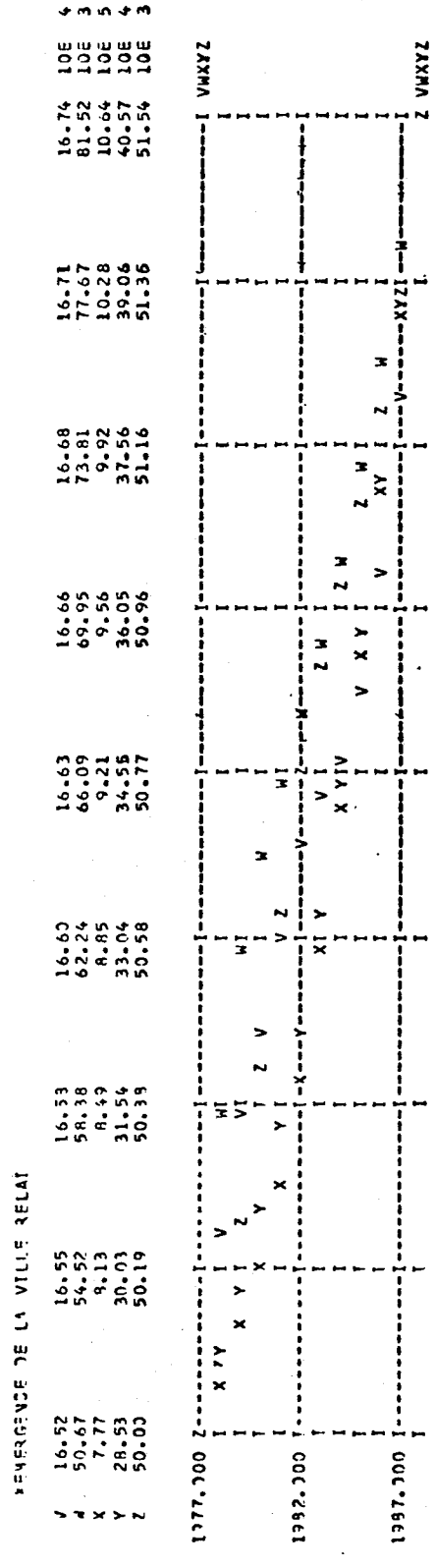
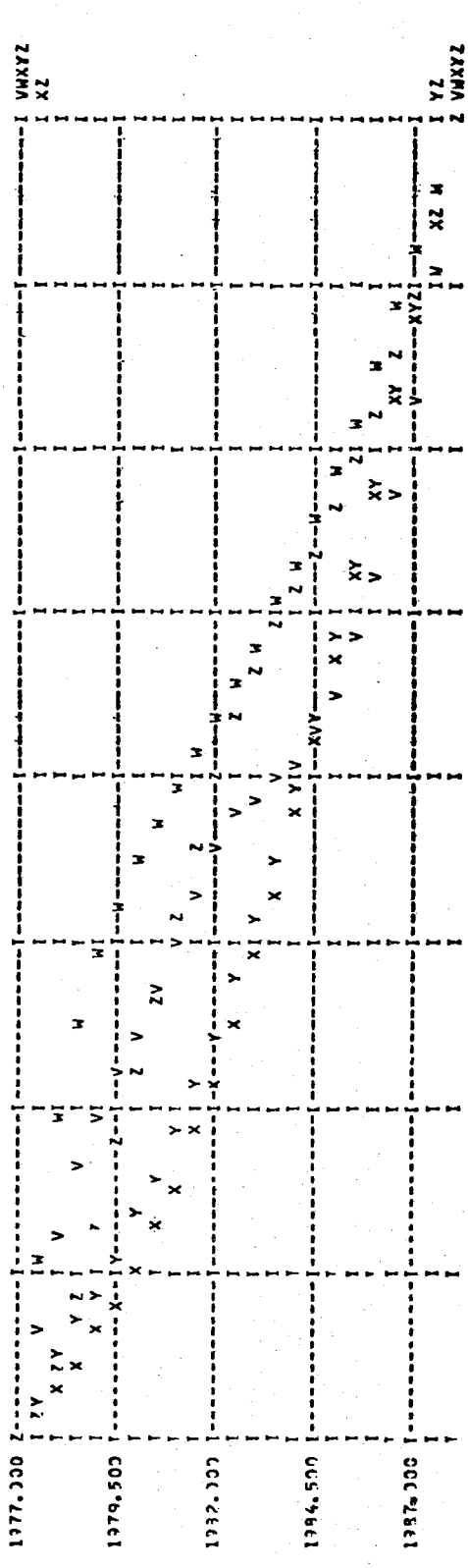


Figure III.26



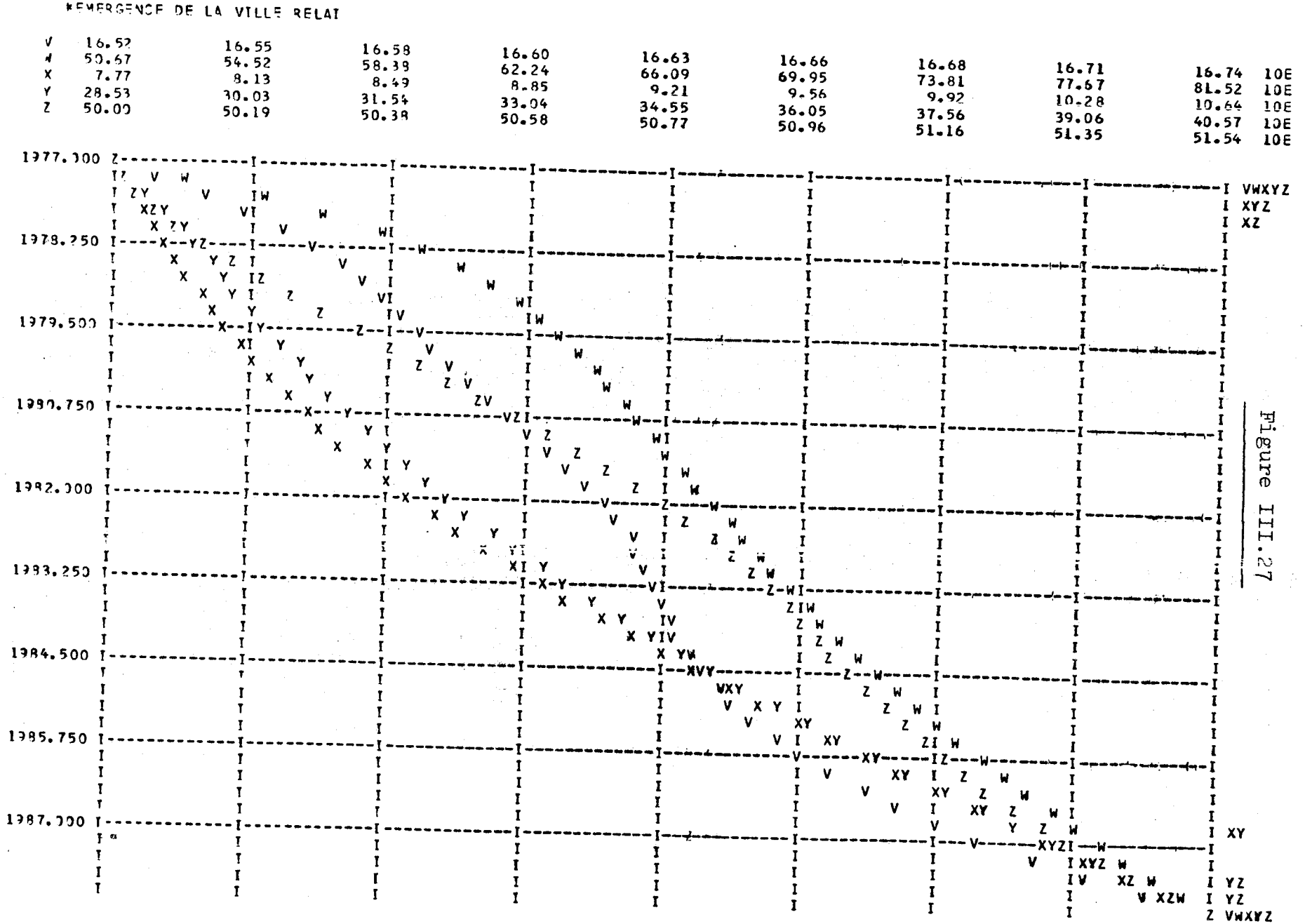


Figure III.27

*EMERGENCE DE LA VILLE RELAI

A	1.63	1.84	2.04	2.25	2.45	2.66	2.86	3.07	3.27	10E	0
B	0.56	0.68	0.79	0.90	1.02	1.13	1.24	1.36	1.47	10E	0
C	1.48	1.61	1.74	1.87	1.99	2.12	2.25	2.38	2.51	10E	0
D	1.38	1.52	1.65	1.78	1.92	2.05	2.18	2.32	2.45	10E	0
E	1.62	1.83	2.03	2.23	2.44	2.64	2.84	3.05	3.25	10E	0

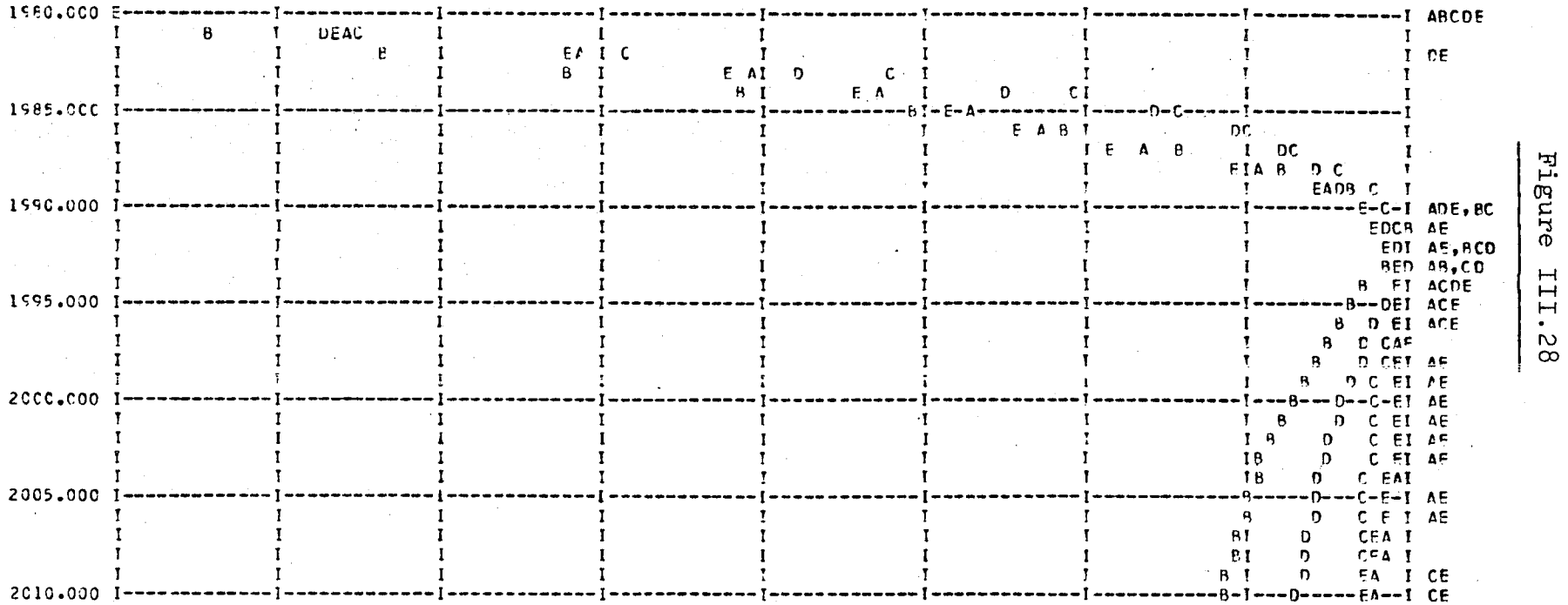
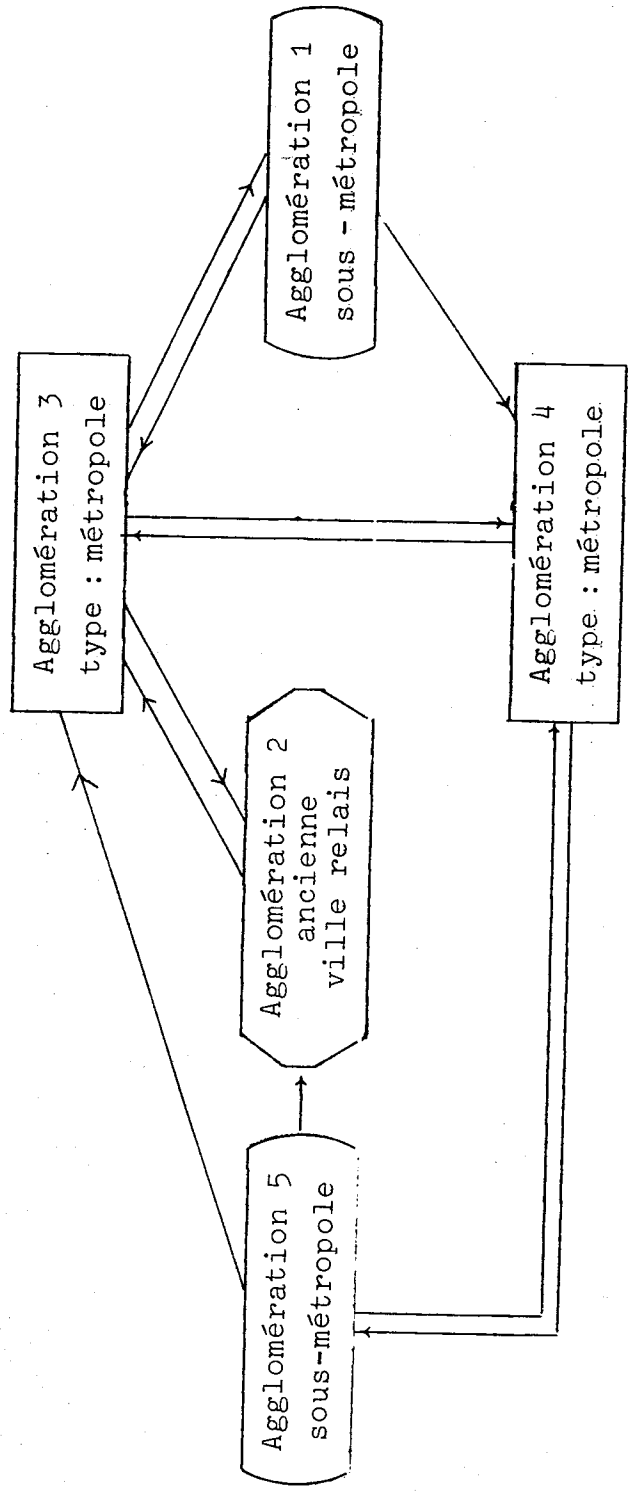


Figure III.28

Figure III.29



Emergence de la ville relais [5] par l'apparition des nouveaux flux migratoires qui accompagnent la saturation des métropoles (Scénario 4).

FIG. III. 30

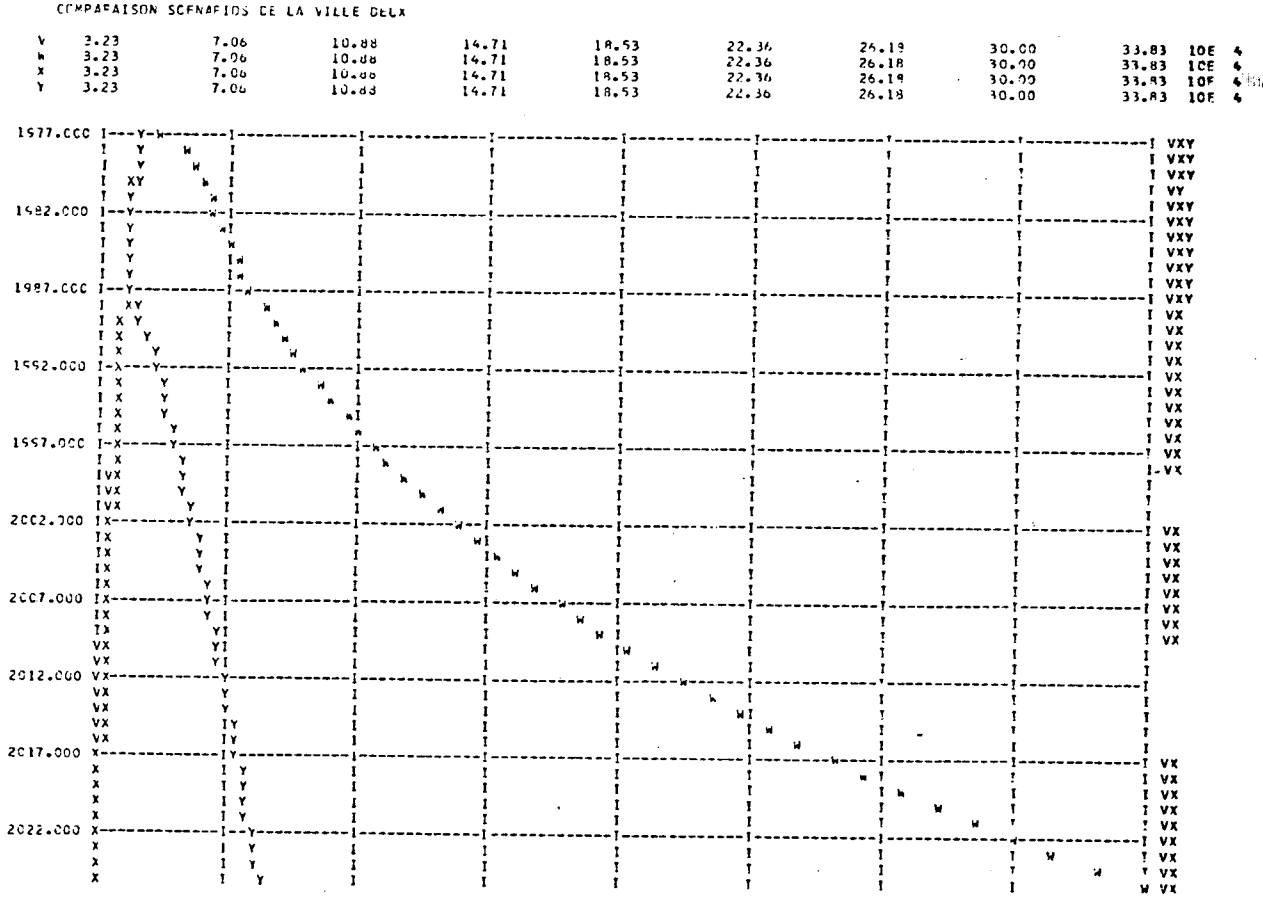
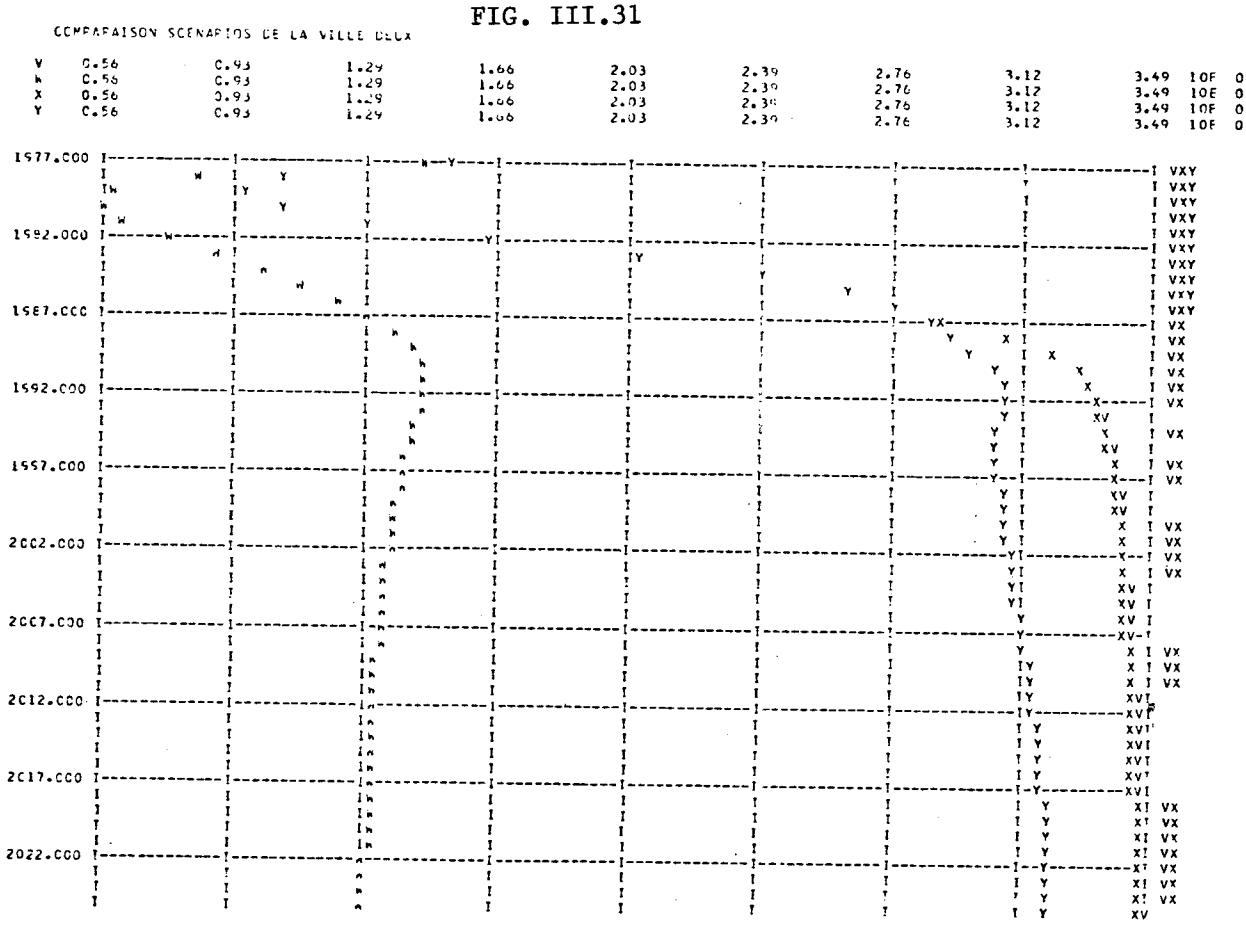
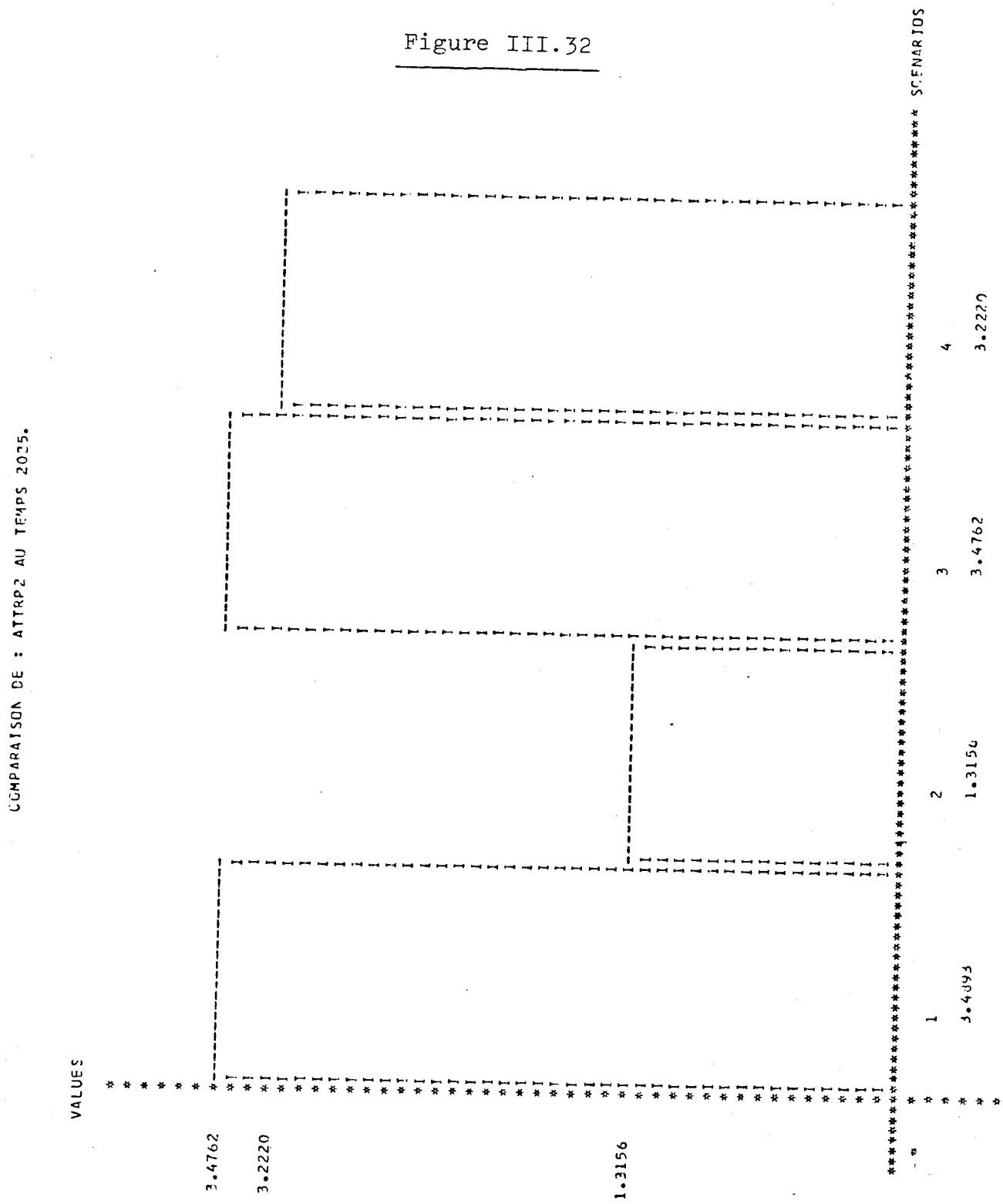


FIG. III.31



← OCOMPAR **ATTRP2** HISTO 2025 RUNS 1, 2, 3, 4 ;
→ * OK : (figure III.32)

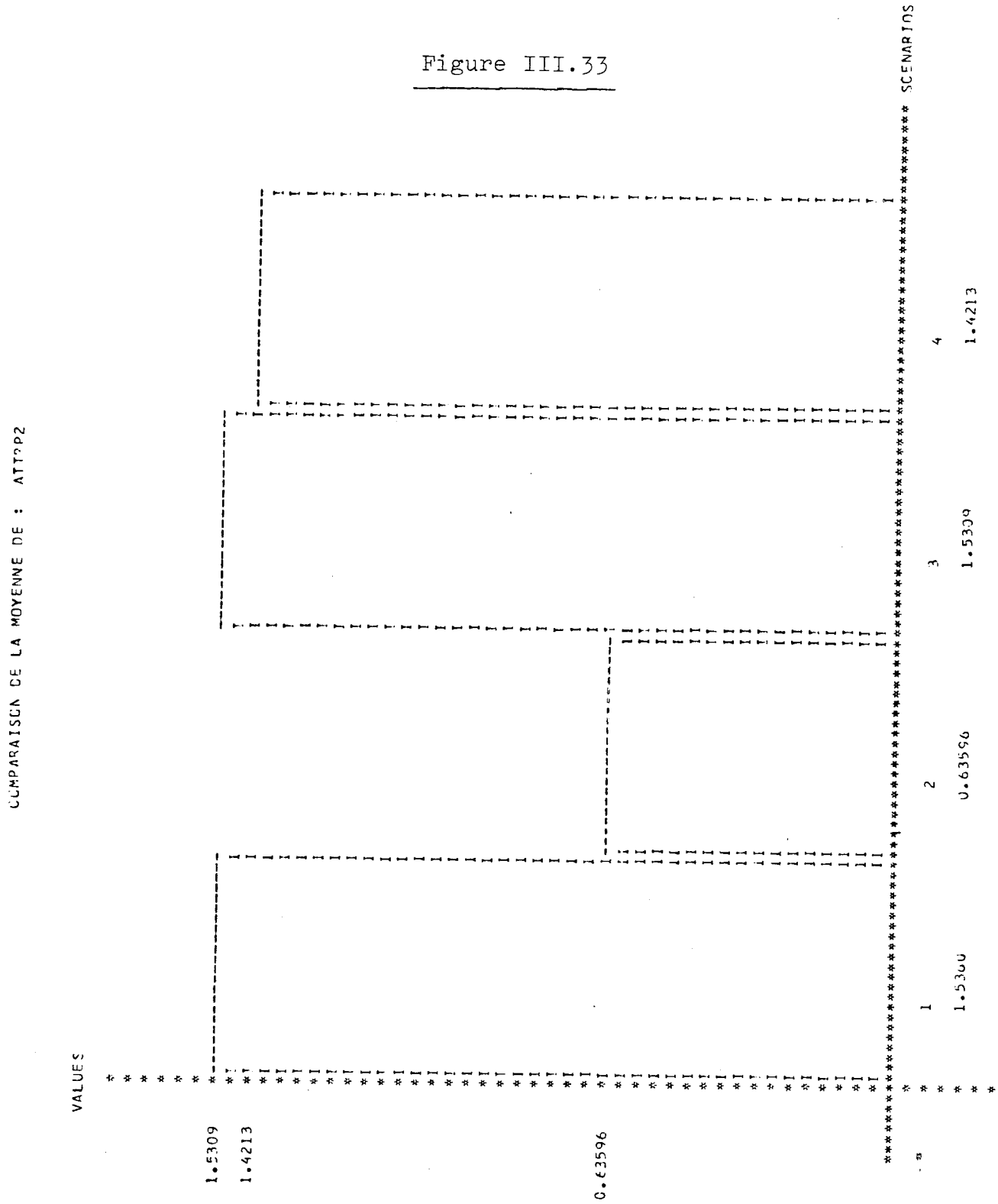
Figure III.32



← OCOMPAR **ATTRP2** HISTO MEAN RUNS 1, 2, 3, 4 ;

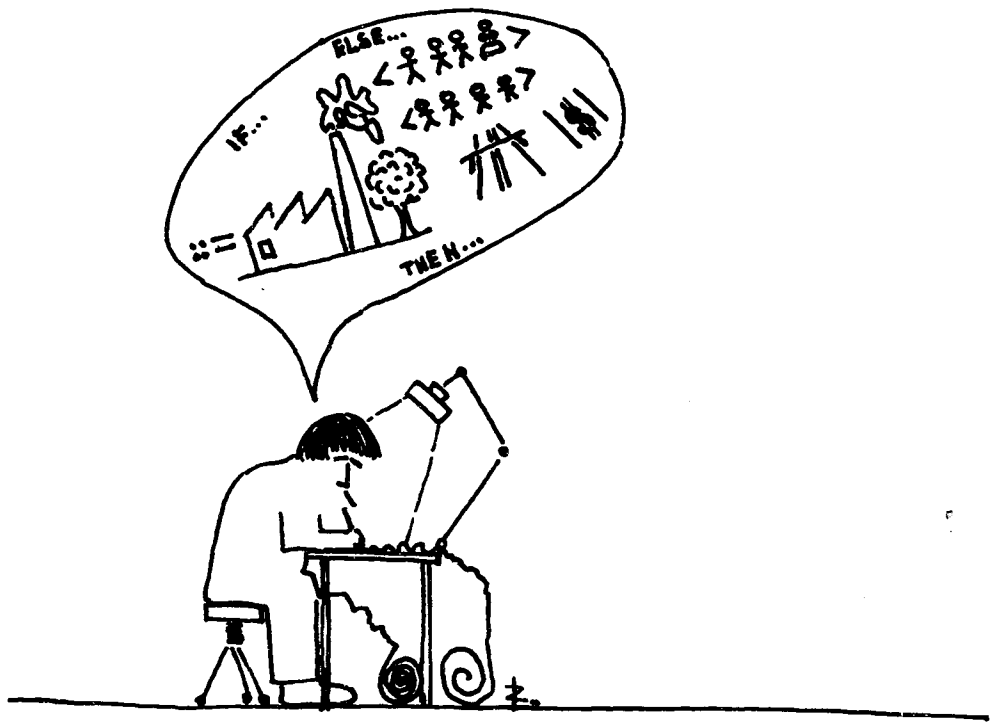
→ * OK : (figure III.33)

Figure III.33



← !	--------- passage à l'environnement d'information
→ * INFORMATION ENVIRONNEMENT	
→ * OK :	
← REPORT ;	--------- impression du dictionnaire
→ * OK : (voir Annexe V)	
← OPRINT VARIANT 3 ;	--------- impression de la variante 3
→ * OK : (voir Annexe V)	
← LIST RUN ;	--------- demande de la liste des produits de simulation existants
→ RUNS 1, 2, 3, 4	
→ * OK :	
← OUT ;	--------- sortie du système SAISYE
→ RETURN TO CMS	

Les différentes simulations que l'on a faites et dont seulement une **petite** partie a été montrée, ont permis de comprendre la dynamique de flux migratoires sous certaines hypothèses ; des améliorations découlent de cet apprentissage pour la construction du modèle que l'on s'est proposé d'étudier (23 villes). D'autre part il a permis de tester le système SAISYE et d'envisager des améliorations possibles.



L'étude du modèle continue !

BIBLIOGRAPHIE

[1] RIVERA E., UVIETTA P.

"Analyse et Modélisation des Systèmes"
Plaquette de Présentation au GRECO Rhône-Alpin d'Analyse de Systèmes - ENSIMAG - Juillet 1976

[2] ALFED L.E., GREHAM A.K.

"Introduction to Urban Dynamics"
Wright-Allen Press, Cambridge
Mars, 1976

[3] MARKLAND R.E., GRANDSTAFF

"Modeling demographic employement interactions in an urban economy"
SIMULATION, pp.33-43, February 1975

[4] UVIETTA P.

"Migration Systems : a multi-level model"
8ème Congrès international de Cybernétique,
Namur, 6-10 septembre 1976

[5] G.R.A.S.S.E.
(Groupe de Recherche et d'Analyse des Systèmes Spatio-Economiques)

"Un modèle de dynamique de l'emploi dans un système régional-urbain"
Actes du 5ème Colloque sur les méthodes mathématiques appliquées à la géographie - Octobre 1976
Cahiers de Géographie de Besançon (à paraître)

[6] MEADOWS et al.

"Dynamics of Growth in a Finite world"
Wright-Allen Press,
Cambridge Mass. 1974

[7] MEADOWS et al.

"Halte à la Croissance ?"
Paris, Fayard - 1972

[8] RECHENMANN F.

"Analyse et Modélisation descendantes des systèmes socio-économiques"
Thèse Docteur-Ingénieur - INPG - Juillet 1976

[9] FORRESTER J.W.

"Industrial Dynamics"
M.I.T. Press 1961

[10] RIVERA E.

"Review of delays in dynamic system simulation models"
Proceedings of Informatica 76, pp.5-104, Bled, Yugoslavie
Octobre 1976

[11] MOHAPATRA P.K.

"Some generalised results on exponential delays"
Trans. IMACS, Vol. XVIII n°4, Octobre 1976.

CHAPITRE IV

RÉALISATION PROGRAMMÉE

TABLE DES MATIERES

1 . LE LANGAGE DE COMMANDE

L'objet de ce chapitre n'est pas de décrire la syntaxe du langage de commande (voir Annexe 1), ni de faire un manuel d'utilisation de celui-ci, mais d'exposer les principales options de sa programmation et mise en oeuvre.

La syntaxe d'un langage de commande doit être très simple et sa traduction sémantique directe car "un langage de commande n'est pas un outil primaire comme les langages de programmation qui sont utilisés pour l'écriture des algorithmes de résolution de problèmes, mais il est un outil secondaire pour exécuter les programmes correctement" [1].

Il existe différentes structures de langages de commande qui partent de la structure classique la plus simple :

commande paramètre 1, ..., paramètre n

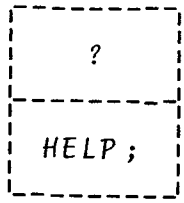
Dans cette structure on introduit parfois soit des paramètres groupés par des parenthèses, soit des modificateurs (clefs qui identifient le paramètre en désordre). Pour de tel types de représentation on peut même penser à un métalangage pour décrire et mettre en oeuvre ces langages [2]. Si la commande a une signification unique et si le nombre et la nature des paramètres sont fixes (par exemple s'il n'y a pas de liste des paramètres, ni de paramètres par défaut), alors ce métalangage est concevable. Dès que l'on veut introduire des éléments de souplesse dans le langage, il est plus convenable de le décrire à travers une grammaire plus évoluée à l'aide de règles syntaxiques pour chaque paramètre.

Le problème de l'ordre des paramètres peut être en grande partie évité, si les commandes longues sont décomposées en plusieurs commandes, de telle sorte que les premières commandes créent le contexte des suivantes ; ceci permet de ne pas répéter à tout moment (à chaque commande) soit l'objet soit le type d'opération que l'on est en train d'effectuer.

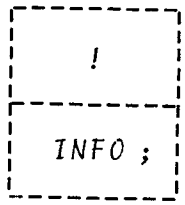
Par exemple dans l'environnement d'entrée et information on précise le modèle avec lequel on va travailler ; il n'est pas nécessaire de renouveler cette précision dans tous les environnements sauf si l'on change de modèle par la même commande.

On a gardé certaines redondances dans le langage de commande qui donnent la possibilité de se rappeler d'une commande ou bien d'identifier la même commande sous des aspects différents. Ainsi par exemple les commandes *?* et *HELP* ont le même effet, de même les commandes *!* et *INFO*.

Commandes communes à tous les environnements :



Ces commandes offrent un guide à l'utilisateur en lui permettant de connaître la liste de commandes disponibles dans l'environnement dans lequel on se trouve.



Lorsque l'utilisateur a fini le travail dans la session, ou lorsqu'il désire une information au niveau général, ou simplement s'il se trouve perdu dans l'utilisation du système, il peut revenir à l'environnement d'information. Si cette commande est émise dans l'environnement d'entrée, on reste dans cet environnement et un message, qui indique comment passer à l'environnement d'information, apparaît.

La notion "listes de paramètres" d'un même type dans une commande est très utile pour effectuer la même opération sur plusieurs objets ou éléments d'un même objet.

Le recours aux paramètres par défaut permet de simplifier beaucoup de commandes. Si l'on veut, par exemple, imprimer seulement une partie de l'évolution de certaines variables on fait

FROM <date> TO <date> PRINT <liste de variables> ;

par contre si on ne spécifie pas les dates bornes d'impression, on prend par défaut toute l'évolution de la simulation :

PRINT <liste de variables> ;

Les séparateurs obligatoires entre unités syntaxiques sont gênants. On a donc adopté la règle suivante : **le blanc ou la virgule** sont **séparateurs obligatoires d'unités syntaxiques du même type, sinon ils sont optionnels.**

Ainsi par exemple :

```
PLOT POP=P(10000,50000) , LOG=L ;
PLOT POP = P  (10000 50000 )LOG = L;
```

ont le même effet.

Afin de pouvoir écrire plusieurs commandes dans une même ligne on a introduit un **séparateur de fin de commande, le point virgule** ; . L'usage de ce point virgule permet également d'écrire une même commande sur plusieurs lignes (car l'entité commande est alors définie entre deux points virgules).

Le point d'exclamation (retour à l'environnement d'information) et le point d'interrogation (liste de commandes de l'environnement) sont les seuls à ne pas se terminer par un ; .

En conséquence **la ligne blanche** (ou **double retour chariot**) ne produit pas d'erreur syntaxique, elle produit simplement **l'impression du nom de l'environnement** où l'utilisateur se trouve et celle **du nom du modèle** sur lequel il travaille.

Si la ligne blanche est produite au milieu d'une commande, alors l'utilisateur peut continuer à écrire sa commande après l'impression de ces derniers messages.

Après l'exécution de chaque commande un message "OK:" va être imprimé, il permet à l'utilisateur de repérer où il en est dans l'exécution de ces commandes. Si l'utilisateur a mis n commandes dans la ligne, alors le n-ième message OK: lui indique que le système est en attente de commande.

Les erreurs syntaxiques sont détectées par le système. Un message indiquant la nature de l'erreur est imprimé, l'utilisateur peut aussitôt recommencer à écrire sa commande (il n'y a pas de message OK:). Evidemment si l'utilisateur a émis plusieurs commandes dans la même ligne, et qu'une erreur syntaxique apparaît la commande en question n'est pas exécutée, on imprime le message d'erreur et on continue à exécuter les commandes suivantes.

2 . L'ANALYSE SYNTAXIQUE ET LA PARTIE CONTROLE

"Les problèmes vraiment difficiles dans le développement de langages de programmation sont ceux de représentation, structuration de données, et allocation de mémoire. Il est clair que ces problèmes ne seront pas résolus par les seules méthodes syntaxiques" [3] B.M. LEAVENWORTH.

Le langage de commande n'est certes pas un langage de programmation ; mais les vrais problèmes sont ceux de la gestion du système, de la gestion des objets et de l'allocation mémoire ; et les méthodes syntaxiques ne les résolvent pas.

L'analyse syntaxique n'est donc qu'une partie limitée (mais nécessaire) de l'interpréteur. Néanmoins un bon choix pour l'analyseur syntaxique permet d'avoir souplesse et rapidité dans la partie contrôle.

On a vu que les commandes à elles seules ne déclenchent pas **une action** ; cette action **dépend** aussi de l'**environnement** et du **contexte** (état du système, communication entre commandes). C'est donc l'interpréteur qui doit gérer l'analyse syntaxique et non l'inverse.

Le langage de commande est extrêmement simple, chaque commande peut être décrite par une grammaire de type 3 [4], ou par un automate d'états finis, le transit entre états assure l'exécution des fonctions sémantiques (par exemple appel de sous-programmes avec passage de paramètres). Il existe à l'heure actuelle des générateurs d'analyseurs de langages de commande comme ESPACE [5], [6] construits sur cette idée ; l'interpréteur est dirigé dans ces cas par l'analyseur syntaxique ; et donc le changement d'environnement, la gestion de l'état du système et la communication entre commandes exigent à chaque commande des échanges avec la mémoire secondaire.

Par ailleurs ces analyseurs "tout-faits" véhiculent des contraintes de transportabilité, encombrement mémoire etc... et des restrictions sur les formats syntaxiques et contenu lexicographiques.

Le choix fait pour l'analyseur syntaxique s'est donc porté sur un outil plus fondamental existant à l'Université de Grenoble : le transformateur de grammaire de Griffiths-Peltier [7], [8]. Bien que conçu pour des langages plus généraux (type LL1) il génère un analyseur syntaxique efficace surtout si la plupart des règles et alternatives de la grammaire commencent par un symbole terminal [9]. Ce transformateur de grammaire est utilisé pour générer l'analyseur correspondant à chaque environnement, un programme FORTRAN codifie les différents appels de macros dans une chaîne interprétable d'entiers. Ce sont ces chaînes numériques interprétables de faible encombrement mémoire qui seront lues au début d'une session et validées au fur et à mesure que l'on change d'environnement. Ceci permet d'avoir la même structure d'analyseur pour tous les environnements, en particulier le même pré-processeur (ou analyseur lexicographique), le même détecteur d'erreurs et le même sytème d'appel et sauvegarde de fonctions sémantiques.

Un environnement est considéré comme **actif** si à l'instant considéré l'interpréteur a comme grammaire de contrôle (la chaîne numérique qui dirige l'analyse) la grammaire de l'environnement en question ; d'où une exclusion mutuelle entre environnements. Au moment de l'activation du système la grammaire est celle de l'environnement d'entrée. Un tableau de mots-clès est associé à chaque environnement et validé à chaque changement. Ceci permet en outre de changer l'apparence du métalangage (qui actuellement est de l'anglais) avec un minimum d'effort, en changeant seulement les tableaux de mots-clès de chaque environnement ainsi que le tableau contenant les divers messages.

C'est l'analyseur lexicographique qui demande à l'utilisateur une ligne de façon à permettre la continuation d'une même commande ou d'une unité syntaxique d'une commande sur la ligne suivante.

Une erreur syntaxique empêche l'exécution de la commande. Or la seule façon de s'assurer que l'on n'a pas d'erreur syntaxique est de faire l'analyse de la commande toute entière. C'est pourquoi l'interpréteur, dès qu'il rencontre une fonction sémantique, crée un code intermédiaire et le mémorise dans une zone de travail ; ce code assurera l'exécution de la commande dès la rencontre du ";". En conséquence l'interprétation est au niveau instruction et non pas au niveau mot.

Ces dernières considérations permettent de classifier les fonctions sémantiques employées en trois types :

- mémorisables : celles qui sont codifiées (la grande majorité)
- contrôle de mémorisation : par exemple celles qui indiquent le type de variable (FORTRAN) qui va être mémorisé par la suite
- directement exécutable : la fin de commande.

COMMANDES DE CHANGEMENT D'ENVIRONNEMENT

```
MODEL <nom model> ;
```

Spécification du modèle et passage de l'environnement d'entrée à celui d'**Information.**

```
CHECK ;
```

Passage à l'environnement de **Vérification** ; cette commande existe dans les environnements d'Information, de Construction, de Mise à jour, et de Visualisation.

```
DISPLAY RUN <identification simulation> ;
```

Cette commande de l 'environnement d'Information permet le passage à l'environnement de **Visualisation et Analyse** si et seulement si existe un objet produit de simulation ayant le même identificateur que la commande.

```
U P D <objet à mettre à jour> ;
```

Cette commande existant dans les environnements d'Information, Construction, Mise à jour, Vérification, et Visualisation permet le passage à l'environnement de **Mise à jour**, en spécifiant l'objet (cf. § 7).

```
BUILD ;
```

Passage à l'environnement de **Construction**. Cette commande existe dans les environnements d'information, mise à jour et vérification.

```
GO <variante à simuler> <jeu de données à simuler> ;
```

Cette commande existe uniquement dans l'environnement de Vérification et permet le passage à l'environnement de **Simulation** en spécifiant les deux objets sur lesquels la simulation va porter. Ce passage est possible si et seulement si les deux objets,variante et jeu de données,ont été vérifiés correctement.

```
FROM <date> TO <date> RUN <identificateur optionnel> ;
```

Cette commande exécute la simulation et fait le passage à l'environnement de **Visualisation et Analyse.** Elle se trouve seulement dans l'environnement de simulation (cf. § 8).

3 . LA GESTION MEMOIRE : LA PAGINATION, LE RAMASSE-MIETTES

Un des buts proposés était de pouvoir traiter des modèles et jeux de données de taille très grande. Les systèmes de simulation existants sont toujours très limités de ce point de vue.

La mémoire est en partie occupée par le système de simulation, de plus s'il existe un pré-compilateur (qui occupe une place non négligeable), alors la zone libre pour les objets est réduite (mais cela dépend étroitement de la machine). Ceci est d'autant plus vrai qu'en FORTRAN la place mémoire est réservée statiquement, on peut donc arriver facilement à la saturer.

Une solution consiste à diviser les objets encombrants (modèle, variantes, documentation, jeux de données) en zones de taille fixe (dûe à l'allocation statique) appelées "pages". Cela va permettre de faire toutes les manipulations simples : construction, mise à jour (que l'on fait en mémoire principale) page par page et non sur l'objet tout entier.

Une "page" correspond à un fichier de mémoire secondaire. Le mécanisme appelé "pagination" n'est donc ici qu'un système de gestion de certains fichiers de la mémoire secondaire, ce n'est pas la pagination classique. Le mécanisme proposé n'est pas non plus une segmentation [10] car les pages ne sont pas des segments (sous-programmes logiquement indépendants) et ce n'est pas la méthode de programmation qui découpe les pages mais le système de façon automatique et aveugle. Ce mécanisme est transparent à l'utilisateur.

Cette *pagination* est utilisée dans trois cas :

- S'il existe un **pré-compilateur** du langage de description de type incrémentiel ; le compilateur peut lire page par page le modèle et les jeux de données.

- Lors de la **construction** les modules de la bibliothèque de même que les lots (images de cartes perforées) sont ajoutés à la suite dans la dernière page de l'objet correspondant.

- Pour la **mise à jour** le problème est un peu plus compliqué et justifie l'emploi de pages à la place de toute autre solution, comme l'emploi de tableaux unidimensionnels pour l'allocation dynamique en FORTRAN. En effet l'élément de travail de l'environnement de mise à jour est la ligne d'un objet.

La ligne adressée peut être située partout dans le modèle et il existe une probabilité très grande pour que la ligne suivante adressée soit voisine de l'antérieure.

Pour minimiser le nombre de changements de page en mémoire (opération relativement coûteuse en FORTRAN) lors de l'insertion d'une ligne dans le texte d'un objet, on est amené à se définir une zone de débordement de taille fixe à la suite de chaque page.

Le nombre de pages, leur taille, ainsi que la taille de la zone de débordement sont des paramètres inter-reliés. Un bon choix est fonction des besoins, c'est-à-dire de la taille moyenne des objets (modèle, module, etc...) que l'on utilise. Une façon de faire ce choix est présentée dans l'annexe II.

Dans l'environnement de mise à jour, quand on efface une ligne (donc une ligne d'une page), on tasse les lignes suivantes de façon à laisser de l'espace libre seulement à la fin de chaque page. Il se peut que lors de la construction on ait besoin de cet espace non utilisé, si à la dernière page il n'y a plus d'espace libre. A ce moment un programme ramasse-miettes comprime les pages en laissant libre seulement la zone de débordement.

Le ramasse-miettes comme tous les programmes qui manipulent des pages (images de fichiers gérés par FORTRAN) a besoin pour faire une modification d'en lire une totalement puis de la réécrire totalement. Ce procédé gênant est imposé par la gestion des pointeurs de lecture et d'écriture en FORTRAN, tout au moins sous CMS. Ainsi le programme ramasse-miettes est lent et coûteux, car il implique parfois la lecture et l'écriture totale d'un objet en mémoire secondaire. D'autre part, le ramasse-miettes pour son propre fonctionnement n'a besoin en mémoire que d'une zone tampon de la taille d'une page en plus de la page qu'il est en train de comprimer.

4 . VERIFICATION ET INTERFACE AVEC LE SYSTEME-HÔTE

L'**environnement de vérification** est chargé de la tâche de compilation du modèle (ou variante) et du jeu de données. Trois solutions sont envisageables pour la compilation :

- Un **compilateur** qui traduit du langage de modélisation en code exécutable (translatable).

- Un **pré-compilateur** qui traduit du langage de modélisation en FORTRAN, suivi d'une compilation FORTRAN.

- Le langage de modélisation est **FORTRAN**, la compilation est réduite à un **appel au compilateur FORTAN.**

Tout en reconnaissant les avantages de la première solution , pour la version prototype on a pris la deuxième solution car le système de simulation était conçu à la base pour travailler en combinaison avec le pré-compilateur LADESH [11].
En attendant la mise au point du dit pré-compilateur on a réalisé la troisième solution qui est compatible avec la deuxième.

On a voulu réaliser un appel dynamique du compilateur FORTRAN, ce qui entraîne pour celui avec lequel on a travaillé (FORTRAN-CMS) la remise de la mémoire à zéro, d'où le besoin de protéger le système de simulation dans la même zone que le système-hôte. Malgré cela le compilateur FORTRAN n'étant pas fait pour être appelé en tant que sous-programme, fonctionnait de façon défectueuse, d'où un déroulement des programmes *imprévisible*.

Pour s'assurer d'un fonctionnement correct, il fallait que le système hôte appelle le compilateur FORTRAN. (Ceci est facilement réalisable à travers le système d'extensions EXEC.)
Un petit programme en EXEC se charge d'assurer l'interface entre le système de simulation et le système hôte (CMS), et ceci reste évidemment transparent à l'utilisateur.

Avec un véritable compilateur du langage de modélisation on aurait évité ce problème.

```
MODEL ;
```

Cette commande est une demande de vérification du modèle courant (variante 1).

```
VARIANT <identificateur variante> ;
```

C'est une demande de vérification de la variante indiquée comme paramètre.

```
DATASET <identificateur jeu de données> ;
```

Commande de vérification d'un objet jeu de données.

5 . LA MISE EN OEUVRE DE LA CONSTRUCTION, LA BIBLIOTHEQUE ET LES LOTS

L'**environnement de construction** sert à la création par modules des objets suivants : modèle, variante, jeu de données et documentation.

Le modèle est spécifié à l'environnement d'information. Les seuls objets qui doivent donc être spécifiés sont la variante et le jeu de données que l'on va construire. Ces objets deviennent "courants" par la suite, et la construction leur fait référence implicitement.

Il y a deux façons de créer des nouvelles variantes et jeux de données : soit on spécifie un numéro qui les identifie, soit on laisse le système leur allouer un numéro par défaut, dans ce cas le système répondra à l'utilisateur le numéro qu'il leur a alloué. Si l'on veut rajouter un module soit à une variante, soit à un jeu de données on doit nécessairement spécifier leur numéro d'identification.

```
NEWVAR ;
```

Demande de création d'une nouvelle variante, avec identification par défaut.

```
VARIANT <identificateur variante> ;
```

Demande soit la création d'un nouvelle variante dont le numéro est donné dans la partie identificateur, soit on demande une variante qui existe déjà, et que l'on souhaite compléter.

Lors de la création d'une nouvelle variante non privilégiée (différente de 1) une copie de la variante 1 est faite sur fichier et allouée à la variante en question. On a adopté la solution de faire des copies réelles et non des copies "virtuelles" (ensemble de pointeurs et modifications) car de toute façon au moment de la compilation on est obligé de créer des copies réelles.

Par contre pour les jeux de données, il n'y a pas de copie qui leur est allouée, les commandes servent exclusivement à les identifier, de manière à les construire par la suite

```
NEWDTSET ;
```

Demande l'identification d'un nouveau jeu de données qui sera créé par la suite, de même que pour les variantes, l'utilisateur ne se soucie pas de lui donner une identification, c'est le système qui la donne.

```
DATASET <identificateur jeu de données> ;
```

Identifie le jeu de données en question en vue de sa création ou de la construction d'un module du jeu de données existant.

La construction des objets est faite essentiellement par traitement par lots. Un lot est un ensemble d'images cartes. Dans chaque lot la première carte du paquet est une protection car elle sert à identifier le paquet par rapport à un nom de modèle, la dernière carte sert de séparation entre les paquets.

A l'intérieur de ces paquets il y a des sous-paquets qui contiennent la description de la documentation, d'une variante, et d'un jeu de données. L'ordre de ces sous-paquets est sans importance.

```
BATCH ;
```

Par cette commande le lot suivant est lu, et la description contenue dans chacun des sous-paquets est insérée à la fin des objets courants indiqués dans le sous-paquet.

La bibliothèque sert exclusivement pour la construction soit du modèle (variante 1) soit des autres variantes, elle est mise à disposition de tous les modèles par le système, les modules de la bibliothèque sont donc partageables pour tous les modèles. Elle joue donc le rôle exclusif de moyen de communication entre les modèles. C'est très important pour la conception modulaire (ascendante ou descendante) d'un modèle, car elle permet de construire et tester indépendamment les modules qui feront partie du modèle. Une fois que le test d'une partie d'un module (construit comme s'il était lui-même un modèle) a donné entière satisfaction il est mis sous forme d'un module en bibliothèque de façon à pouvoir être inséré dans un autre modèle.

Trois opérations sur la bibliothèque sont donc permises : création, copie et destruction.

```
LIBRARY <type d'opération> <nom du module> ;
```

Si le type d'opération est :

- PUT c'est la création d'un module en bibliothèque, la variante courante devient un module dont le nom est indiqué.
- GET on fait une copie du module indiqué, existant dans la bibliothèque, que l'on insére à la fin de la variante courante.
- OFF destruction du module en question de la bibliothèque.

6 . INFORMATION ET DOCUMENTATION

L'environnement d'Information remplit des tâches multiples. C'est dans cet environnement que l'on peut :

- Sortir du système.
- Exploiter la documentation.
- Avoir accès à certaines informations sur l'état du système et sur certains objets.
- Faire les sauvegardes et effacer certains objets.

```
OUT ;
```

```
BYE ;
```

Ces deux commandes ont le même effet : la sortie du système. La fonction sémantique associée à ces commandes transmet cette information au superviseur général de l'interpréteur au moment même de la fin de l'exécution de la commande ; grâce à cette information le superviseur ne donne pas le contrôle à l'analyseur syntaxique (pour faire une nouvelle demande de commande), mais fait une sauvegarde de l'état du système ainsi qu'une indication de fin. Ensuite on retourne au programme EXEC déjà mentionné grâce à l'indication sauvegardée. Puis on retourne au système hôte.

Si au moment de sortir du système, on n'a pas indiqué que l'on veut sauvegarder certains produits de simulation créés dans la session, tous les résultats de la présente session seront effacés, ceci, bien évidemment, pour ne pas encombrer inutilement la mémoire secondaire.

```
KEEP RUN <indication produit de simulation> ;
```

Cette commande permet la sauvegarde des objets "produit d'une simulation" ; les autres objets étant sauvegardés automatiquement, sauf si on les efface spécifiquement.

```
ERASE <objets à effacer> ;
```

C'est la commande pour effacer (ou détruire) certains objets dont on désire ne plus se servir. Ces objets peuvent être : une variante, un jeu de données, la documentation, un produit de simulation et même un modèle (dans ce dernier cas on efface tous les objets correspondant à ce modèle. Si l'on efface le modèle courant, alors on revient automatiquement à l'environnement d'entrée). Un cas particulier : la commande ERASE RUN <identificateur simulation> ; efface l'objet produit d'une simulation. Elle n'est nécessaire que pour l'effacement des objets que l'on a sauvegardés grâce à la commande KEEP (voir aussi, environnement de visualisation).

La documentation est composée de deux parties : la structure et le dictionnaire de variables. La structure est composée de lignes commentaires qui commencent par un *, qui décrivent la structure du modèle. Il était prévu que le pré-compilateur puisse fournir automatiquement la structure modulaire du modèle. La partie dictionnaire est composée de lignes qui commencent par le nom de la variable, le reste de la ligne est libre pour l'associer à des renseignements, par exemple la signification de la variable, ses unités, et le type d'équation qui la définit.

```
STRUCT ;
```

Provoque l'impression sur le terminal de la partie structure de la documentation.

```
REPORT ;
```

Imprime sur le terminal la partie dictionnaire de la documentation, ayant fait auparavant un tri, ce qui permet de disposer d'un tableau de variables ordonnées facile à consulter.

AUTRES COMMANDES D'INFORMATION

```
LIST <objets à énumérer> ;
```

Cette commande permet l'impression de certaines informations sur l'état du système, en particulier le nombre et l'identification de certains objets existants : modèles, variantes, jeu de données, produits de simulation.

```
NAMES ;
```

A le même effet que "LIST MODEL ;", c'est-à-dire l'impression des noms de modèles existant dans le système.

```
PRINT <objets à imprimer> ;
```

```
OPRINT <objets à imprimer> ;
```

Ces commandes sont des demandes d'impression sur terminal (PRINT) ou sur fichier (OPRINT) des principaux objets : le modèle, une variante, un jeu de données ou la documentation (c'est-à-dire la partie structure et la partie dictionnaire, cette dernière partie est imprimée sans que le tri soit fait).

```
MODEL <identificateur modèle> ;
```

Cette commande permet de changer le modèle auquel on fera référence implicitement pendant la session.

7 . L'ENVIRONNEMENT DE MISE A JOUR

L'environnement de mise à jour permet de créer, modifier ou détruire des éléments des objets suivants : le modèle, les variantes, les jeux de données et les macros. On doit définir ou redéfinir l'objet *courant* sur lesquel les opérations seront effectuées.

```
U P D <objet à mettre à jour>
```

Cette commande est normalement celle de transfert vers l'environnement de **mise à jour** ; elle porte en plus une information sur l'objet courant à mettre à jour pendant que l'on reste dans l'environnement. Ces objets sont : un modèle, une variante, la documentation, un jeu de données ou une macro-définition. La commande dans ce contexte n'implique pas évidemment le changement d'environnement.

De même que pour l'environnement de construction, si l'on fait référence à une variante non créée auparavant, alors on alloue une copie du modèle à la variante appelée. C'est la voie normale pour la création des variantes.

Les modifications doivent porter sur des éléments des objets, or ces éléments doivent être libres de toute sémantique particulière (à cause de l'indépendance que l'on s'est imposée vis-à-vis du langage de description). L'élément de base retenu est la ligne. En conséquence il y a des commandes *classiques* d'un éditeur, pour se positionner sur la ligne désirée (LOCATE, TOP, BOTTOM, GOTO, UP, NEXT) et des commandes pour effectuer les changements (DELETE, REPLACE, INSERT, CHANGE). A ces commandes on a ajouté deux commandes de vérification de ligne LI (qui imprime le numéro de la ligne courante) et P (impression de la ligne courante et des suivantes).

Les commandes LOCATE et CHANGE ont besoin de l'unité syntaxique "chaîne", INSERT et REPLACE de l'unité syntaxique "ligne". Ces unités ne peuvent pas être décrites en termes d'éléments plus simples dans une grammaire LL1 et leur définition lexicographique (sans utiliser un compteur) n'est pas possible ; en effet il y aurait des ambiguïtés (car en principe une chaîne ou une ligne permettent tous les caractères, blanc, virgule, point-virgule compris) comme par exemple : Est-ce que le dernier point-virgule correspond à la ligne ou est l'unité syntaxique qui indique la fin de la commande ? C'est pourquoi on a adopté une solution simple : l'existence de caractères obligatoires et uniques pour délimiter une chaîne et une ligne. Ainsi une chaîne est définie comme une suite de caractères (max. 32) comprise entre apostrophes. Une ligne est une suite de caractères (max.72) comprise entre deux symboles dollar.

```
AGAIN <nombre de fois> ;
```

Répétition implicite de n fois l'exécution de la commande antérieure.

Cette commande oblige à sauvegarder systématiquement la commande précédente dans cet environnement. Un compteur associé à cette commande avertit le superviseur général de l'interpréteur qui alors exécute un sous-programme d'interception de l'analyseur lexicographique ; ce dernier au lieu de faire une demande de lecture d'une nouvelle commande, prend en compte celle sauvegardée.

La répétition de l'analyse syntaxique à chaque itération demandée par "AGAIN" est faite pour assurer le bon fonctionnement du système dans le cas où la commande précédant un "AGAIN" est une macro requête. En effet comme c'est la macro requête qui est mémorisée, l'interception est faite non pas pour transmettre la macro requête mémorisée mais pour sa définition avec remplacement de paramètres .

Afin de simplifier et de ne pas avoir besoin d'un mécanisme de piles, la commande AGAIN ne peut pas faire partie de la définition d'une macro, et la macro ne peut pas être récursive. La ligne vide, étant une commande, peut être répétée par AGAIN (ce cas se produit si AGAIN est la première commande émise dans l'environnement).

Commandes d'Impression :

```
LI ;
```

Impression du numéro de la ligne courante.

```
P <nombre de lignes> ;
```

Impression du nombre de lignes spécifié, par défaut, impression de la ligne courante. On se positionne à la dernière ligne imprimée.

Commandes de position avec impression de la ligne courante :

```
L <chaîne> ;
```

Positionne sur la ligne où se trouve la première apparition de la chaîne.

```
TOP ;
```

Positionne sur la première ligne.

```
BOTTOM ;
```

Positionne sur la dernière ligne.

```
UP <nombre de lignes> ;
```

On remonte du nombre de lignes indiqué ; si ce nombre n'apparait pas, on remonte d'une ligne seulement.

```
NEXT <nombre de lignes> ;
```

On descend sur l'objet, du nombre de lignes indiqué, si ce paramètre n'apparait pas, ou descend une ligne.

```
GOTO <numéro de la ligne> ;
```

On se positionne à la ligne indiquée.

Commandes de changement :

```
D <nombre de lignes> ;
```

Effacer le nombre de lignes indiqué (à défaut, une) à partir de la ligne positionnée.

```
R <ligne> ;
```

Remplacer la ligne courante par la ligne donnée en paramètre.

```
I <ligne> ;
```

Insertion de la ligne en paramètre à la suite de la ligne courante et positionne sur cette nouvelle ligne.

```
C <chaîne> <chaîne> ;
```

Remplacement de la première occurrence dans la ligne courante du premier paramètre par le deuxième paramètre, et impression de la ligne après modification.

Commandes macro :

```
NAMES MACRO ;
```

Impression des noms des macros existantes.

```
ERASE MACRO <identificateur macro> ;
```

On détruit la macro indiquée en paramètre.

```
MACRO <identificateur macro>
      <liste de paramètres formels> : <liste d'opérations simples> ;
```

Définition d'une macro - Voir Annexe III.

```
<identificateur macro> <liste de paramètres réels> ;
```

Requête d'une macro.

La manipulation de macros est simple et valable seulement pour un sous-ensemble de commandes. Néanmoins l'utilisation effective des macros exige un minimum de connaissance informatique.
La définition et la requête d'une macro, ainsi que le mécanisme macro processeur sont présentés dans l'annexe III.

8 . MISE EN OEUVRE DE LA SIMULATION

Les commandes de l'**environnement de simulation** préparent les spécifications de la simulation à lancer : la méthode d'intégration, le pas d'intégration, la demande de calcul d'erreur et l'ajustement par rapport à l'erreur. Le lancement de la simulation est fait avec la commande suivante :

```
FROM <date> TO <date> RUN <identification Opt. du run> ;
```

Dans cette commande on indique obligatoirement le début et la fin de la simulation, l'identification de la simulation est optionnelle. Les autres options de la simulation sont prises soit par défaut soit grâce aux autres commandes effectuées auparavant dans le même environnement. Cette commande effectue automatiquement le changement d'environnement vers celui de **visualisation.**

Les seules spécifications obligatoires restent évidemment le début et la fin de la simulation. La date de début initialise la variable interne TIME qui, incrémentée par le pas d'intégration, sert au superviseur de simulation à tester la fin et l'arrêt de la simulation ; cette variable TIME est la seule variable interne disponible pour être utilisée dans la description du modèle (N.B. le pas d'intégration par contre n'est pas disponible, cf. Chapitre II).

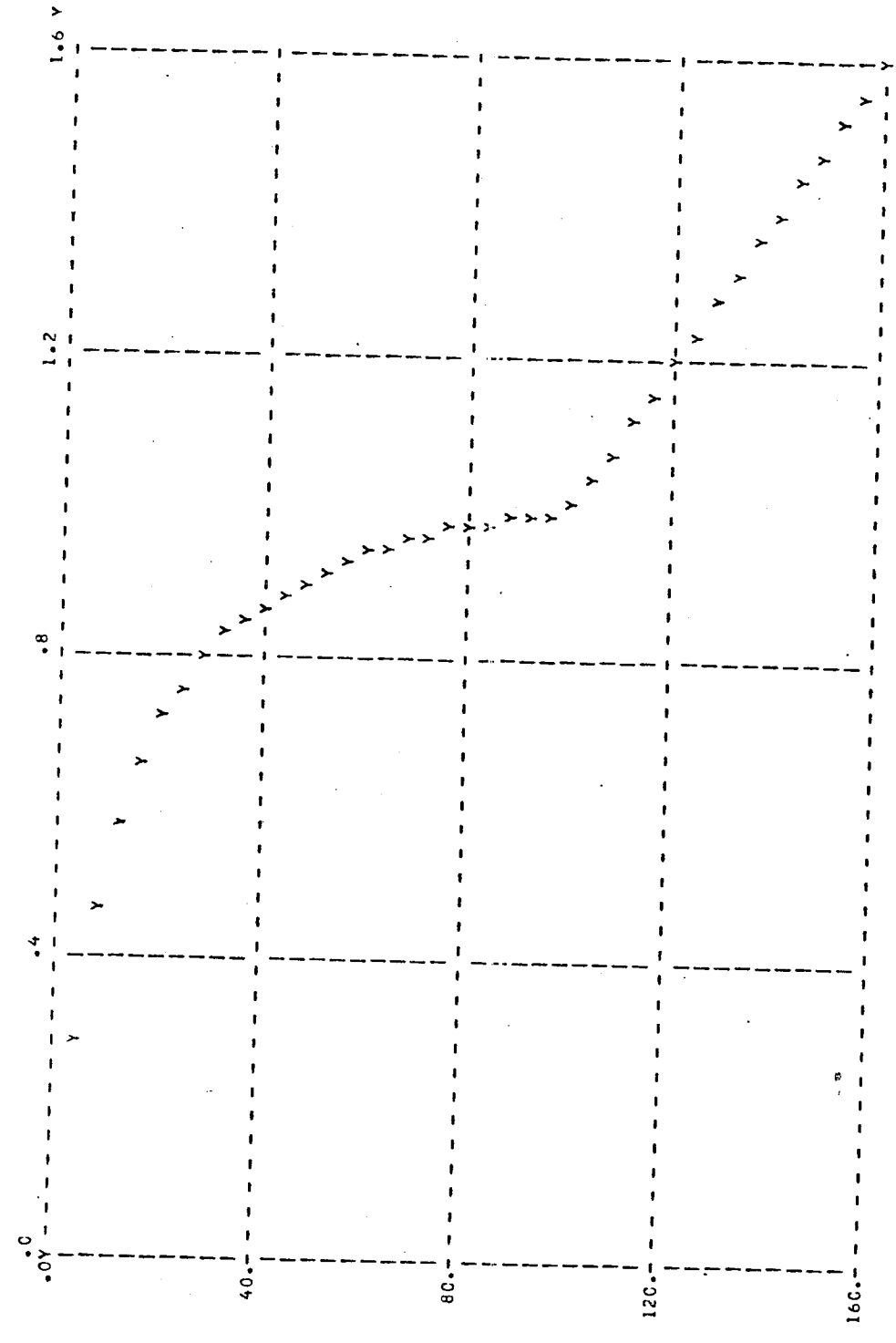
L'identification de la simulation est faite par un numéro, le système alloue comme identificateur par défaut le plus petit nombre entier non utilisé. Un choix des méthodes d'intégration relativement grand est offert (voir Annexe IV), la méthode prise par défaut étant celle d'Euler.

Le pas d'intégration, s'il est spécifié (nombre réel positif) reste constant dans la méthode choisie sauf si l'on demande l'ajustement du pas, auquel cas le pas spécifié est seulement le pas initial ; si la durée de la simulation (tleng) est plus grande que 100 alors le pas par défaut sera

$$dt = \frac{tleng}{100} \text{ sinon } dt = \frac{10 * tleng}{9\ tleng * 100}$$

(voir figure IV.1).

Figure IV.1



Il existe une demande optionnelle d'évaluation de l'erreur commise à l'intégration ; l'erreur absolue est calculée pour chacune des variables d'état (Annexe IV). La demande d'ajustement du pas est aussi optionnelle ; si l'on désire un tel ajustement, il faut indiquer un pas minimal. La méthode d'intégration exponentielle fait systématiquement l'ajustement du pas d'intégration, dans ce cas c'est à la fois un pas minimal et maximal qu'il faut indiquer (Voir Annexe IV).

La sauvegarde des résultats doit être faite à chaque itération de la simulation. Une correspondance entre les valeurs à sauvegarder dans la simulation et le sous-programme de sauvegarde doit être assurée afin d'exploiter correctement les résultats. Ceci devrait être fait normalement par le pré-compilateur ; à défaut de celui-ci on devra écrire la correspondance dans le modèle lui-même. L'existence du pré-compilateur permettrait d'introduire une commande "SAVE" pour indiquer au niveau système les variables à sauvegarder.

Le modèle doit être composé (comme dans le cas de modèles en CSSL) de deux parties : une partie initiale et une partie dynamique. Cette séparation néanmoins doit être faite au niveau de la pré-compilation et non par l'utilisateur comme avec CSSL. Le système de simulation travaille donc avec le modèle ainsi divisé :

- Un sous-programme INIT qui se charge fondamentalement de récupérer les données numériques initialisées dans le jeu de données (qui aura la forme en FORTRAN d'un BLOCK-DATA) et de faire des calculs statiques préalables à la simulation, comme par exemple l'initialisation des variables d'état transparentes au niveau modèle (i.e. délais, fonctions de mémorisation etc...).

- Un sous-programme DERIV qui contient la description dynamique du modèle c'est-à-dire de l'ensemble des équations à l'exception des intégrales. Ce sous-programme exécute donc une itération à la fin de laquelle il fait la correspondance pour la sauvegarde de résultats de l'itération. Il retourne en paramètre au superviseur de la simulation les valeurs des fonctions à intégrer au temps considéré.

L'exécution du modèle ne peut pas être demandée par un programme FORTRAN car le modèle n'est pas chargé en même temps que le système. Conscients que "la fonction la plus fondamentale qu'un système conversationnel réalise pour l'utilisateur est le démarrage de l'exécution" [12], on a incorporé cette facilité dans le système. Le chargement, l'édition de liens et l'exécution sont faits implicitement par la commande simulation.

La commande de simulation transmet au programme EXEC les informations nécessaires au chargement et lancement de l'exécution. C'est le programme superviseur de la simulation qui se charge alors d'appeler les différentes options et méthodes d'intégration, celles-ci alors font exécuter comme des sous-programmes : le modèle (avec les fonctions du langage), le jeu de données et la sauvegarde des résultats. Après l'exécution le programme ré-initialise le superviseur de l'interpréteur dans l'environnement de visualisation.

Afin de contrôler les temps de simulation on a ajouté dans le prototype un programme écrit en assembleur qui calcule le temps de simulation grâce aux horloges internes de la machine.

0 : NON DEMANDE OU PAR DEFAUT 1 : SPECIFIE

CAS	1	2	3	4	5	6	7	8	9	10
METHODE	1	0	1	1	1	1	1	1	1	0
PAS	0	1	0	0	0	1	1	1	1	0
EVALUATION ERREUR	0	0	1	0	1	1	1	0	0	0
AJUSTEMENT PRECISION	0	0	0	1	1	1	0	1	0	0

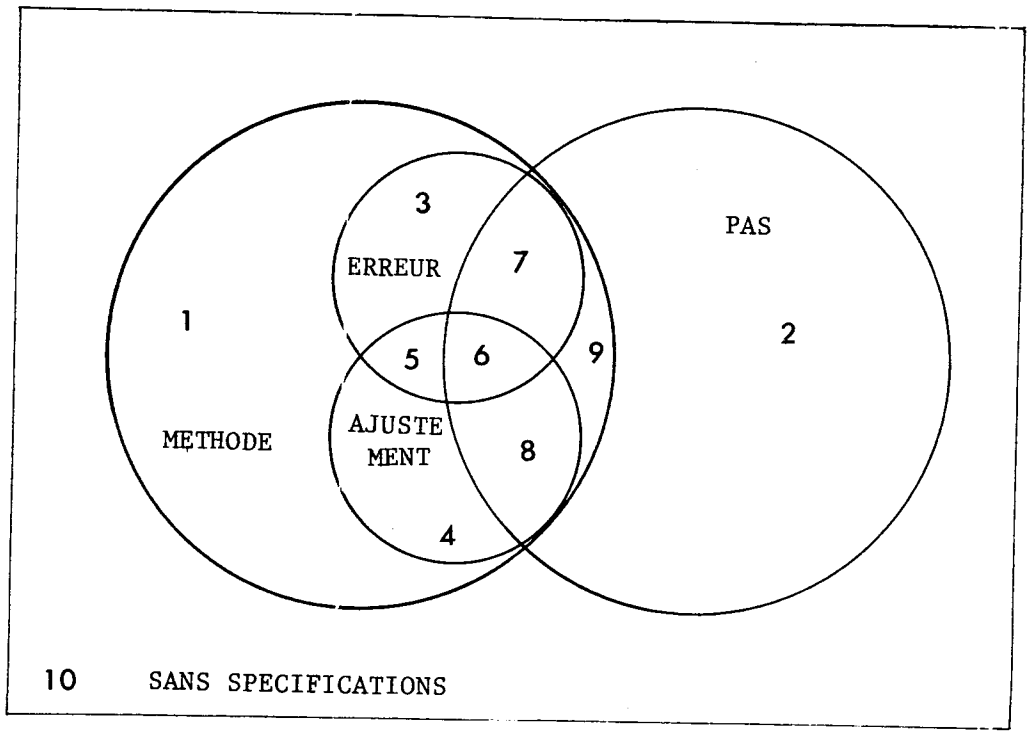


DIAGRAMME DE VENN
MONTRANT LES DIFFERENTES POSSIBILITES
DE SPECIFICATION D'UNE SIMULATION
(IDENTIFICATION DE LA SIMULATION NON-COMPRISE)

Figure IV.2

Les commandes suivantes permettent de spécifier la méthode d'intégration, éventuellement le calcul de l'erreur et l'ajustement du pas d'intégration. Voir l'Annexe I pour la syntaxe de ces deux paramètres, et l'Annexe IV pour leur utilisation.

```
EULER <évaluation optionnelle de l'erreur>
                <ajustement optionnel du pas> ;
```

La méthode choisie est celle d'Euler.

```
TRAPEZ <évaluation optionnelle de l'erreur>
                <ajustement optionnel du pas> ;
```

Méthode trapézoïdale.

```
SIMPS <évaluation optionnelle de l'erreur>
                <ajustement optionnel du pas> ;
```

Méthode Simpson.

```
HEUN <évaluation optionnelle de l'erreur>
                <ajustement optionnel du pas> ;
```

Méthode de Heun.

```
CAUCHY <évaluation optionnelle de l'erreur>
                <ajustement optionnel du pas> ;
```

Méthode de Cauchy ou de la tangeante améliorée.

```
PREDCO <évaluation optionnelle de l'erreur>
                <ajustement optionnel du pas> ;
```

Méthode Prédicteur-Correcteur (Milne) de 2ème ordre.

```
EXPON <spécification pas minimal et pas maximal> ;
```

On spécifie comme méthode d'intégration la méthode exponentielle, celle-ci a besoin comme paramètres obligatoires du pas minimal et maximal, dans un ordre quelconque.

```
DT <nombre réel> ;
```

Spécification

- soit du pas d'intégration (méthodes à pas constant)
- soit du pas initial si l'on demande l'ajustement du pas d'intégration.

9 . VISUALISATION ET ANALYSE DES RESULTATS

Grâce à une programmation de type modulaire on a tiré profit du fait qu'il y a des tâches communes pour plusieurs des commandes de cet **environnement de visualisation et analyse des résultats.**

Cet environnement offre les commandes suivantes :

```
KEEP RUN <identificateur simulation> ;
```

Il s'agit d'une commande de sauvegarde des résultats d'une simulation, car il faut se rappeler que si on n'a pas émis cette commande et si l'on sort du système, l'objet résultat de la simulation n'est pas sauvegardé pour les sessions suivantes.

Cette commande est donc indispensable si l'on veut comparer les résultats d'une session à ceux d'une autre. Pour des raisons de sécurité, cette commande est également disponible dans l'environnement d'information. Par contre on n'a pas permis dans cet environnement une commande ERASE RUN afin d'éviter que l'on efface l'objet produit de la simulation sur lequel on est en train de travailler.

```
OTHRUN <identificateur simulation> ;
```

Si l'on veut vérifier les résultats d'une simulation antérieure ou bien visualiser certains résultats d'une façon que l'on n'a pas fait , on peut **changer grâce à cette commande l'objet produit d'une simulation** sur laquelle les visualisations de cet environnement sont faites.

```
VARIABLE ;
```

Cette commande imprime les **noms de toutes les variables sauvegardées** dans l'objet produit de la simulation ; celle-ci est très utile quand on revient à un ancien objet produit d'une simulation et que l'on a oublié les variables qui peuvent être exploitées.

```
MINMAX <liste de variables> ;
```

Cette commande a comme effet de calculer et imprimer les valeurs minimale et maximale obtenues au cours de la simulation pour l'ensemble de variables spécifiées.

```
TITLE <ligne> ;
```

Impression d'un titre (optionnel) comme en-tête des graphiques demandés à la suite de la simulation.

```
MEAN <liste de variables> ;
```

Calcul et impression de la **moyenne temporelle** d'un ensemble de variables tout au long de la simulation. Pour une simulation à pas d'intégration constant, elle est obtenue comme $\overline{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$ où les X_i sont les valeurs de X dans la simulation et n le nombre d'itérations. Par contre pour une méthode à pas variable, elle est obtenue comme $\overline{X} = \frac{1}{L} \sum_{i=1}^{n} X_i \cdot dt_i$ où L est la longueur de la simulation, et dt_i est le pas d'intégration à **chaque itération** i.

```
HISTO <liste de variables> <plage opt.> ;
OHISTO <liste de variables> <plage opt.> ;
```

Cette commande imprime l'histogramme soit sur terminal (HISTO) soit sur fichier (OHISTO) pour l'ensemble de variables indiqué sur la liste. Un tel histogramme est obtenu soit pour une date particulière (<plage> = <date>), soit pour la moyenne temporelle (par défaut ou <plage> = MEAN). L'histogramme a comme en-tête celui indiqué par la commande TITLE.

```
COMPAR  <variable> <moyenne et identification run> ;
OCOMPAR <variable> <moyenne et identification run> ;
```

La comparaison de résultats pour une même variable appartenant à plusieurs simulations (par exemple divers scénarios, ou certains tests de sensibilité) peut être effectuée grâce à ces deux commandes, les résultats pouvant être visualisés soit sur terminal (COMPAR) soit sur imprimante rapide via fichier (OCOMPAR). On a retenu pour l'instant deux moyens clairs de faire les comparaisons :

- soit l'histogramme de la variable à une date donnée ou de sa moyenne temporelle :
  HISTO <plage> RUNS <liste identificateurs run>
- soit le graphique de l'évolution temporelle des variables résultant de leur simulation :
  PLOT RUNS <liste identificateur run = lettre>

S'agissant d'une comparaison, cette option graphique fait l'ajustement automatique de toutes les échelles par rapport à la plus grande échelle de l'ensemble de courbes (cf. option UNIF de la commande PLOT).

Les commandes de visualisation sont les suivantes :

```
PRINT <liste de variables> ;
OPRINT <liste de variables> ;
```

Impression soit sur terminal (PRINT) ou sur ficher (OPRINT) des valeurs de la variable interne **temps** et des autres variables indiquées dans la liste. Ces séries chronologiques ne sont pas homogénéisées dans le temps, c'est-à-dire qu'elles rendent compte exclusivement des valeurs telles qu'elles sont obtenues à la simulation même pour les méthodes d'intégration à pas variable.

```
**********************************************************************

        *MODELLING LANGUAGE: LADESH              *SIMULATION SYSTEM: SAISYE

        *VERSION IMAG JUNE 1975

**********************************************************************

        *MODEL : TEST          *VARIANT : 1      *DATASET :  2    *RUN : 4

        *SCENARIO DESCRIPTION :  DY(T) = -T * Y(T) ,  Y(0)=1

        *SIMULATION TIME :          1.789 SEC

**********************************************************************

        *INTEGRATION METHOD : PREDICTOR -CORRECTOR (MILNE-HEUN)
        * WITH ERROR CALCULATION

        *INTEGRATION STEP :  0.10000E 00

**********************************************************************
```

TIME	Y(1)
0.0	1.0000
0.100	0.99013
0.200	0.97065
0.300	0.94216
0.400	0.90546
0.500	0.86158
0.600	0.81172
0.700	0.75718
0.800	0.69931
0.900	0.63947
1.000	0.57896
1.100	0.51898
1.200	0.46060
1.300	0.40473
1.400	0.35210
1.500	0.30328
1.600	0.25862
1.700	0.21834
1.800	0.18249
1.900	0.15101
2.000	0.12371
2.100	0.10033
2.200	0.80547E-01
2.300	0.64017E-01
2.400	0.50366E-01
2.500	0.39227E-01
2.600	0.30241E-01
2.700	0.23078E-01
2.800	0.17432E-01
2.900	0.13033E-01
3.000	0.96442E-02

```
**************************************************

        LAST CALCULATED ERROR FOR STATE VARIABLES

                -0.77220E-04

**************************************************
```

PLOT <liste d'objets à tracer> ;

OPLOT <liste d'objets à tracer> ;

Trace dans un même graphique l'évolution temporelle d'un ensemble de variables, sur terminal (PLOT) ou sur fichier (OPLOT).
Chaque variable est identifiée par une lettre :

variable = lettre

par exemple :

POP1 = B.

Il existe trois options sur les échelles de graphique pour les variables :

- UNIF cette option originale permet, en indiquant ce paramètre avant la liste de variables, d'**uniformiser automatiquement toutes les échelles** des variables apparaissant sur le même graphique par rapport à l'échelle la plus étendue dans la liste des variables spécifiées.

- (<limite inférieure de l'échelle> <limite supérieure de l'échelle>) Cette option permet de fixer arbitrairement l'échelle de chacune des variables, en indiquant à la suite de chacune les bornes désirées.

- ne rien indiquer. Cette option choisit l'échelle pour chaque variable, les bornes de l'échelle seront les valeurs extrêmes de la variable obtenues.
  - . soit au cours de toute la simulation,
  - . soit déterminée par les bornes temporelles de l'option FROM <date> TO <date> (voir ci-après).

ZOOM <augmentation/diminution> <liste d'objets à tracer> ;

OZOOM <augmentation/diminution> <liste d'objets à tracer> ;

Trace dans un même graphique l'évolution temporelle d'un ensemble de variables, comme pour les commandes PLOT et OPLOT respectivement, mais à la différence qu'un **effet ZOOM (positif et négatif)** est obtenu sur l'**axe temporel** (axe des abscisses) ; cette commande originale est très utile lorsque l'on doit travailler avec des pas d'intégration petits (par exemple pour des raisons de stabilité) et que le tracé graphique obtenu point par point (commande PLOT) est trop grand pour permettre une vision globale. Il est également très utile lorsque l'on veut visualiser un détail agrandi d'une trace (par exemple un changement brusque) sans pour autant refaire la simulation avec un pas d'intégration plus petit.
Les indications et options pour les échelles sont les mêmes que pour les commandes PLOT ou OPLOT. L'effet ZOOM est la façon naturelle de surmonter les inconvénients d'une représentation discrète d'un tracé continu. En conséquence les options d'augmentation sont exprimées en **entiers.**

Ainsi par exemple :

ZOOM +1 a le même effet que PLOT ; il correspond au graphe de tous les points obtenus dans toute ou partie de la simulation.

ZOOM 2 correspond à l'obtention d'un point supplémentaire entre chaque couple de points du produit de la simulation. L'obtention des points supplémentaires est faite par interprétation linéaire.

ZOMM -2 correspond à un graphe obtenu d'un point sur deux à partir d'un produit de la simulation.

Il est clair que ni ZOOM 0 ni ZOOM -1 n'ont de signification, et sont détectés comme erreurs.

Un vrai effet ZOOM peut être réalisé en jouant simultanément sur l'axe des ordonnées (à travers l'échelle de la variable) et sur l'axe des abscisses à travers la commande ZOOM. On note que la commande ZOOM ne remplace pas un changement du pas d'intégration (conditions de stabilité, précision, etc...) ; elle sert exclusivement à mieux apprécier les mêmes résultats et ce n'est qu'un palliatif des contraintes d'impression du terminal ou de l'imprimante.

```
BISPACE  <variable> <variable> ;
```

```
OBISPACE  <variable> <variable> ;
```

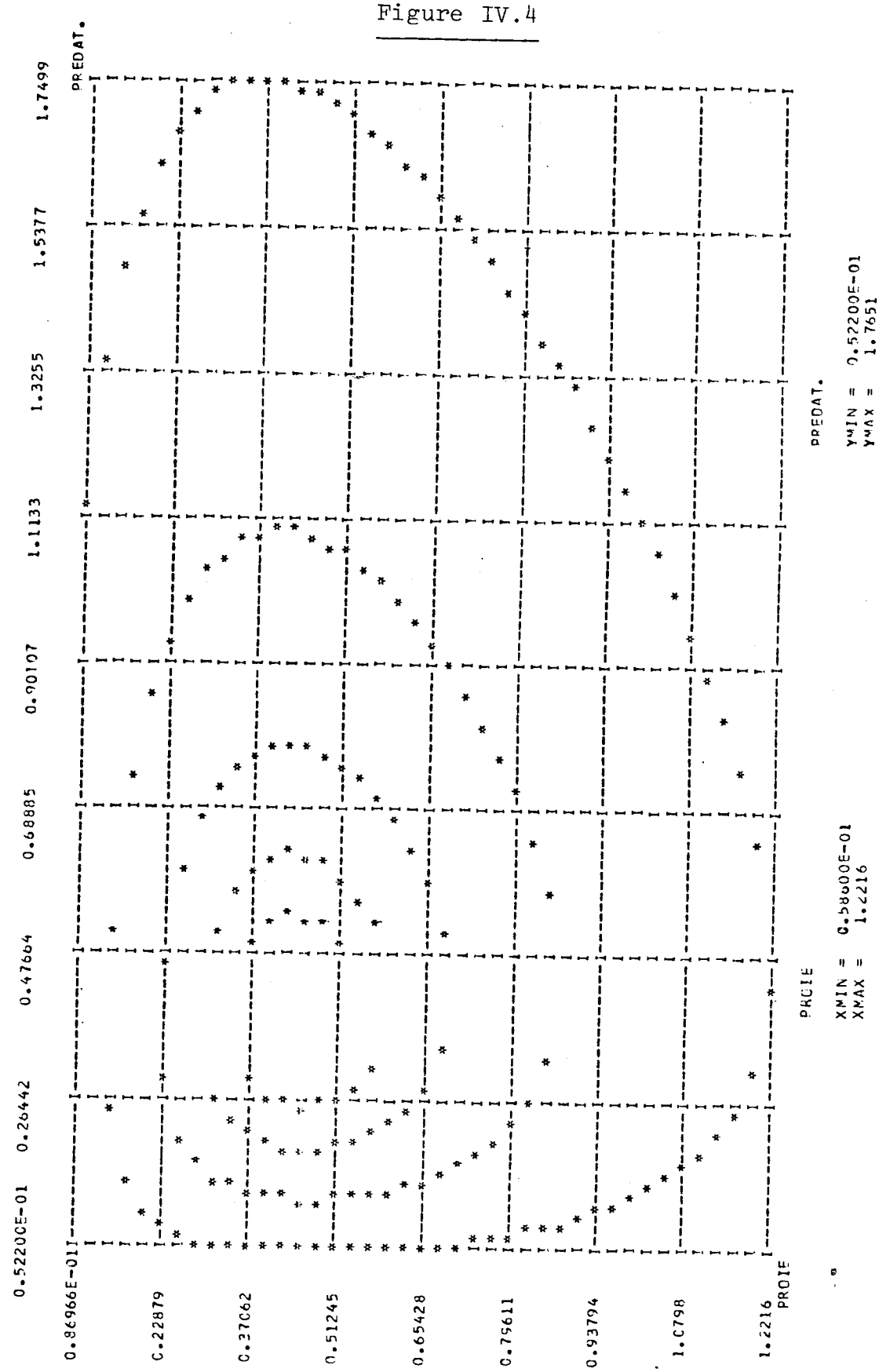
Ces deux commandes originales permettent d'obtenir un **graphique bidimensionnel** ayant comme axe des abscisses, la première variable indiquée comme paramètre et comme axe d'ordonnées, la deuxième variable. Le graphique est cadré automatiquement à la taille d'une feuille d'imprimante (donc on n'a pas besoin de spécifier les échelles). Afin de faciliter la lisibilité du graphe, on ne permet pas la superposition de courbes dans un même graphique ; en conséquence on ne spécifie pas la lettre (comme dans PLOT) qui représente la courbe.

Un cas particulier, et certainement très utile, est le cas d'un graphique d'espace de phase (une variable et sa dérivée).
Un exemple classique d'utilisation d'espace bidimensionnel sont les équations de Volterra

$$x' = + a x - b x y$$
$$y' = - c y + d x y$$

(Voir figure IV.4).

Figure IV.4


MODELE LUTTE POUR LA VIE DE VOLTERRA
0.52006E-01 0.26442 0.47664 0.68885 0.90107 1.1133 1.3255 1.5377 1.7499
PREDAT.
0.86966E-01
0.22879
0.37062
0.51245
0.65428
0.79611
0.93794
1.0798
1.2216
PROIE
PROIE
XMIN = 0.58000E-01
XMAX = 1.2216
PREDAT.
YMIN = 0.52200E-01
YMAX = 1.7651

Au moment d'entrer dans l'environnement de visualisation ou lorsque l'on change d'objet produit de la simulation, on lit en mémoire cet objet. Si l'objet correspond à une simulation à pas variable alors les résultats sont homogénéïsés automatiquement avant d'être exploités graphiquement ; l'homogénéïsation temporelle est effectuée par interpolation linéaire par rapport au plus petit pas d'intégration.

```
FROM <date> TO <date><commandes visualisation> ;
```

Grâce à cette commande originale on peut **visualiser** les résultats d'une simulation **sur un intervalle de temps plus court** que celui spécifié dans la commande de simulation.

BIBLIOGRAPHIE

[1] GRAM C., HERTWECK F.R.

"Command Languages : Design considerations and Basic Concepts"
dans Proceedings of the IFIP Working Conference on Command Languages
July 74, Lund, Sweden, North Holland - 1975

[2] SCALAN T.R.

"Control Statement Definition using command grammars"
METRA pp.411-427 - Vol. XII n°3, 1973

[3] LEAVENWORTH B.M.

"Syntax Macros and Extended Translation"
Communications of the ACM, Vol. 9 n°11, pp.792, Nov. 1960

[4] SAVARY H.

"Outils de mise au point pour langages de haut niveau : association de modules et contrôle de l'exécution"
Thèse de 3ème Cycle Informatique - USMG, Grenoble, Septembre 1973

[5] CAMPMAS M., LATOMBE J.C.

"Un programme général pour l'interprétation de langages d'application"
Automatisme, Janvier-Février 1976

[6] BOLOPION A., CAMPMAS M., LATOMBE J.C., SABONNADIERE J.C.

"L'interpréteur générateur des langages de Commandes"
E.N.S. d'Electrotechnique et de Génie Physique de Grenoble, Mai 1973

[7] GRIFFITHS M.

"Use of the grammar transformer"
PL/1 Compiler Group Memorandum.
Oct. 1968

[8] GRIFFITHS M., PELTIER M.

"Grammar transformation as an aid to compiler production"
IMAG, Feb. 1968

[9] DE CALUWE R.

"Etude du langage de Commande et de contrôle pour le réseau SOC"
Thèse de 3ème Cycle Informatique, USMG - Sept. 1973

[10] MENAIDIER J.P.

"Structure et fonctionnement des ordinateurs"
Larousse, Paris, 1971

[11] RECHENMANN F.

"Analyse et Modélisation descendantes des systèmes socio-économiques"
Thèse de Docteur-Ingénieur Informatique - INPG, Juillet 1976

[12] DOLOTTA T.A., IRVINE C.A.

"Proposal for a time sharing structure"
Information Processing 68, p.493, North Holland, 1969.

DEUXIEME PARTIE

LES SYSTEMES INFORMATIQUES DE SIMULATION CONTINUE

CHAPITRE V

SYNTHÈSE ET PROLONGEMENTS

TABLE DES MATIERES

1 . EVALUATION DES CONTRAINTES

Le but d'accessibilité que l'on s'était fixé, c'est-à-dire rapprocher au mieux le système de simulation des besoins et du processus de simulation chez l'utilisateur, n'a pas été complètement atteint. En effet cette dernière idée s'est avérée simpliste, car lors de la réalisation on a dû faire des adaptations à cause des outils employés.

Les origines de la plupart des contraintes sont, d'une part le choix du langage FORTRAN comme langage d'écriture du système, d'autre part le fait de modéliser en FORTRAN ou de passer par un pré-compilateur qui génère du FORTRAN.

L'écriture du système en FORTRAN a impliqué :

- Une manipulation lente des caractères et des chaînes alphanumériques. Leur codage interne en tant que nombres entiers ou réels, occupe de la place mémoire. Des appels nombreux aux programmes de construction d'une chaîne à partir de caractères ou de décomposition de cette dernière doivent être faits.
De multiples tableaux de correspondance sont aussi nécessaires.

- L'allocation dynamique de la mémoire ne peut pas être réalisée sans compromettre radicalement la transportabilité, (par exemple à travers certains programmes en assembleur [1]).
Un tel fait a eu de nombreuses répercussions par exemple au niveau de la mise en oeuvre de définitions syntaxiques du type <liste de ...> (voir Annexe I) où la longueur de listes est limitée. Si l'on veut travailler avec des objets parfois grands et parfois petits on est contraint de réserver une place mémoire très grande, la "pagination" (voir Chapitre IV) n'est à ce propos qu'un palliatif.

• Le système exige de travailler avec une mémoire permanente, or l'adressage direct de la mémoire secondaire tel qu'il existe sur certains FORTRAN (p.e.g.IBM sous CMS) n'est pas conçu pour des programmes d'une vie au delà de la session ; en effet, pour pouvoir utiliser un fichier en accès direct, on doit obligatoirement passer par l'instruction "DEFINE FILE", or cette instruction crée en mémoire secondaire autant de nouveaux enregistrements que ceux définis par la taille du fichier, d'où un allongement de la taille fichier à chaque session. L'adressage séquentiel a l'inconvénient de devoir "rebobiner" entre lectures et écritures et la modification d'un seul enregistrement implique obligatoirement la duplication du fichier entier et non une simple superposition d'enregistrements [2].

Le fait de passer par un modèle écrit (ou traduit) en FORTRAN, a pour conséquence l'appel dynamique (par programme) du compilateur FORTRAN ; évidemment cet appel ne peut pas être fait en FORTRAN et, ce qui est pire, en général (comme sous CMS) l'appel ne peut pas être fait en tant que sous-programme (tout au moins avec un déroulement correct). Il existe des façons de tourner le problème comme on le fait avec EXEC-CMS. Mais il se peut que dans certains ordinateurs, on soit amené à utiliser le système de simulation en deux parties séparées : une pour l'information, la construction et la mise à jour, l'autre pour la simulation et la vérification. La compilation et le chargement sont effectués directement dans le système hôte ; évidemment l'environnement de vérification disparaît, les autres environnements restent tels quels (avec des entrées/sorties supplémentaires entre les environnements et le système hôte). Voir Figure V.1.

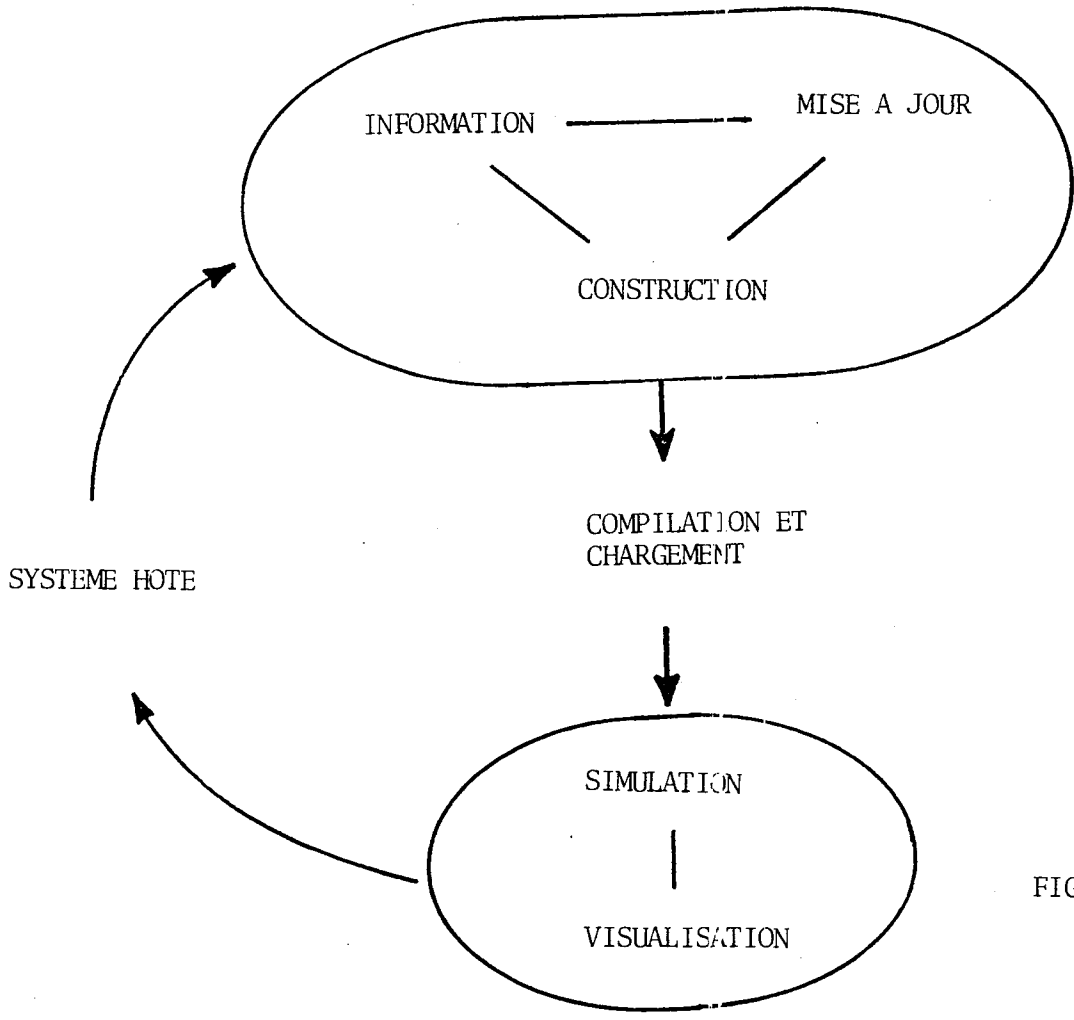


FIGURE 5-1

Evidemment si l'on dispose d'un vrai compilateur du langage de modélisation (cf. Chapitre IV) sans passer par FORTRAN, ces problèmes pourront être évités ; en plus la sauvegarde des résultats pourra être facilement faite par une commande, sans passer par le compilateur FORTRAN.

Il est difficile d'évaluer a priori la transportabilité du système de simulation. Sous la forme de la figure V.1, on l'estime à 90 %. En effet il reste toujours à vérifier entre une machine (et son FORTRAN) et une autre au moins, les problèmes suivants :

- La correspondance entre le nombre des unités logiques d'entrée/sortie et les unités réelles tant de lecture/écriture sur console, que des fichiers de la mémoire secondaire.
- Vérifier la compatibilité de fichiers (longueur maximale du fichier, taille de ces enregistrements, etc...).

- La représentation interne de nombres (logiques entiers, réels) en tant qu'octets ou bits, peut avoir des conséquences importantes, car le codage interne des caractères, des identificateurs etc...,est fait en fonction de cela. Le changement de déclaration et la redéfinition de certains vecteurs et variables peut être nécessaire, ainsi que le changement de certains formats d'écriture.

- Le ré-ordonnancement de certains tableaux doit se faire, car ceux-ci ont été ordonnés par rapport au code EBCDIC de façon à accélérer la recherche sur ces vecteurs.

- On doit aussi vérifier les limites de fonctionnement de certaines fonctions, comme par exemple "EXP" qui sont évidemment particulières à chaque FORTRAN.

2 . PROLONGEMENTS TECHNIQUES

La recherche sur les possibilités techniques d'un système de simulation est encore ouverte. On exposera certaines extensions du système de simulation dont parfois on ne connait ni la solution définitive, ni la pertinence ; avant leur incorporation une étude à ce propos devrait être faite. On note aussi que quelques prolongements exigent une homogénéisation et un travail en commun avec le compilateur du langage de modélisation employé.

Il y a des extensions immédiates comme la suppression de certaines simplifications, par exemple en ce qui concerne la nature et la longueur d'identificateurs.

En ce qui concerne les méthodes d'intégration (voir Annexe IV) il reste encore beaucoup à faire :

- l'introduction de méthodes implicites pour obtenir une meilleure stabilité et précision ;

- la prise en compte de points singuliers par les méthodes spécialement dans le calcul de la précision ;

- l'incorporation d'un dérivateur formel permettant la mise en ceuvre de certaines méthodes ainsi qu'un meilleur calcul de l'erreur.

- l'introduction d'autres types d'ajustement du pas d'intégration, par exemple par rapport à la stabilité et à l'erreur simultanément (en valeur absolue, pourcentage ou nombre de chiffres décimaux) ou simplement par rapport à la différence relative entre deux calculs avec des pas d'intégration différents.

La création d'un mécanisme général de macros commun à tous les environnements, ou plus particulièrement aux environnements de simulation et de visualisation peut être souhaitable.

Si le système travaille en commun avec le compilateur du langage de modélisation, des renseignements utiles à l'information pourront être obtenus (par exemple de la taille d'un module). En particulier on peut envisager la localisation automatique d'éléments d'un modèle (secteurs, événements, etc...) ; la structure hiérarchique du modèle pourrait être fournie automatiquement car celle-ci est connue en vue de l'ordonnancement d'équations. L'association entre la documentation et le modèle peut être facilement réalisée en vue de l'édition automatique d'un rapport technique. Finalement il permettrait le test de cohérence dimensionnelle des équations.

La cohérence dimensionnelle des équations demande, au préalable, une transformation d'unités associées aux données dans des classes d'équivalence. Les classes sont donc formées par des unités égales à un facteur multiplicatif et/ou additif près (par exemple entre barril et tonne de pétrole).
On parccurt les équations déjà substituées et simplifiées par le compilateur, en substituant les variables par leurs classes d'équivalence associées, et on applique les règles de simplification décrites ci-après, qui permettront ensuite de tester l'égalité ou non des classes d'équivalence associées aux deux membres d'une équation.

Les règles sont les suivantes :

Si 1 est la classe de variables sans dimension
$\forall$ I, J $\in$ Q (ensemble de classes) et $\forall$ n, m $\in$ Z

Opérations formant une nouvelle classe :

si $I \neq J$ 1/ $I * J = J * I$

2/ $I / J = I * J^{-1}$

Opérations de simplification :

$1 * I = I * 1 = I$

$I \pm I = I$

$I / I = I^{\circ} = 1$

$I / 1 = I$

$I^m * I^n = I^{m+n}$

$I^n / I^m = I^{n-m}$

L'incorporation de certains test automatiques de sensibilité peut être envisagée par l'introduction d'un dérivateur formel (linéarisation) et par l'introduction de routines génératrices de nombres pseudo-aléatoires (dans une **certaine** plage) ; sa mise en oeuvre est délicate à cause de l'existence de discontinuités possibles et points singuliers.

Les nouvelles techniques de l'infographie sur écran sont attractives à incorporer. En particulier, la réalisation de dessins cartographiques, la visualisation du réseau hiérarchique du modèle ainsi que la visualisation dynamique de flux entre modules, c'est-à-dire quand au cours de la simulation la grosseur de la connexion est proportionnelle à la valeur de la variable représentant le flux.

Les problèmes de sécurité ont été laissés à la charge du système hôte, dans le prototype. Si l'accès est permis à plusieurs utilisateurs, un mécanisme d'identification de l'utilisateur avec contraintes d'accès limité (confidencialité) est important au niveau de l'environnement d'entrée.

D'autre part, certaines commandes "dangereuses" comme la destruction ou l'effacement de certains objets seront éventuellement l'objet d'une confirmation.

Au niveau de "gadget" pour le moment, l'incorporation de commandes "garde-fous" pour arrêter la simulation peut devenir intéressante (par exemple si la valeur d'une variable dépasse un certain seuil) ; au même titre restent l'inhibition d'événements discrets par commande ou l'existence de zones de données locales ou non-partageables.

Un problème plus difficile à réaliser et dont l'intérêt est fort mis en doute, est l'exploitation du système de simulation comme sous-programme d'une application (segmentation, voir Chapitre II). Il parait par contre plus intéressant de laisser à l'utilisateur, la possibilité de se définir un autre environnement, par exemple, pour insérer des programmes d'analyse de données ou certaines routines statistiques utiles à la conception.

La compilation partielle de modules d'un modèle, où l'ordre des équations peut être établie de façon interne, (par exemple les secteurs du langage LADESH [3]), demande une gestion de fichiers fort compliquée ; son intérêt est seulement relatif et pour des temps de compilation et d'exécution très grands.

3 . PEDAGOGIE ET GUIDAGE

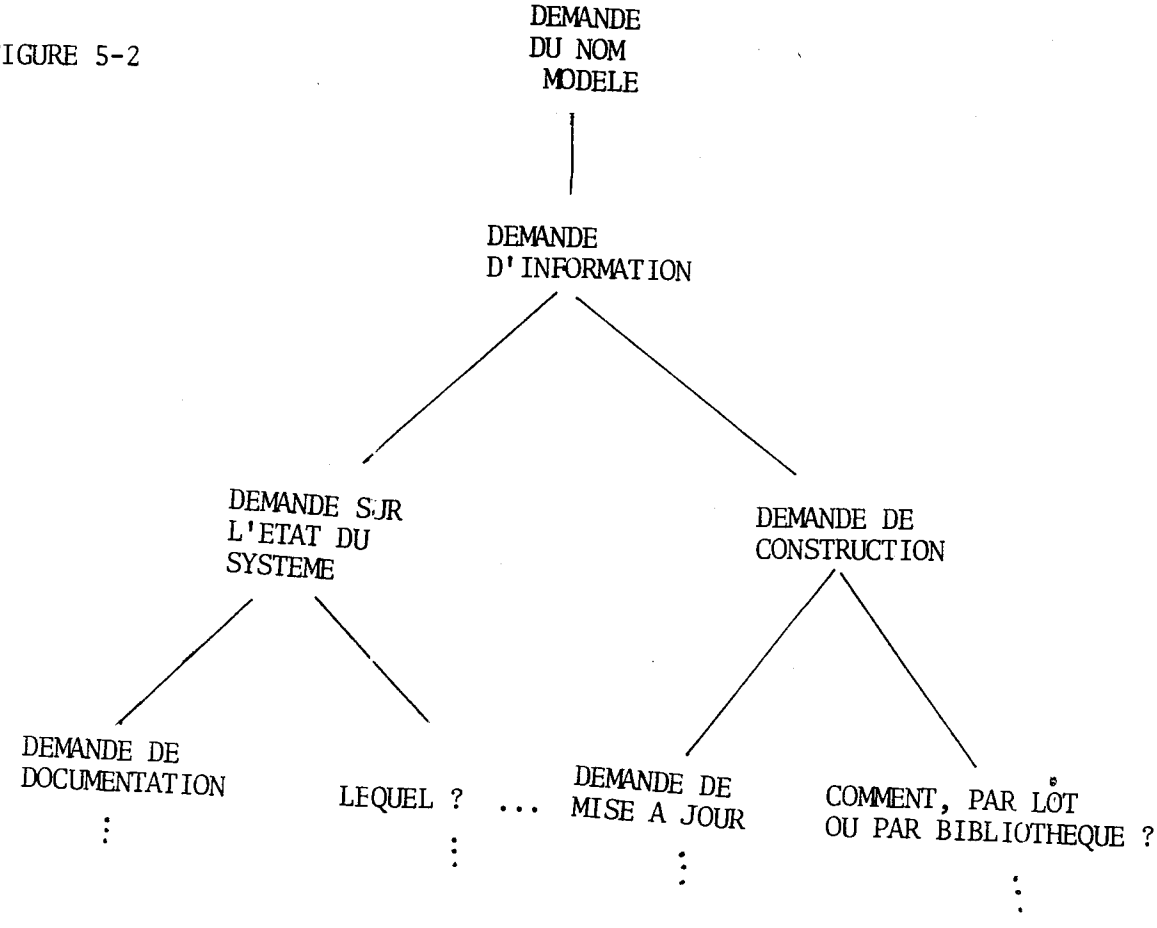
Il ne s'agit pas ici de proposer un système pour l'enseignement, assisté par ordinateur de la simulation ou de la modélisation [4], [5], problèmes sans doute très importants, mais plus modestement d'apprendre à se servir de l'outil proposé.

Le système de simulation étant conçu pour des non-informaticiens, la pédagogie a une rôle important. Il est clair que l'expérience compte et la bonne utilisation du système demande un apprentissage.

Le système dans son prototype actuel incorpore un guidage à travers les environnements, ainsi que certaines commandes de "sauvetage".

il existe le projet de réaliser une version pédagogique dialoguée, où le dialogue serait conduit par le système. Deux possibilités existent ayant pour but de faire apprendre soit les commandes, soit l'utilisation et les possibilités d'un système. Pour le premier cas, un dialogue structuré peut être incorporé et offrir à l'utilisateur un "menu" de commandes à utiliser. Pour le deuxième cas, un dialogue conditionnel [4] s'avère plus efficace ; les réponses ne sont pas toujours oui ou non, mais parfois la sélection d'un objet ou l'indication d'un paramètre. Pour ce dernier cas, il est clair que des simplifications devront être faites (voir figure V.2) sous peine d'avoir un dialogue extrêmement long et fastidieux.

FIGURE 5-2



Un problème particulier d'apprentissage est le cas d'un utilisateur qui n'a pas été le concepteur du modèle. Le problème premier est de se familiariser avec le modèle lui-même, pour cela la documentation devrait permettre de répondre à tout type de réponses sur le modèle, sa structure et ses éléments. Les informaticiens identifieront sans doute ce problème, à la création d'une base de données associée à la documentation.

Si le modèle est écrit dans un langage de modélisation hiérarchique (par exemple LADESH), il comporte donc une structure directement traduisible dans une base de données de type hiérarchique. Mais la structure d'une telle base de données risquerait d'être spécifique au modèle. On peut néanmoins tenter de décrire une structure de base de données associée à chaque langage de modélisation. On montre dans la figure V.3 la structure d'une base de données représentant la documentation du système associée à LADESH selon le modèle relationnel [5]. Par souci de simplicité, on ne spécifie pas la (ou les) relations associées à chaque articulation, par exemple :

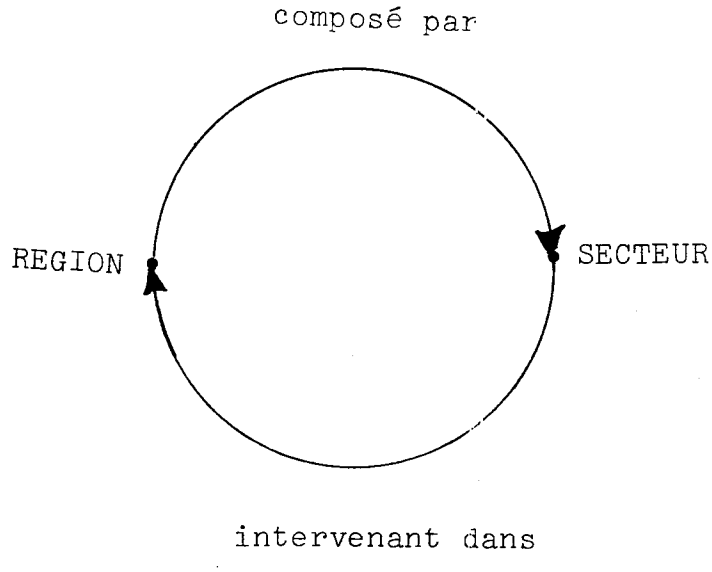
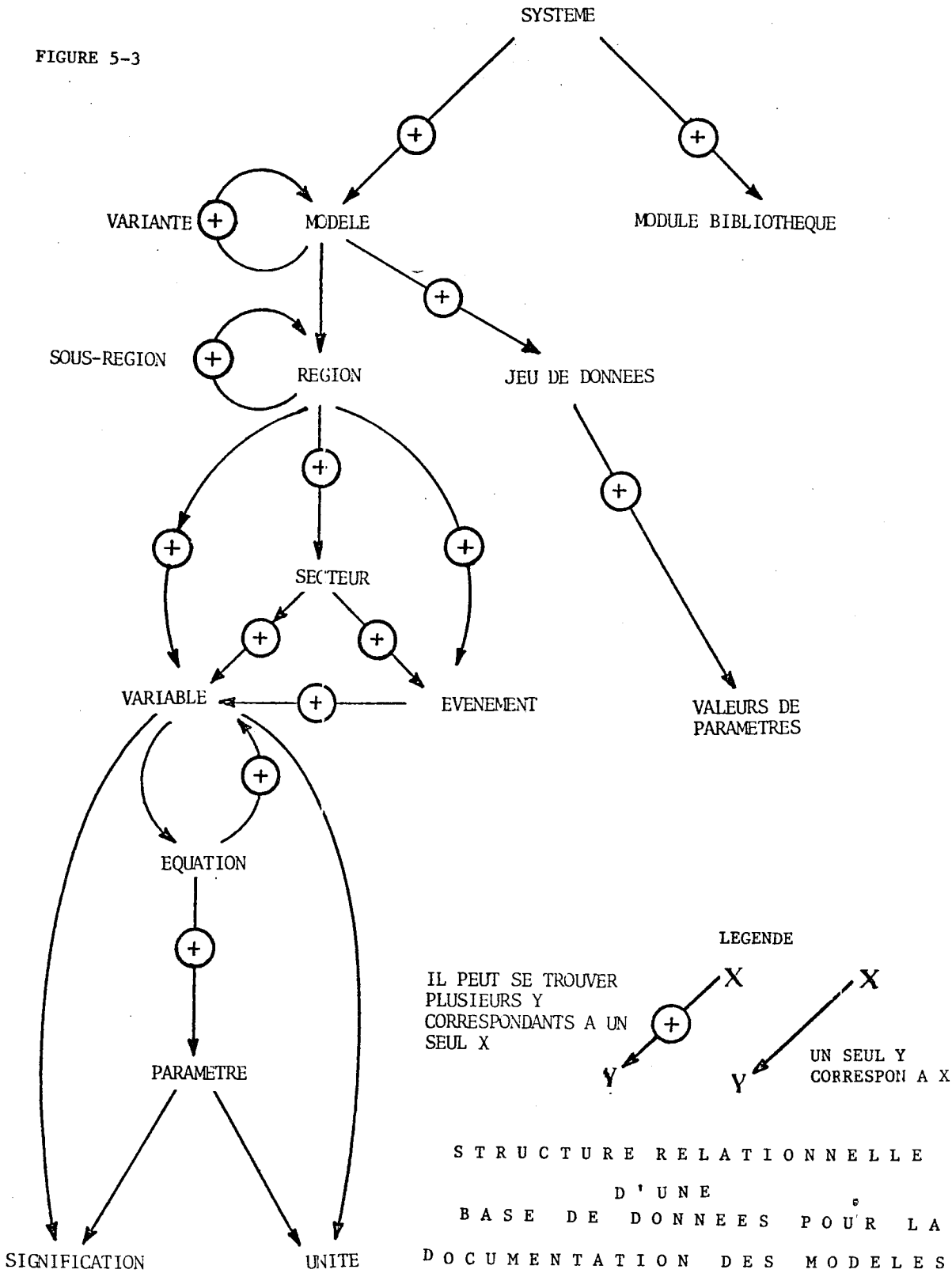


Figure V.3



STRUCTURE RELATIONNELLE D'UNE BASE DE DONNEES POUR LA DOCUMENTATION DES MODELES DECRITS EN "LADESH"

L'exploitation conjointe du système de simulation et d'une base de données peut paraître attractive, par exemple la modification dynamique de données de la base au fur et à mesure que l'on modifie le modèle dans le système. Dans la pratique, ce problème n'est pas facile car une interface compliquée doit être mise en place. Il paraît par contre plus souhaitable d'avoir la documentation dans une base de données à part exploitée en en interrogation et à des fins pédagogiques.

4 . RESTRUCTURATION OU SIMULATION MIXTE ?

L'un des problèmes plus importants que l'on affronte dans la modélisation et la simulation en Sciences Sociales est l'introduction d'événements discrets dans un modèle continu.

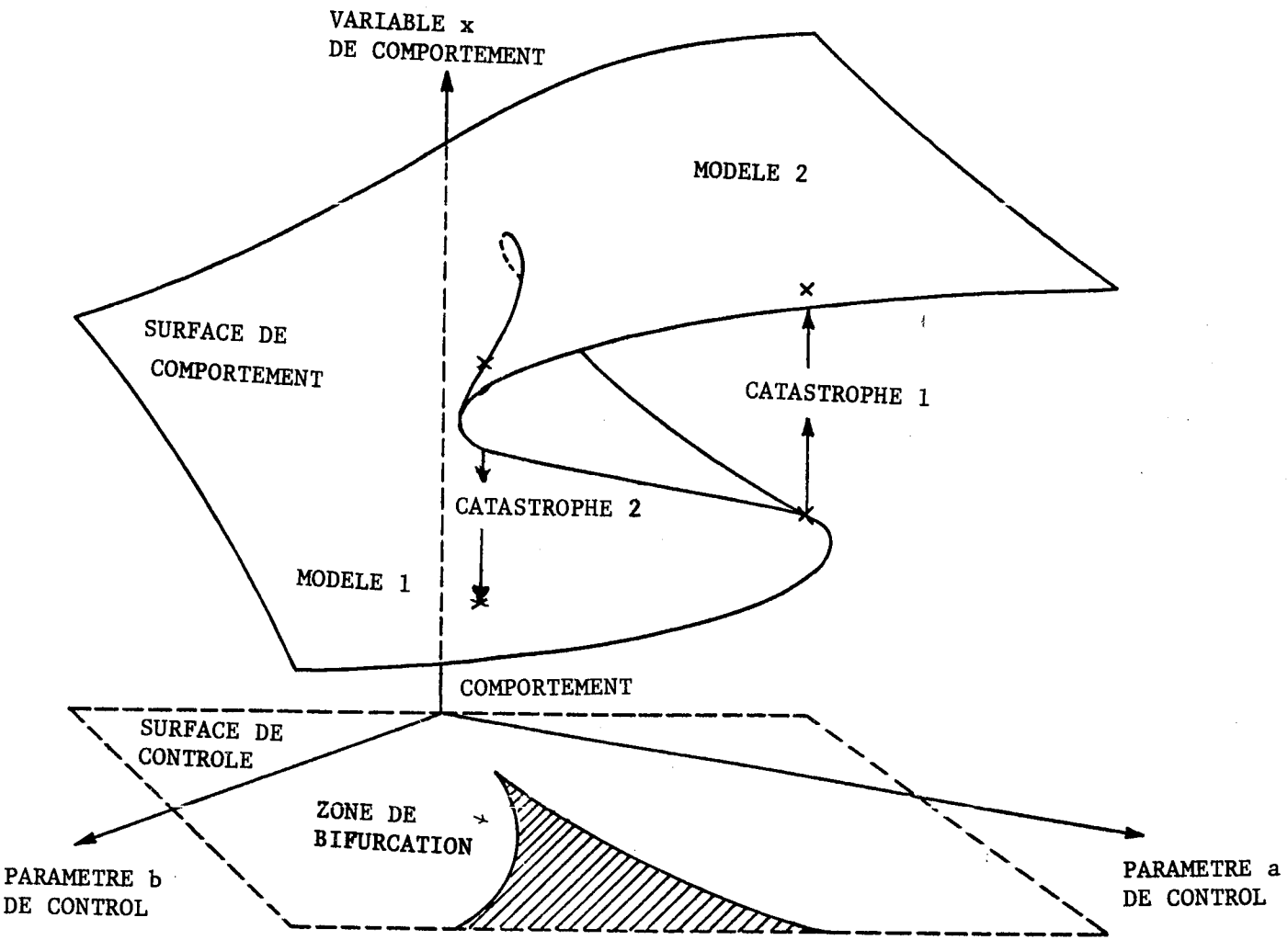
Récemment une nouvelle approche est apparue qui permet de prendre en compte certaines discontinuités dans une approche continue unifiée c'est la théorie des catastrophes du Prof. R. Thom [8] ; certaines applications qualitatives en Sciences Sociales ont été déjà envisagées, principalement dans l'équipe du Prof. Zeemman [7], [8]. L'identification de la "catastrophe élémentaire" n'est pas simple, d'autant plus qu'il faut remplir certaines conditions comme le fait d'associer au comportement observé une fonction de type potentiel dont les paramètres de contrôle sont représentés par les coefficients de la fonction, le comportement étant représenté par les variables.

On va prendre comme exemple un comportement x dont la fonction est $\frac{1}{4} x^4 - ax - \frac{1}{2} bx^2$ les paramètres de contrôle sont "a" et "b", et la surface de comportement est déterminée par tous les points où la dérivée est nulle (voir figure V.4).

C A T A S T R O P H E D E L A F R O N C E (C U S P)

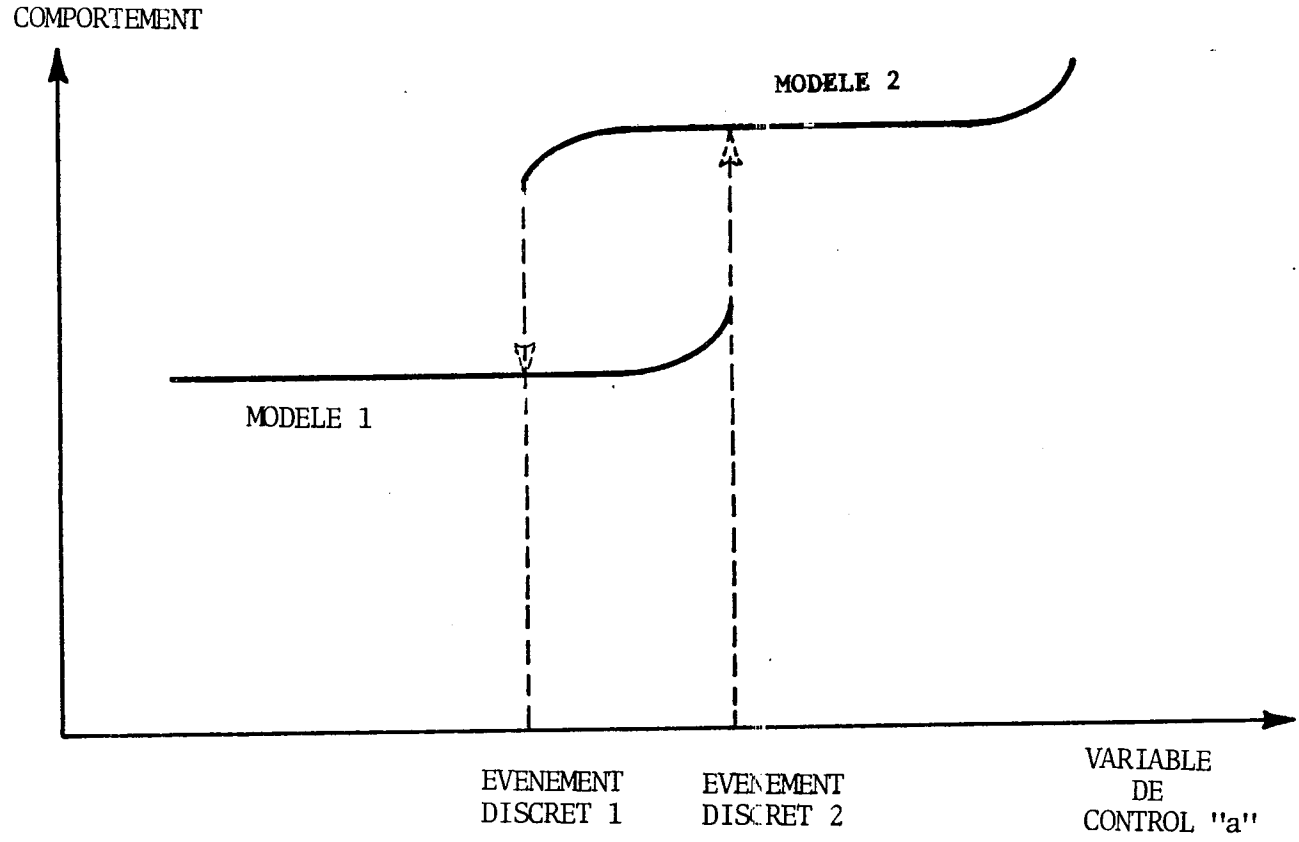
O U D E R I E M M A N - H U G O N I O T

FIGURE 5-4



Si l'on arrive à déterminer qu'une seule variable de contrôle (par exemple le paramètre "a"), alors le système apparaîtra comme fondamentalement discret et donc modélisé par deux (ou plus) modèles et plusieurs événements (voir figure V.5).

FIGURE 5-5



Ainsi modélisés les événements discrets seront compris comme des "catrastrophes" d'un même modèle continu. Malheureusement une généralisation de ce phénomène à toute descontinuité est loin d'être faite (et de pouvoir l'être auusi !). De toute façon la manipulation mathématique n'est pas simple et de nouveaux outils dans la simulation devront être introduits.

A l'heure actuelle deux tendances s'affrontent pour l'introduction des événements discrets dans la simulation continue. La première est l'introduction directe de l'événement discret dans la description au même titre qu'une équation continue (cf. § I.10). C'esr la SIMULATION MIXTE (à ne pas confondre avec la Simulation Hybride). La deuxième tendance préconise l'introduction d'événements discrets dans des niveaux hiérarchiques différents à ceux où la description continue apparaît ; les événements d'un niveau hiérarchique supérieur assurent les interconnexions (valides) entre blocs continus d'un niveau hiérarchique inférieur, pour ainsi dire la description discrète assure une configuration et se charge de restructurer le modèle ; cette approche s'appelle RESTRUCTURATION dans la SIMULATION.

Apparemment la Simulation Mixte est plus générale et n'introduit pas de contraintes dans la simulation. Les problèmes techniques que cela peut impliquer (par exemple les méthodes d'intégration, le ré-ordonnancement des équations ou l'homogénéité du langage de description) chaque système de simulation les a résolus à sa manière [11].

A juste titre, on doit se demander si les événements discrets interviennent dans la compréhension d'un système social au même titre que les relations continues de causalité. Traditionnellement "la simulation discrète ne peut s'appliquer facilement qu'à des systèmes de petites dimensions (par le nombre d'éléments pris en compte) et à un fin niveau d'agrégation" [3]. Or depuis quelques années les sociologues commencent à parler de structures sociales (discrètes) et, ce qui est un progrès, de structures dynamistes, donc d'une étude discrète à un haut niveau d'agrégation. C'est-à-dire, selon G. Balandier [12], d'étudier autre chose que les relations qui constituent la structure, les incompatibilités, contradictions et tensions de toute structure sociale.

La Sociologie Dynamiste s'intéresse donc à l'"aspect transitoire permanent de structuration-destruction : d'où vient la configuration actuelle, vers quoi tend-elle ? Si l'on admet des structures invariantes le problème ne se pose pas. Si l'on tient plutôt pour vrai que la structure du phénomène est en état provisoire d'équilibre entre tensions et forces multiples, la saisie de lois qui constituent son devenir conduit alors à des modèles **explicatifs**, représentant des lois diachroniques du phénomène étudié" [13].

Pour le moment cette approche ne fait que débuter. Cependant, le besoin est déjà ressenti d'étudier ensemble la dynamique de la structure (continu) et de ses transformations (discret) :

"La modélisation des structures *sous-tensions*, reste fondamentalement une théorie du discret et du discontinu, nécessitant sans aucun doute pour certains problèmes son complément par des outils mathématiques traitant du continu ou du quantitatif" [14].

Un des prolongements sans doute le plus important pour la simulation en Sciences Sociales est de permettre le changement d'une structure au cours d'une simulation, le changement étant commandé par les lois "quantifiées" des structures sociales.

5 . SYNTHESE DE L'EXPERIENCE

L'objectif principal, la définition et réalisation d'un système de simulation qui tient compte du processus de simulation et des besoins tels qu'ils sont ressentis à l'heure actuelle en Sciences Sociales, tout particulièrement en Prospective et Analyse de Systèmes, a été atteint. C'est sans aucun doute au niveau des objectifs secondaires (Chapitre II.2) que des améliorations seront nécessaires.

C'est essentiellement du fait d'une connaissance, nécessaire, à la fois des problèmes spécifiques à l'objet des Sciences Sociales, et de ceux résultant de la modélisation dans ce cadre, que la simulation a pu être re-située dans son contexte d'utilisation (Chapitre I).

Fait apparemment paradoxal, c'est l'approche systémique qui a permis de dégager, de clarifier et de définir les étapes du long, lourd et lent processus de simulation, analyse et mise au point d'un modèle. Ainsi leurs liaisons ont été envisagées dans une optique globale mais non pour autant mélangée ! C'est pourquoi on peut parler de SYSTEME de Simulation et surtout en définir un (Chapitre II).

Le prototype du système de simulation présenté, n'a pas, pour le moment, la prétention d'être **compétitif** par ses performances techniques (temps d'exécution, place mémoire occupée, ...), mais plutôt d'offrir une approche méthodique et d'incorporer de nouvelles facilités **non-gadgets** qui aident effectivement à la Simulation en Sciences Sociales. La confrontation avec les objectifs a été le grand critère de sélection d'éléments, d'outils et de facilités à retenir. Les critiques de futurs usagers seront donc les bienvenues.

Il est difficile de conclure, sans signaler de nouveau qu'il reste encore beaucoup à faire dans le domaine de Simulation et Analyse de Systèmes en Sciences Sociales. Une partie de l'effort s'est donc consacrée à la recherche de nouvelles voies, dont une partie est exposée dans ce dernier chapitre.

Finalement, c'est en facilitant le processus d'analyse et simulation, en le démystifiant et en le rendant un peu plus accessible aux chercheurs des Sciences Sociales, que l'on espère avoir contribué **de loin**, à la compréhension de systèmes dynamiques en Sciences Sociales.

BIBLIOGRAPHIE

[1] MOREL B.

"Gestion dynamique de mémoire en FORTRAN (OS/MVT, CP/CMS)"
Note Technique T38, CICG - juin 1976

[2] DREYFUS M.

"FORTRAN IV"
C.I.R.O., Dunod - Paris - 1967

[3] RECHENMANN F.

"Analyse et Modélisation descendantes des systèmes socio-économiques"
Thèse de Docteur-Ingénieur - I.N.P.G. - 1976

[4] BOLOPION A.

"Mise en oeuvre et expérimentation pédagogique d'un système d'enseignement assisté par ordinateur en électronique (ESPACE)"
Thèse de Docteur-Ingénieur - I.N.P.G. - Octobre 1975

[5] BRETON L., JULIEN-LAFERRIERE B., LIGNIER S., MOZZATI M., PERRIAULT, STECKMEYER D.

"Guidage d'étudiants dans des activités autonomes de constructions et de traitements de modèle en ordinateur"
Ecole des Hautes Etudes en Sciences Sociales
Paris, Juin 1976

[6] MEADOW C.T.

"Man-Machine Communication"
John Wiley & Sons, 1970

[7] DELOBEL C.

"Les systèmes de base de données"
Polycopie cours DEA, U.S.M.G., 1975

[8] THOM R.

"Stabilité Structurelle et morphogénése : Essai d'une théorie générale des modèles"
W.A. Benjamin, Inc. 1972 (Ediscience)

[9] ZEEMAN E.C.

"Catastrophe theory"
Scientifc American, pp.65-83, April 1976

[10] CASTI J. SWAIN H.

"Catastrophe theory and Urban Process"
Research Memorandum 75-14
I.I.A.S.A. Autriche, Avril 1975

[11] WORTMAN D.B.

"Simulation with GASP IV. A combined discrete continuous simulation langage"
Pritsker & Associates, Inc. Lafayette, Indiana 1976

[12] BALANDIER G.

"Anthropologie Politique"
P.U.F. - Paris 1969

[13] RIBEILL G.

"Modèles et Sciences Humaines"
METRA, Vol. XII n°2, pp.271-303 - 1973

[14] RIBEILL G.

"Tensions et Mutations Sociales"
P.U.F. - Paris 1974, p.177

- ANNEXE I -

SYNTAXE DU LANGAGE DE COMMANDE SAISYE

SIMULATION ET ANALYSE INTERACTIVES DES SYSTÈMES SOCIO-ECONOMIQUES

1. ENVIRONNEMENT D'ENTREE

< axiome > → < commande > ; | ? | !

< commande > → HELP |
INFO |
NAMES |
OUT |
BYE |
MODEL < identificateur modèle >

< identificateur modèle > → < nom >

2. ENVIRONNEMENT D'INFORMATION

< axiome > → < commande > ; | ? | !

< commande > → HELP |
INFO |
NAMES |
OUT |
BYE |
MODEL < identificateur modèle > |
CHECK |
UPD < corps mise à jour > |
STRUCT |
REPORT |
KEEP RUN < indication run > |
LIST < objets à énumérer > |
PRINT < objets à imprimer > |
OPRINT < objets à imprimer > |
BUILD |
ERASE < objets à effacer > |
DISPLAY RUN < identificateur run >

```
< identificateur modele >  →  < nom >
< corps mise à jour >  → MODEL |
                         VARIANT < identificateur variante >|
                         DOC |
                         DATASET < identificateur jeu de données >|
                         MACRO < identificateur macro >
< indication run >  →  ALL |
                       < identificateur run >< liste identif.run >
< objets à énumérer > -- MODEL |
                         VARIANT |
                         DATASET |
                         RUN
< objets à imprimer > → MODEL |
                        VARIANT <identificateur variante > |
                        DATASET <identificateur jeu de données > |
                        DOC
< objets à effacer >  → MODEL < identif. modele optionnel > |
                        VARIANT < identificateur variante > |
                        DATASET < identificateur jeu de données >|
                        DOC|
                        RUN < identificateur run >
< liste identif.run >  → < identificateur run > < liste identif.run >|
                         < chaîne vide >
< identificateur variante >  → < nombre entier >
< identificateur jeu de données >  → < nombre entier >
< identificateur run >  →  < nombre entier >
< identif.modèle optionnel >  → < identificateur modele >|
                                < chaîne vide >
< identificateur modele >  →  < nom >
< identificateur macro >  →  < nom >
```

3. ENVIRONNEMENT DE CONSTRUCTION

< axiome > → < commande > ; | ? | !

< commande > → HELP |
INFO |
NEWVAR |
NEWDTSET |
VARIANT < identificateur variante > |
DATASET < identificateur jeu de donnees > |
UPD < corps mise à jour >|
CHECK |
LIBRARY < sens > < liste de modules >|
BATCH

< identificateur variante > → < nombre entier >

< identificateur jeu de données > → < nombre entier >

< corps mise à jour > → MODEL|
VARIANT < identificateur variante > |
DOC|
DATASET < identificateur jeu de données > |
MACRO <identificateur macro >

< identificateur macro > → < nom >

< sens > → GET |
OFF |
PUT

< liste de modules > → <identificateur module> <liste identif.modules>

< liste identif.modules > → < identificateur module > < liste identif. modules >|
< chaîne vide >

< identificateur module > → <nombre entier >

4. ENVIRONNEMENT DE MISE A JOUR

< axiome > → < commande > ; | ? | ! |
< macro-définition > ; |
< macro-requête > ;

< commande > → UPD < corps mise à jour > |
HELP|
INFO|
ERASE MACRO < identificateur macro > |
BUILD |
CHECK |
NAMES MACRO |

A <nombre entier> |
L <chaîne> |
D <nombre de fois> |
TOP |
BOTTOM |
GOTO <nombre entier>
LI |
P <nombre de fois>|
U <nombre de fois>|
N <nombre de fois>|
C <chaîne> <chaîne>|
R <ligne>|
I <ligne>

< corps mise à jour > → MODEL |
VARIANT <identificateur variante> |
DOC|
MACRO <identificateur macro> |
DATASET < identificateur jeu de données >

< nombre de fois > → < nombre entier > |
< chaîne vide >

< identificateur variante > → < nombre entier >

< identificateur macro > → < nom >

< identificateur jeu de données > → < nombre entier >

< macro-définition > → MACRO < identificateur macro >
< liste de paramètres formels > :
< liste d'opérations simples >

< liste de paramètres formels > → < paramètre formel >
< liste de paramètres formel > |
< chaîne vide >

< paramètre formel > → & < nombre entier >

< liste d'opérations simples > → < opération simple >
< liste d'opérations simples > |
< chaîne vide >

< opération simple > → L < paramètre formel > |
D < paramètre formel optionnel > |
TOP |
BOTTOM |
LI |
P < paramètre formel optionnel > |
U < paramètre formel optionnel > |
N < paramètre formel optionnel > |
C < paramètre formel > < paramètre formel > |
R < paramètre formel > |
I < paramètre formel >

< paramètre formel optionnel > → < paramètre formel > |
< chaîne vide >

< macro-requête > → < identificateur macro >
< liste de paramètres réels >

< liste de paramètres réels > → < nombre entier >
< liste de paramètres réels > |
< chaîne >
< liste de paramètres réels > |
< ligne >
< liste de paramètres réels > |
< chaîne vide >

5. ENVIRONNEMENT DE VERIFICATION

< axiome > → < commande > ; | ? | !
< commande > → HELP |
INFO |
MODEL |
VARIANT < identificateur variante > |
DATASET < identificateur jeu de données > |
BUILD |
UPD < corps mise à jour > |
GO < variante à simuler > <jeu de données à simuler >
< identificateur variante > → < nombre entier >
< identificateur jeu de données > → < nombre entier >
< corps mis à jour > → MODEL |
VARIANT < identificateur variante > |
DOC |
MACRO < identificateur macro > |
DATASET <identificateur jeu de données >
< identificateur macro > → < nom >
< variante à simuler > → MODEL |
VARIANT < identificateur variante >
< jeu de données à simuler > → DATASET < identificateur jeu de données >

6. ENVIRONNEMENT DE SIMULATION

<axiome > → < commande > ; | ? | !
<commande > → HELP |
INFO |
EULER < calcul erreur opt.>
< ajustement précision opt.> |
TRAPEZ < calcul erreur opt.> |
< ajustement précision opt.> |

SIMPS < calcul erreur opt.>
< ajustement precision opt.>|

HEUN < calcul erreur opt.>
< ajustement precision opt.>|
CAUCHY < calcul erreur opt.>
< ajustement precision opt.>|
PREDCO < calcul erreur opt.>
< ajustement precision opt.>|
EXPON < determination min-max >|
DT < nombre reel >|
FROM < date > TO < date > RUN
< identificateur run >|
CHECK

< calcul erreur opt.> → ERR |
< chaîne vide >

< ajustement precision opt.> → PREC < nombre reel > % < pas minimal> |
< pas minimal > PREC < nombre reel > % |
< chaîne vide >

< determination min-max > → < pas minimal > < pas maximal >|
< pas maximal > < pas minimal >

< pas minimal > → DTMIN < nombre reel >

< pas maximal > → DTMAX < nombre reel >

< nombre reel > → < nombre entier >.< rien ou entier >|
. < nombre entier >

< rien ou entier > → < nombre entier > |
< chaîne vide >

< date > → < nombre entier >

< identificateur run > → < nombre entier >

< identificateur run opt.> → < identificateur run > | < chaîne vide >

7. ENVIRONNEMENT DE VISUALISATION

< axiome > → < commande > ; | ? | !

< commande > → HELP |
INFO |
CHECK |
KEEP RUN < identificateur run >|
UPD < corps mis à jour > |

OTHRUN < identificateur run > |
MINMAX < liste de variables > |
MEAN < liste de variables > |
TITLE < ligne > |
COMPAR < variables > < moyen + identif. run > |
OCOMPAR < variable > < moyen + identif. run > |
HISTO < plage > < liste de variables > |
OHISTO < plage > < liste de variables > |
FROM < date > TO < date>< display >|
<display >

< identificateur variante > → < nombre entier >

< identificateur jeu de données > →< nombre entier >

< corps mis à jour > → MODEL |
VARIANT < identificateur variante >|
DOC|
MACRO < identificateur macro >|
DATASET < identificateur jeu de données >

< identificateur macro > →< nom >

< identificateur run > → < nombre entier >

< liste de variables > → < variable >< suite de variables >

< suite de variables > → < variable >< suite de variable >|
< chaîne vide >

< variable > → < nom >

```
< liste identif.run >   →  < nombre entier > < suite de nb entiers >
< suite de nb.entiers > →  < nombre entier > < suite de nb.entiers >|
                           < chaîne vide >
< moyen + ident.RUN >   →  HISTO < plage > RUNS
                                        < liste identif.run >|
                           PLOT RUNS < liste run=lettre >
< plage >               →  MEAN |
                           < date > |
                           < chaîne vide >
< liste run=lettre > → < run=lettre >< suite run=lettre >
< suite run=lettre > → < run=lettre >< suite run=lettre >|
                       < chaîne vide >
< run=lettre >  →  < identificateur run > = < identificateur lettre >
< date >        →     < nombre entier >
< identificateur lettre > → < nom >
< display >     →     ZOOM < augmentation/diminution >
                           < liste objets à tracer >|
                      OZOOM < augmentation/diminution >
                            < liste objets à tracer >|
                      PLOT  < liste objets à tracer >|
                      OPLOT < liste objets à tracer >|
                      BISPACE  < variable > < variable > |
                      OBISPACE < variable > < variable > |
                      PRINT < liste de variables >|
                      OPRINT < liste de variables >

< augmentation/diminution > →  + < nombre entier >|
                               - < nombre entier >|
                                 < nombre entier >
```

< liste d'objets à tracer > → < liste à échelle uniformisée >|
< liste à échelles indépendantes >

< liste à échelle uniformisée > → UNIF < liste variable=lettre >

< liste variable=lettre > → < variable=lettre >
< suite variable=lettre >

< suite variable=lettre > → < variable=lettre >
< suite variable=lettre >|
< chaîne vide >

< liste à échelles indépendantes > → < objet à tracer >
< suite d'objets à tracer >

< suite d'objets à tracer > → < objet à tracer >
< suite d'objets à tracer >|
< chaîne vide >

< objet à tracer > → < variable=lettre > < échelle optionnelle >

< variable=lettre > → < variable > = < lettre >

< échelle optionnelle > → (< nombre entier > < nombre entier >)|
< chaîne vide >

NOTAS

1. Le blanc et la virgule sont les séparateurs universels entre unités syntaxiques.

2. Les caractères non alphanumériques n'ont pas besoin de séparateurs pour les identifier, cela donne une souplesse supplémentaire au langage et permet l'écriture habituelle de certaines expressions, par exemple :
   - un paramètre formel pourra être écrit comme &4 et non pas seulement & 4
   - l'indication du ZOOM : OZOOM+3 et non pas seulement OZOOM + 3
   - l'indication de l'échelle peut s'écrire : (2000,3000) et non pas seulement (2000 , 3000)

3. Une chaîne et n'importe quelle chaîne de caractères y compris le blanc et la virgule, compris entre deux apostrophes, la longueur de la chaîne est limitée à 32 caractères. Par exemple :

   'POP=IND-MOR;'

4. Une ligne et n'importe quelle chaîne de caractères compris entre deux symboles dollar. La longueur d'une ligne est limitée à 69 caractères.

5. L'unité syntaxique variable est pour l'instant prise comme l'unité < nom > (chaîne alphanumérique commençant par une lettre et, limitée à 8 caractères).
   Evidemment la définition de variable dépend de chaque langage de modélisation, une révision de cette définition devra alors être faite.

6. L'unité syntaxique lettre a été aussi prise comme un < nom >, sachant que c'est la dernière lettre du nom qui sera prise comme la lettre effective.

7. Les délimitateurs, apostrophe et dollar ne peuvent pas faire partie de la chaîne ou la ligne respectivement.
   Par exemple :

   'i $ IND = SIN(TRAP) $ ' est seulement une chaîne.

- ANNEXE II -

UNE PAGE, SA COMPOSITION ET SA TAILLE

Une page mémoire est composée d'une zone standard appelée "corps de la page" et d'une zone annexe à chaque page appelée "zone de débordement" (Voir Fig.1).

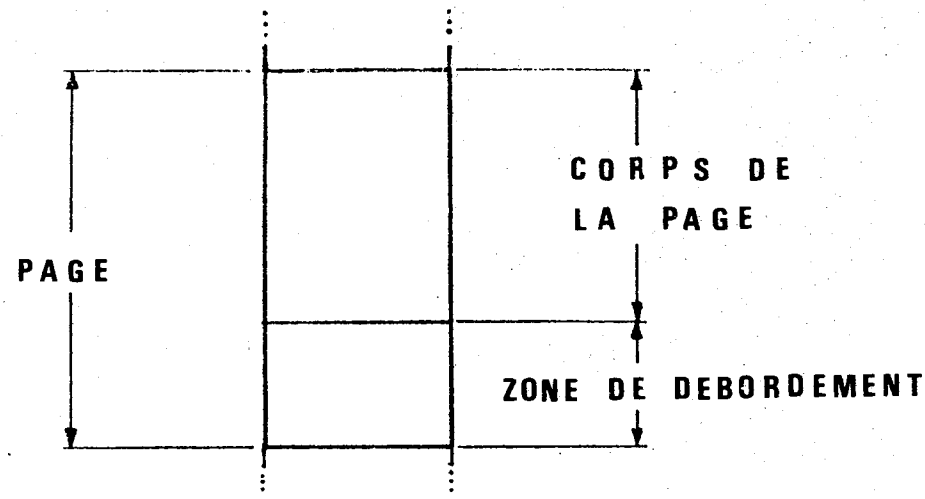


Figure 1

Le nombre de pages, la taille de la page, et de la zone de débordement ne doivent pas être choisis de façon aléatoire, leur choix est fonction des type et taille des objets manipulés. Malheureusement la taille des objets varie d'une application à une autre. On devrait donc faire une étude statistique sur la taille des objets et à partir de cela déterminer la composition et la taille d'une page.

L'objet de cette annexe est de montrer les relations qui peuvent exister entre les tailles des objets, des pages et des zones, ainsi que d'esquisser une solution satisfaisant certaines contraintes.

La taille maximale d'un objet (modèle, variante, jeux de données, documentation) est limitée par le nombre de pages multiplié par la taille de chaque page.

Si l'on considère la taille maximale d'un objet comme une donnée (par exemple à travers des statistiques) et que l'on fixe la taille d'une page par un critère quelconque, on obtient le nombre de pages. Un critère pour définir la taille d'une page (corps) peut être le suivant : la probabilité pour qu'un module faisant partie d'un objet (par exemple un secteur par rapport au modèle) soit contenu entièrement dans une page, doit être au moins égal à 1/2. Ceci est justifié car la plupart des opérations de mise à jour sont effectuées à l'intérieur d'un module.

PROPOSITION :

La probabilité p pour qu'un module de taille t_m soit contenu entièrement dans une page quelconque de taille t_p est égal à :

$$p = \frac{t_p - t_m + 1}{t_p} .$$

En effet :

Si $t_m = t_p$ alors $p = \frac{1}{t_p}$

Si $t_m = t_p - 1$ alors $p = \frac{2}{t_p}$

⋮

Si $t_m = 1$ alors $p = 1$

Pour t_p suffisamment grand, un bon choix peut être : $t_m = \frac{t_p}{2}$, car

$$p = \frac{t_p + 2}{2\, t_p} > \frac{1}{2}$$

La zone de débordement sert exclusivement à l'éditeur (environnement de mise à jour) lorsque dans une page déjà pleine l'on veut insérer des lignes.

On a donc intérêt à minimiser la taille de la zone de débordement (que l'on suppose plus petite que la taille du corps de la page) car le but même de la pagination est de ne pas encombrer inutilement la mémoire (dû à la réservation statique de place mémoire); d'autre part on a intérêt à ce que la taille de la zone de débordement soit suffisamment grande pour insérer "une quantité raisonnable de lignes", sans avoir besoin d'appeler en mémoire la page suivante ou faire exécuter le ramasse-miettes (cas où l'on est à la dernière page).

La taille de la zone de débordement doit être telle que si les n-1 premières zones de dépassement sont totalement occupées (où n=nombre maximal de pages) alors on n'excède pas la taille maximale du modèle. Or il y a occupation d'une zone de débordement si et seulement si le corps de la page est occupé totalement. On a donc :

$$(n-1)(t_p + t_d) \leq n.t_p$$

d'où la contrainte :

$$t_d \leq \frac{t_p}{n-1}$$

comme la taille doit être un nombre entier et qu'on veut qu'elle soit maximale tout en respectant la contrainte on obtient :

$$t_d = \lfloor \frac{t_p}{n-1} \rfloor$$

où $\lfloor X \rfloor$ est le plus grand nombre entier inférieur ou égal à X.

- A N N E X E I I I -

LE MECANISME MACRO

LES MACROS

1. GENERALITES ET DEFINITION

Dans le travail présent on va dire qu'une macro est un mécanisme qui permet d'associer à une commande spécifique une suite d'éléments de base du langage de commande avec éventuellement transformation de certaines chaînes en éléments syntaxiques du langage de base. Ceci est une adaptation particulière de la définition du macro-syntaxique généralisé par Leavenworth : "un macro-syntaxique défini de transformations de chaînes basées en éléments syntaxiques du langage de base "(1).

L'introduction des macros dans notre système conversationnel doit être tout à fait cohérent avec le reste du langage de commande. Comme toutes les macros, elles sont formées d'une structure (qui décrit la syntaxe du texte source qui va être reconnue comme macro) et une définition (qui décrit la sémantique de la macro correspondante).

L'utilisation des macros étant une possibilité simple d'extension syntaxique du langage de commande est une idée séduisante pour l'informaticien, néanmoins pour le non-informaticien elle représente parfois un nouveau langage à apprendre. Dans un but pédagogique c'est-à-dire pour aider le non-informaticien à utiliser de telles possibilités on a décidé d'adopter comme principe que aussi bien la déclaration de structure et définition de macro que tout appel d'une macro doivent être aussi des commandes du langage de base; il faut donc respecter la syntaxe générale d'une commande :

< commande > → < nom de la commande >
< liste de paramètres > ';'

de même avoir une syntaxe type LL(1) et donc qui peuvent être reconnus par le même analyseur syntaxique que le langage de commande.

Une macro-syntaxique va être déclarée comme MACRO U : V ; où U est la structure et V la définition.

Le problème de cohérence est donc réduit à trouver une expression syntaxique générale aussi bien pour U que pour V et pour toutes les macros possibles. Ces deux contraintes vont nous amener à la solution suivante (*) :

U → < nom macro > < liste de paramètres possibles >

V → < nom fonct.de base > < liste de paramètres formels >< suite >

< suite > → / < nom fonct.de base >< liste de paramètres formels > < suite > | ε

Malheureusement la < liste de paramètres possibles > n'a pas de sens syntaxique précis, puisqu'ils peuvent être des chaînes, des noms, des nombres, des lignes; c'est-à-dire tout élément syntaxique qui peut être reconnu par l'analyseur syntaxique à l'exception des caractères non alphanumériques isolés.

Afin de résoudre cela, et pour introduire une correspondance souple entre les paramètres de la structure et ceux de la définition, on a défini :

< liste de paramètres possibles > = < liste de paramètres formels >

étant entendu que :

< liste de paramètres formels > → < paramètre formel > < liste de paramètres formels > | ε

C'est-à-dire la structure de la macro sert à indiquer :

1°/ le nom qui définit la macro

2°/ le nombre de paramètres formels que doit avoir cette macro

3°/ la correspondance entre le nom des paramètres formels qui apparaissent dans la structure et le nom de paramètres formels de la définition.

(*) *Le métasymbole 'ε' veut dire la chaîne vide*

| indique la séparation entre alternatives.

Par contre la structure ne pourra pas servir à identifier syntaxiquement les paramètres; lors de l'appel à une macro, le système conversationnel fera appel à un macro-processeur qui se chargera de reconnaître le nom de la macro, le nombre de paramètres réels employés, et de les faire comparer avec le nombre de paramètres formels, de remplacer dynamiquement la commande macro par sa définition avec remplacement des paramètres.

C'est donc au moment de l'exécution de chacun des composants de la définition de la macro appelée que l'analyseur syntaxique général du système fera l'analyse syntaxique sur les paramètres déjà remplacés. En conséquence l'analyse syntaxique sur la nature des paramètres formels employés dans une définition macro ne pourra pas être faite mais l'analyse syntaxique sera faite lors de l'appel à la macro; donc il est laissé à l'utilisateur le soin de bien définir sa macro. En tout cas un appel incorrect à une macro sera détecté incorrect et empêchera ainsi toute exécution des commandes dont les paramètres sont incorrects.

Dans ce sens on offre des possibilités semblables à ce qu'on obtient avec l'utilisation des procédures ou sous-routines vis-à-vis de l'utilisateur; car "un sous-programme est évalué lors de l'exécution du programme tandis que l'évaluation d'un appel de macro est fait dans une phase préliminaire à cette exécution" (2). Notre mécanisme est néanmoins différent de celui des sous-routines puisque le texte d'un sous-programme doit être connu en face du compilateur tandis que dans notre cas, dû à l'existence de mécanismes de substitution de chaînes de caractères (typiquement macro) le texte n'est connu que lors d'une telle substitution après l'appel à la macro. On ne fait pas d'appel par nom des paramètres (qui est une transmission de l'adresse où se trouve l'argument effectif) comme pour les sous-routines, mais une substitution effective de paramètres. C'est pour cela qu'on a préféré garder l'appellation de macro.

Il peut exister un conflit si on permet que le nom de la macro soit le premier élément d'un appel macro puisque le nom de la macro peut coïncider avec le premier nom d'une commande; en effet cette possibilité est exclue des langages du type LL(1) pour lesquels pour toutes les règles (pour toutes les classes syntaxiques) les premiers éléments terminaux des différentes alternatives de génération d'une même classe syntaxique doivent être différents. Trois solutions pour résoudre ce conflit :

1°/ Interdire dans l'utilisation du nom-macro, un mot réservé du langage de commande,

2°/ Déterminer a priori comme ordre d'examen syntaxique d'abord les noms de macros puis les éléments terminaux du langage de base,

3°/ Imposer à la structure de commencer par un élément terminal qui ne génère pas de conflit (par exemple un mot clé : CALLMAC).

La solution retenue est un compromis entre la souplesse et la structure du système; en effet la structure de l'analyse lexicographique nous impose un ordre d'examen des éléments terminaux du langage puisque c'est au niveau lexicographique que l'on détermine si un mot est un mot réservé du langage ou non. Au niveau syntaxique l'ordre d'examen pour les identificateurs est d'abord mots réservés; et si ce n'est pas un mot réservé, c'est donc un mot quelconque; donc si on veut utiliser le même analyseur syntaxique que pour les commandes, on doit interdire l'utilisation d'un mot réservé comme un nom de la macro. Cette contrainte nous permet par contre de ne pas imposer à la structure d'une macro de commencer par un élément terminal unique.

< macro-définition > → MACRO < nom macro >< liste de paramètres formels >
< liste non vide d'opérations simples >;

< liste de paramètres formels > → <paramètre formel >
< liste de paramètres formels > | ε

< liste non vide d'opérations simples > → < opération simple >
< liste d'opérations simples >

< liste d'opérations simples > → / < opération simple >
< liste d'opérations simples > | ε

Donc un appel macro ou macro-requête a la forme syntaxique de la structure effective de la macro :

< macro-requête > → < nom macro > < liste de paramètres réels >;
< liste de paramètres réels > → < paramètre réel > <liste de paramètres réels > | ε

Le nom d'une macro doit avoir les caractéristiques essentielles de l'unité syntaxique NOM, donc :

< nom macro > → NOM

Il est clair qu'il y aura une correspondance biunivoque entre les paramètres formels qui apparaissent dans la structure de la définition MACRO et les paramètres effectifs de la macro-requête; la correspondance étant faite par la position des paramètres, c'est à l'utilisateur de respecter l'ordre et le nombre de paramètres lors d'un appel macro.
Afin de repérer facilement les paramètres formels on a décidé d'utiliser une syntaxe simple en utilisant un nombre, mais de façon à pouvoir distinguer un nombre d'un paramètre formel dans la définition (corps) de la macro, le nombre est précédé comme dans le cas de EXEC-CMS par un caractère spécial '&'. D'où la définition de paramètre formel :

< paramètre formel > → '&' nombre

Il ne nous reste à définir que < opération simple > et < paramètre réel > .

Il est clair qu'un macro-processeur est associé à un langage, les langages étant définis pour chaque environnement, il faut considérer la possibilité d'introduire un macro-processeur dans chaque environnement, le mécanisme pouvant être le même; ce qui va caractériser le macro-processeur pour chaque environnement est la définition de < opération simple > et < paramètre réel > .

II. REALISATION

Il est vrai que les macros permettent de personnaliser le langage de commande, d'étendre le langage en ajoutant des requêtes qui sont combinaison d'autres requêtes, et de permettre d'économiser le temps d'utilisation en enchaînant automatiquement plusieurs commandes; mais l'utilisation fondamentale des macros est d'éviter la répétition fastidieuse (et en conséquence les risques d'erreur) à l'écriture d'une commande ou d'un groupe de commandes dont les paramètres peuvent éventuellement différer, il est clair que c'est ce dernier cas qui est plus intéressant, c'est-à-dire pouvoir regrouper plusieurs commandes en une seule.

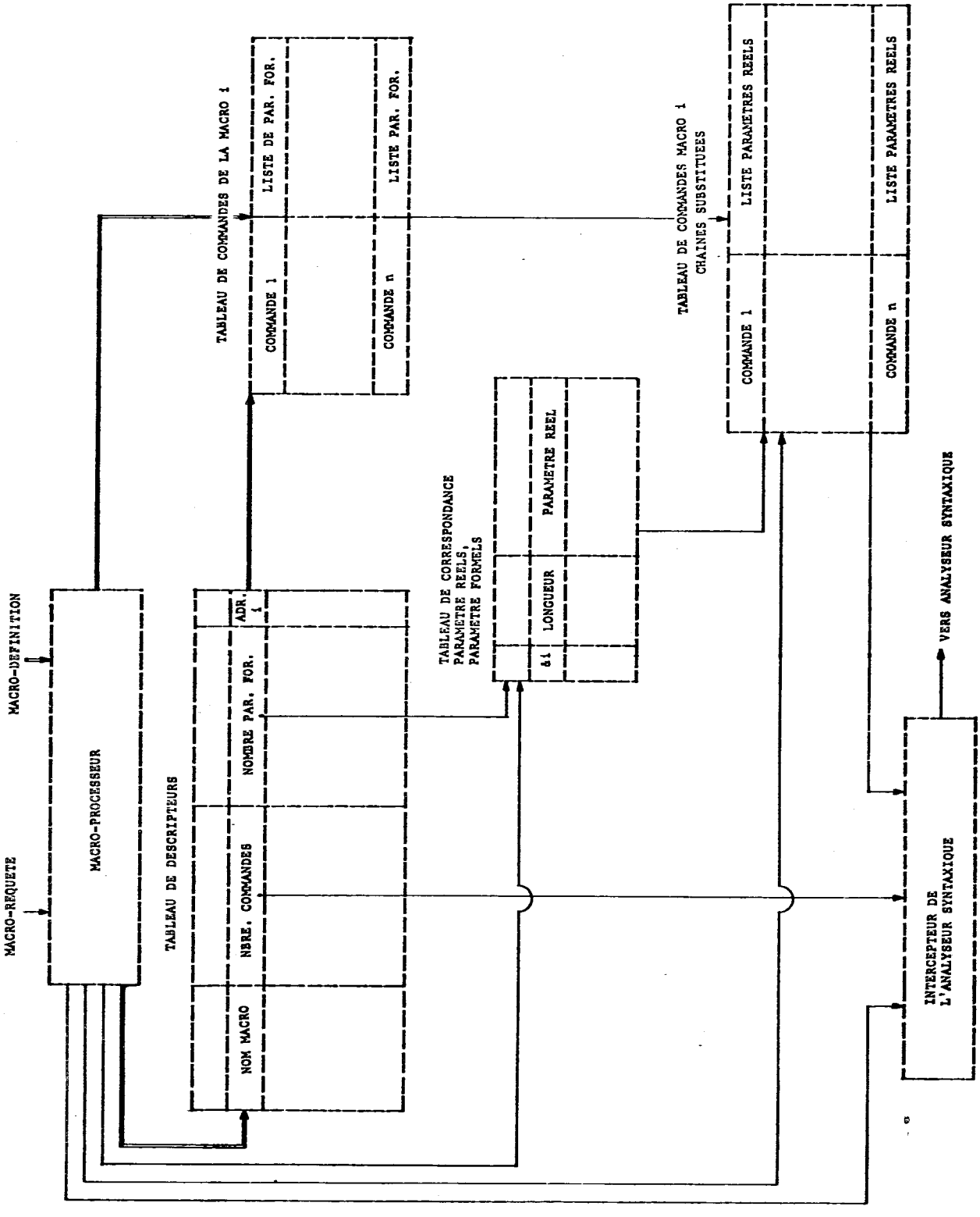
Parmi tous les environnements qui composent le système conversationnel, le seul où la répétition des commandes (dans l'environnement même) devient intéressant est l'environnement de mise-à-jour; d'autant plus que la possibilité de considérer une macro comme une commande quelconque nous permet dans l'environnement en question de répéter plusieurs fois la même suite d'opérations en combinaisen avec la commande de répétition "again".
Par exemple "répéter n fois la macro suivante : localiser l'équation X, remplacer dans celle-ci la chaîne de caractères Y par la chaîne Z, et indiquer le numéro de ligne".

La définition du macro-processeur se concrétise en définissant exclusivement < opération simple > et < paramètre réel >. Pour l'environnement de mise-à-jour nous avons décidé de ne grouper ou répéter que les commandes intéressantes. Néanmoins la définition de macro-processeur étant faite de façon abstraite, une extension ou modification peut être faite de manière très simple en re-définissant <opération simple > et < paramètre réel >. Notre volonté est pour une re-définition du mécanisme macro à travers les commentaires et suggestions des utilisateurs du système.

La présente version (prototype expérimental) comporte pour l'environnement de mise-à-jour les définitions suivantes :

```
<opération simple > → 'L'  < paramètre formel >< paramètre formel >|
                      'D' |
                      'TOP'|
                      'BOTTOM'|
                      'GOTO'    < paramètre formel > |
                      'LI' |
                      'P'  < option paramètre formel >|
                      'U'  < option paramètre formel >|
                      'N'  < option paramètre formel >|
                      'C'  < paramètre formel ><paramètre formel >|
                      'R'  < paramètre formel >|
                      'I'  < paramètre formel >|
                      'I'  < paramètre formel >

< paramètre réel > →  'NOM' |
                      'NUMERO'|
                      'CHAINE'|
                      'LIGNE'|
                       ε
```

MISE EN OEUVRE DU MACRO-MECANISME



L'implémentation du macro-processeur dans le système conversationnel a été réalisé comme le montre schéma ci-dessus.

Lors de la déclaration d'une macro (commande MACRO) les liaisons doubles sont établies, c'est-à-dire on crèe une entrée dans la table de descripteurs qui contient : le nom de la macro, le nombre de commandes que comporte la macro, le nombre de paramètres formels employés et finalement l'adresse où se trouve le tableau de commandes de la macro en question; ce tableau de commande comporte le nom de la commande et la liste de paramètres formels assignés à chaque commande.

Lors de l'appel macro on établit d'abord la correspondance paramètres formels-paramètres réels en spécifiant la longueur (en nombre de caractères) du paramètre réel, grâce à ce tableau de correspondances on procède à la substitution de chaînes, en recréant le tableau de commandes, finalement un intercepteur de l'analyseur syntaxique principal se charge d'envoyer commande par commande (déjà substituées) à l'analyseur syntaxique qui se chargera d'analyser les commandes et d'exécuter les fonctions sémantiques résultantes. L'interception étant faite lorsque l'analyseur syntaxique demande une nouvelle lecture à la console. Il est clair d'autre part que l'intercepteur doit se trouver au niveau du superviseur général du système conversationnel, tandis que le reste des opérations sont faites par facilité comme le résultat des appels aux fonctions sémantiques appelées par l'analyseur syntaxique lors de la définition ou requête macro.

IV. QUELQUES LIMITATIONS

Les macros telles que nous les avons définies et implémentées représentent des limitations à un mécanisme général de macro dans le sens que notre macro-processeur :

- N'a pas d'instructions de rupture de séquence, ni d'étiquettes pour réaliser des boucles. Cela afin d'éviter des instructions propres à la macro autres que les commande et de ne pas permettre des notions algorithmiques propres à un langage de programmation classique.

- Ne permet pas de récursivité d'une part; son utilisation requiert un certain entrainement, d'autre part sa définition exige des instructions de test et retour qui n'existent pas.

- N'a pas de possibilité d'imbrication : une macro ne peut pas faire appel à d'autres macros; l'intérêt d'une telle possibilité est réduit dans le sens qu'il est toujours possible d'écrire l'expansion de la macro appelée par la macro courante. Néanmoins son implémentation est très facile, en ajoutant simplement un système de piles.

- N'offre pas la possibilité de test sur la façon dont se déroule une commande.

- Ne permet pas l'utilisation de variables locales ou globales.

- Le nombre des macros et le nombre d'opérations simples à l'intérieur de la macro, aussi bien que le nombre de paramètres possibles ne pourront pas excéder un maximum , cela est dû aux contraintes d'allocation statique de mémoire en Fortran.

- A ces limitations,il faut ajouter les contraintes déjà signalées dont la plus importante est de ne pas faire l'analyse syntaxique sur la nature des paramètres formels.

V. UN MECANISME GENERAL DE MACROS

Le problème des extensions syntaxiques en général et d'un mécanisme général de macros en particulier, c'est-à-dire un mécanisme sans les contraintes énoncées au paragraphe précédent ont été déjà étudiées pour les langages de programmation par Leavenwort (1), Jorrand (3), Vidart (4) et autres. A travers ces travaux on voit que l'apparition de nouvelles règles de production de la grammaire d'un langage sans les restrictions simplistes que l'on avait faites, a besoin d'un macro-mécanisme de re-définition ou extension de la grammaire totalement intégré au système; quand le support est conversationnel le système peut devenir très lourd et lent. Une continuation de ce travail dans ce sens devrait être poursuivi.

VI. QUELQUES EXEMPLES D'UTILISATION

```
DEFINITION :
        MACRO CAMBIO &2, &3, &1, &4 :
        TOP / L &2 &3 / C &4 &1 / LI ;

REQUETE :
        CAMBIO EVENT, FAMINE, 'PROD', 'AGRIC' ;
```

Cette macro a pour effet de localiser l'évènement "famine" et dans cet évènement changer la chaîne "agric" par "prod" et imprimer le numéro de la ligne sur laquelle le changement a été réalisé.

```
DEFINITION :
        MACRO ALLERA &1, &2, &3 :
        GOTO &1 / R &2 / I &3 / N / D / P ;
```

REQUETE :

```
ALLERA 35    , $PROD=P-X+Q-D;$
   $POP = T - Y * R; $ ;
```

Cette macro a pour effet :

- de remplacer la ligne 35 par la ligne
  PROD=P-X+Q-D;
- d'insérer à la suite (ligne 36) la ligne POP=T-Y*R;
- de supprimer l'ancienne ligne 36
- de conserver et imprimer la ligne 37 (et suivantes) dans leur état initial

REFERENCES

(1) LEAVENWORTH B.M.

"Syntax Macros and Extended Translation".
C.of ACM Vol 9/Nb 11/Novembre 1966,
pp 790-793.

(2) CREVEUIL M.

"Macro-mécanisme pour le Langage de Commande d'un système conversationnel".
Thèse 3e Cycle Informatique USMG.
Grenoble, Juillet 1974.

(3) JORRAND P.

"Contribution au développement des langages extensibles".
Thèse d'Etat USMG. Grenoble, Janvier 1975.

(4) VIDART J.

"Extensions syntaxiques dans un contexte LL(1)"
Thèse 3e Cycle Informatique USMG, Grenoble,
Septembre 1974.

- ANNEXE IV -

LES METHODES D'INTEGRATION

LES METHODES D'INTEGRATION IMPLEMENTEES

Les notions de base sont présentées, afin que l'utilisateur de Sciences Sociales puisse utiliser efficacement les diverses méthodes d'intégration en simulation. Il est clair que pour "l'initier" ce sont des choses connues, qu'il peut sauter sans nuire à la compréhension du reste.

I . DEFINITION DU PROBLEME

Soit l'équation différentielle de premier ordre

$$\frac{dy}{dt} = F(y,t) \quad \ldots\ldots \text{(A)} \quad \text{avec la condition initiale } y(t_o)=y_o.$$

Si la fonction F est régulière dans l'intervalle $[t_o, t_o+\ell]$ il existe une et seulement une solution vérifiant l'équation différentielle et sa condition initiale.

La solution à ce problème est exprimée par

$$y(t) = \int_{t_o}^{t_o+\ell} F(y,t)\, dt \quad \ldots\ldots \text{(B)}.$$

Il s'agit alors de résoudre l'intégrale (B).
Nous rappelons que c'est précisément l'impossibilité de trouver une méthode générale par l'obtention d'une fonction analytique comme résultat de l'intégrale qui a poussé à trouver des solutions approchées, non-analytiques, grâce aux diverses méthodes d'intégration numérique.

L'équation différentielle aux conditions initiales (A) étant le coeur d'un modèle en Dynamique des Systèmes (voir Ch 1) on a besoin de sa solution pour connaître l'état du système, car elle rend compte du processus d'accumulation provenant des divers flux à variation continue. C'est l'évolution de ces flux qui détermine l'évolution du système ainsi décrit.

II . QUELQUES CONCEPTS

Nous éviterons l'excès de formalisme mathématique pour les concepts suivants, une approche rigoureuse peut être trouvée dans la plupart des manuels d'analyse numérique (1), (2), (3) et (4).

La plupart des méthodes d'approximation sont des méthodes dites "à pas", c'est-à-dire on divise l'intervalle d'intégration $[t_o, t_o+\ell]$ en sous intervalles $[t_o, t_o+h_1]$, $[t_o+h_1, t_o+h_1+h_2]$, ..., $[t_o+h_1+\ldots+h_{n-1}, t_o+h_1+\ldots+h_n]$ où $h_1+h_2+\ldots+h_n=\ell$, $t_o+h_1+\ldots+h_n=t_n$

Y étant connu pour t_o (condition initiale). S'il existe une méthode (grâce à une relation de récurrence) pour calculer la valeur approximative de $Y(t_o+h_1)$ à partir de la valeur de $Y(t_o)$, on peut répéter le même raisonnement pour calculer $Y(t_o+h_1+h_2)$ à partir de $Y(t_o+h_1)$, et ainsi de suite, jusqu'à $Y(t_n)$.
On dit qu'on intégre numériquement l'équation (A). h_i est la longueur du i-ème pas d'intégration ou simplement le i-ème pas d'intégration. Il est clair que les h_i, i=1,...n peuvent être différents, mais la majorité dés méthodes que nous présentons sera à pas d'intégration constant.

CONVERGENCE :

une méthode est convergente si pour la même valeur initiale $Y_o=Y(t_o)$, la solution approximative Y_n tend à s'approcher de la solution réelle $Y(t_n)$ quand le pas d'intégration h tend vers 0.

ERREUR :

l'erreur dûe à l'approximation au n-ième pas est
$e_n=Y(t_n)-Y_n \quad (e_o=o)$

STABILITE FORTE :

une méthode est fortement stable si pour n'importe quel point t_k ; l'erreur e_k n'augmente pas quand h tend vers 0.

STABILITE FAIBLE :

une méthode est faiblement stable si pour un même pas d'intégration h quelconque, l'erreur au K-ième pas e_K n'est pas plus grande que les erreurs obtenues aux pas postérieurs au K-ième pas (c'est-à-dire pour tout $K > 0$, $K \in Z$ $e_K \leq e_{K+1}$).

On note que :
La convergence implique toujours la stabilité forte, mais la réciproque n'est pas toujours vraie.
Si une méthode est fortement stable, il peut y avoir des instabilités faibles ; c'est-à-dire que même si l'erreur e_K reste bornée quand $h \to 0$, pour un h fixe quelconque alors l'erreur est plus grande que la solution réelle $Y(t_K)$ quand $t_K \to \infty$.

Même, si une méthode est convergente et faiblement stable, si h n'est pas suffisamment petit la solution approximative peut différer beaucoup de la solution réelle quand t_K est grand. Ce phénomène s'appelle divergence pas-à-pas (stepwise divergence) (voir Figure 10).

Si on peut exprimer la forme générale de la relation de récurrence comme suit :

$$Y_{K+1} = Y_K + h_K \Phi (t_K, Y_K, Y_{K+1}, h_K) \quad ; \quad Y_o = n \quad K \geq 0$$

les méthodes seront dites "explicites" ; si Y_{K+1} apparaît explicitement dans les deux nombres de l'égalité ci-dessus alors la méthode sera dite "implicite". Toutes les méthodes utilisées par le système sont de type explicite.

On appel "méthodes à pas séparés", les méthodes qui utilisent la relation ci-dessus si Φ est calculé à partir de $F(Y_K, t_K)$ seulement. Si Φ requiert la connaissance de $F(Y_{K+1}, t_{K+1})$, les méthodes sont dites "à pas liés" et une formule de départ est nécessaire.

III . METHODES DE RUNGE-KUTTA

Les méthodes qui suivent sont des cas particuliers d'une méthode, à pas séparés, générale appelée de Runge-Kutta et qui essentiellement calcule Y_{t+1} comme une combinaison linéaire des valeurs de F. Il est démontré (4) que toute ces méthodes sont convergentes, faiblement et fortement stables.

On appelle rang ou ordre de la méthode, le nombre de fois qu'on évalue la fonction F afin de calculer Y_{K+1} à partir de Y_K et F.

III.1 Méthode d'Euler ou de la tangeante : (Ordre 1)

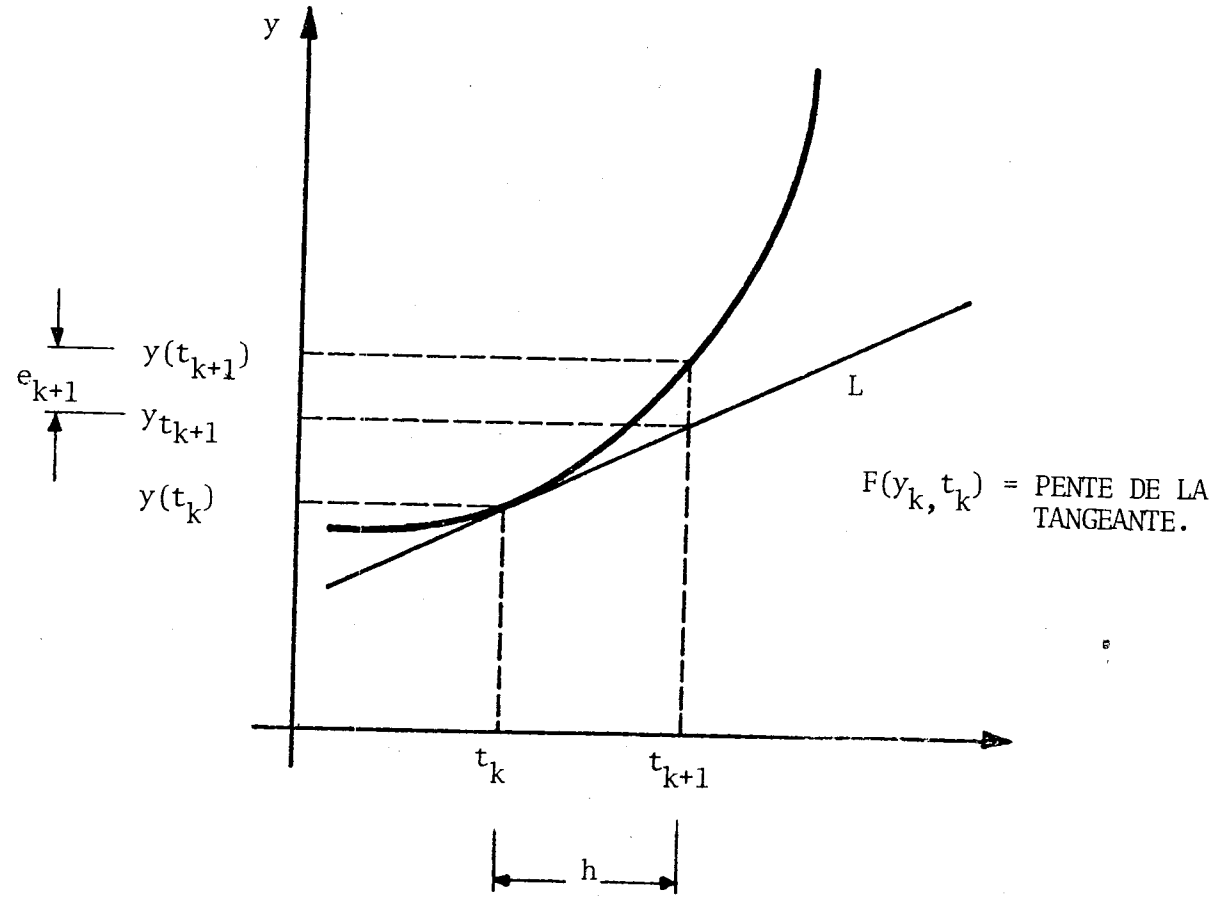
$$Y_{K+1} = Y_K + h \cdot F(Y_K, t_K)$$

cette approximation peut être trouvée facilement à partir de l'équation (A), voir figure (1).

$$F(Y_K, t_K) = \frac{dy}{dt} /_{t_K} \simeq \frac{Y_{K+1} - Y_K}{h}$$

où $h = t_{K+1} - t_K$.

FIG. 1



III.2 Méthode de la tangeante améliorée : (Ordre 2)

$$Y_{K+1/2} = Y_K + \frac{h}{2} F(Y_K, t_K)$$

$$Y_{K+1} = Y_K + h\, F(Y_{K+1/2}, t_{K+1/2})$$

cette formule est semblable à celle d'Euler, sauf que F n'est plus la pente de la tangeante au point t_K mais au point $t_{K+1/2}$.

III.3 Méthode d'Euler-Cauchy ou Trapezoidale : (Ordre 2)

$$Y_{K+1} = Y_K + h \,.\, F(Y_K, t_K)$$

$$Y_{K+1} = Y_K + h \left(\frac{F(Y_K, t_K) + F(Y_{K+1}, t_{K+1})}{2}\right)$$

cette méthode est semblable également à celle d'Euler, mais la pente de la tangeante est remplacée par la moyenne arithmétique des pentes aux points t_K et t_{K+1}.

III.4 Méthode de Heun : (Ordre 2)

$$Y_{K+2/3} = Y_K + \frac{2}{3} h \,.\, F(Y_K, t_K)$$

$$Y_{K+1} = Y_K + \frac{h}{4} (F(Y_K, t_K) + 3\, F(Y_{K+2/3}, t_{K+2/3})$$

cette méthode est dite optimale pour l'ordre 2.
Car elle minimise les composantes de la fonction principale d'erreur, voir (1).
On rappelle que la fonction principale d'erreur Φ pour une méthode d'ordre p est tel que

$$F(Y_K, t_K, h) = \Delta(Y_K, t_K, h) + h^p\, \Phi(Y_K, t_K) + G(h^{p+1})$$

Δ étant l'incrément exact tel que

$$Y(t_{K+1}) = Y(t_K) + h \,.\, \Delta(Y_K, t_K, h)$$

G étant une fonction qui dépend exclusivement de h^{p+1}.

III.5 Méthode de Simpson : (Ordre 3)

$$Y_{K+1/2} = Y_K + \frac{h}{2} F(Y_K, t_K).$$

$$Y'_K = Y_K - h\, F(Y_K, t_K) + 2h\, F(Y_{K+1/2}, t_{K+1/2})$$

$$Y_K = Y_K + \frac{h}{6} F(Y_K, t_K) + \frac{2}{3} h\, F(Y_{K+1/2}, t_{K+1/2}) + \frac{h}{6} F(Y'_K, t_{K+1})$$

III.6 Remarques :

On remarque que pour une méthode d'ordre p, on a besoin de p évaluation de la fonction F.
Ceci signifie p fois le calcul (simulation) du modèle tout entier.
Dès lors le temps de calcul augmente beaucoup, plus l'ordre de la méthode est élevé.
En général si on utilise une méthode d'ordre supérieur, on obtient une précision plus grande.

Les modèles traités étant peu précis, il semble inutile d'avoir une méthode plus précise que le modèle. D'autre part le gain de précision obtenu avec un ordre 4 est peu sensible, et la propagation d'erreurs de troncature (*) devient relativement plus importante.

(*) les erreurs de troncature ou erreurs d'arrondi sont dûes à la représentation (interne à l'ordinateur) des nombres réels à virgule flottante, et son traitement arithmétique par l'ordinateur.

IV . METHODES DE MILNE

Les méthodes de Milne ou de Prédiction Correction, sont des méthodes qui utilisent deux formules à pas liés de type différent. La première formule ou formule de prédiction doit être une formule explicite qui fournit une approximation $Y_{t_{n+1}}$ de $Y(t_{n+1})$.

La deuxième formule ou formule de correction est une formule implicite où l'on a besoin de la connaissance de $Y_{t_{n+1}}$.

Il est parfois recommandé de faire plusieurs itérations sur la formule de correction (par exemple jusqu'à ce que les deux valeurs soient égales à une certaine précision). Ce procédé n'améliore pas beaucoup la précision, c'est pourquoi dans la méthode que nous emploierons on ne fera qu'une seule itération.

De façon à s'assurer de la stabilité et de la convergence de la méthode ainsi que pour pouvoir faire facilement un calcul approximatif de l'erreur, on utilise comme formules de prédiction et correction, deux méthodes explicites de même ordre p, qu'on sait être convergentes.

Comme prédiction on utilise la formule de Heun et comme correction la formule de Cauchy.
On obtient alors une formule de Milne-Heun-Cauchy

$$Y_{K+2/3} = Y_K + \frac{2}{3} h\, F(Y_K, t_K)$$

Prédicteur $$Y'_{K+1} = Y_K + \frac{h}{4} [F(Y_K, t_K) + 3\, F(Y_{K+2/3}, t_{K+2/3})]$$

Heun 2e ordre

Correcteur $$Y_{K+1} = Y_K + \frac{h}{2} [F(Y_K, t_K) + F(Y'_{K+1}, t_{K+1})]$$

On a choisi l'ordre 2 pour ne pas augmenter le temps de calcul.
La méthode de Heun comme prédiction parait la plus intéressante, car au cours des essais sur plusieurs modèles, elle a donné l'erreur la plus petite pour l'ordre 2.

V . EVALUATION APPROXIMATIVE DE L'ERREUR DANS LA METHODE

Le choix entre plusieurs méthodes d'intégration n'est pas intéressante si on ne connait pas la précision obtenue par la méthode choisie, en la comparant à la précision demandée. Si l'on dispose de méthodes donnant une précision comparable, on choisira celle qui coûte le moins en temps de calcul.

La méthode d'évaluation de l'erreur que nous avons implémentée est décrite dans (1). Cette méthode est basée sur une comparaison de l'incrément approximatif obtenu avec un pas d'intégration h et deux pas d'intégration $\frac{h}{2}$. Soit δ_1 le premier incrément et δ_2 le second. On évalue alors la fonction principale d'erreur Φ comme

$$h^{p+1} \cdot \Phi(Y_K, t_K) = - \frac{1}{1 - 2^{-p}} (\delta_2 - \delta_1)$$

Or, les erreurs se propagent entre deux calculs d'intégration ; on peut utiliser comme équation de propagation de l'erreur au (K+1)ième pas, la relation

$$e_{K+1} = (1 + h \cdot \frac{dF(Y_K, t_K)}{dY_K}) \, e_K + h^{p+1} \, \Phi(Y_K, t_K)$$

avec $e_0 = 0$.

L'évaluation de l'erreur dûe à la méthode est correcte (1) si on connait $\frac{dF}{dY_K}$ avec précision. Mais en général pour les modèles traités, la fonction F est extrêmement complexe et non linéaire ; cela amène dans la pratique à faire une évaluation approximative d'une telle dérivée. Donc le calcul d'erreur est moins précis, mais reste néanmoins satisfaisant comme indication de l'ordre de grandeur.

L'évaluation de l'erreur et sa comparaison avec l'erreur réelle obtenue pour les différentes méthodes d'intégration retenues sont présentées sur un exemple académique dont on connait la solution exacte.

Exemple numérique

Soit l'équation différentielle $\frac{dy(t)}{dt} = - y . t$ avec la condition initiale $y(0)=0$
sa solution est $y(t) = e^{-t^2/2}$.

On a utilisé les différentes méthodes d'intégration, entre le temps t=0 et t=3.0 avec un pas constant dt=0.1.

Pour t=3.0 la solution exacte étant y(3.0)=0.011109

Méthode	Val y(3.0)	Erreur obtenue	Erreur calculée
Euler	0.0077912	0.0033178	0.00074124
Tangeante améliorée	0.0082464	0.0028626	0.000045073
Trapézoïdale	0.0099671	0.0011419	0.00055809
Heun	0.0090471	0.0020619	0.00027848
Milne (Pred.Co)	0.0096442	0.0014648	0.00032476
Simpson	0.0086403	0.0024687	0.000089320

Figure 2

Comparaison des résultats

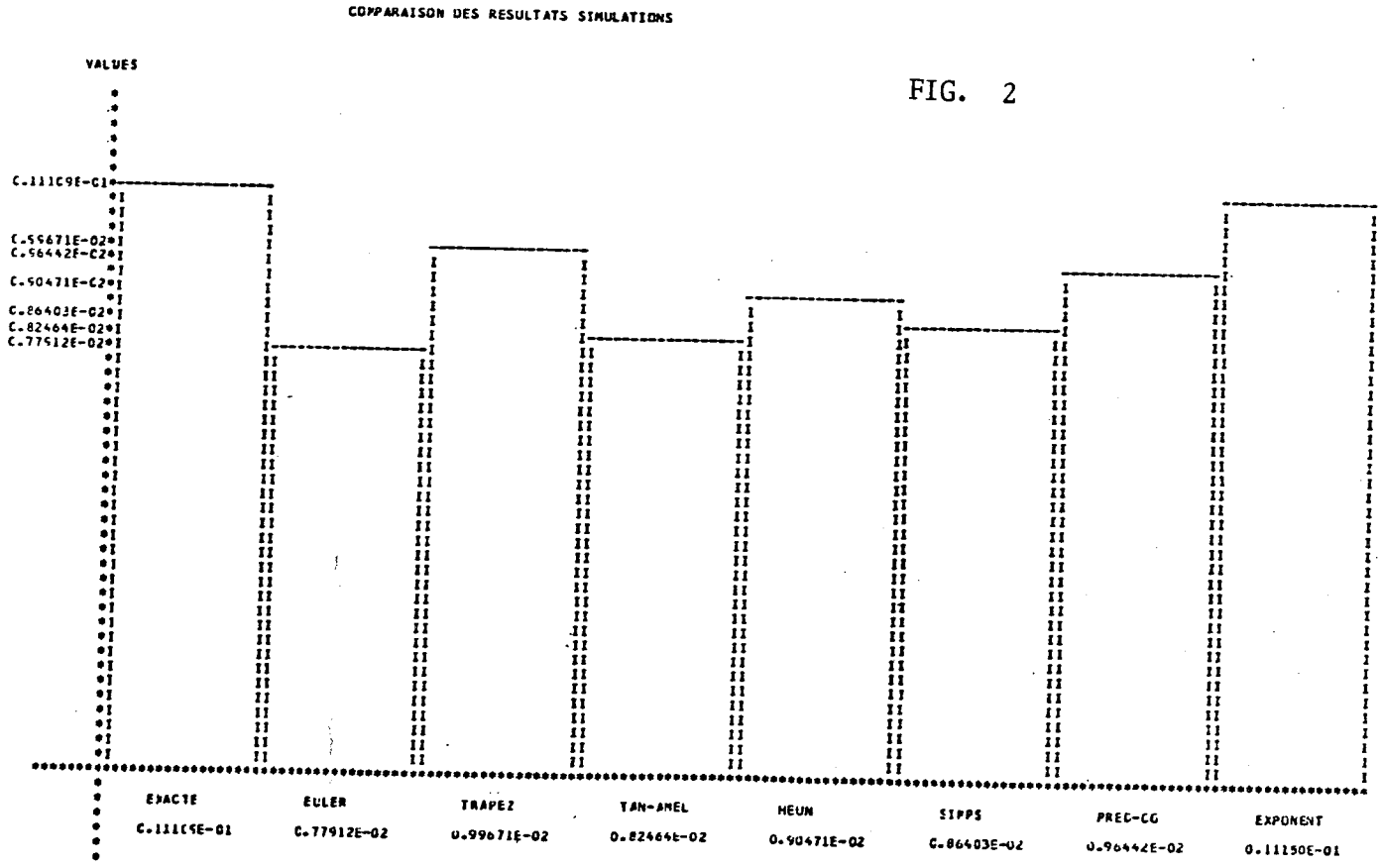


FIG. 2

Figure 3

Comparaison des erreurs obtenues en %

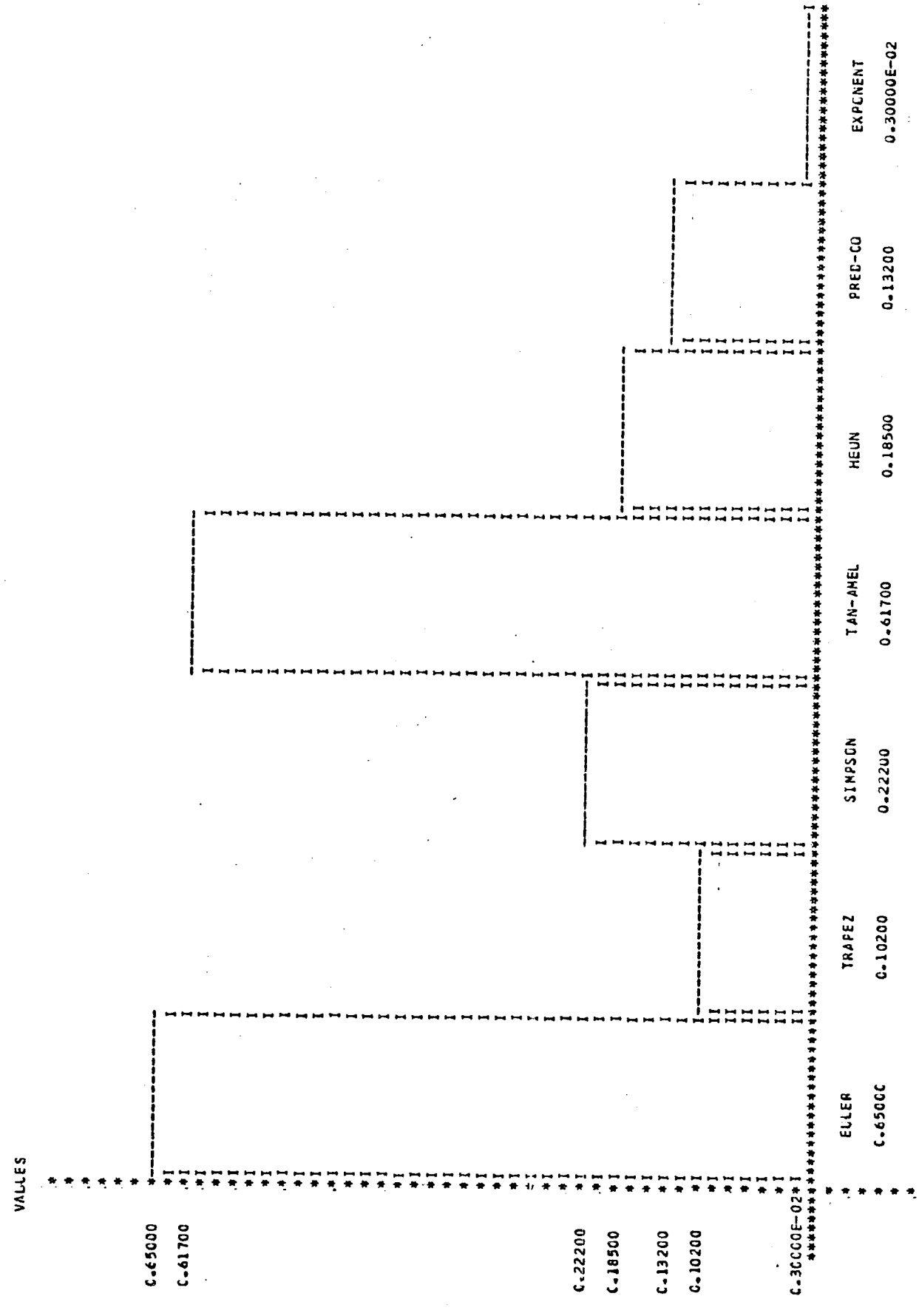


Figure 4

Comparaison erreur calculée / erreur obtenue.

Pour la même équation, on a obtenu les résultats suivants (au temps t=3.0) figures 5 et 9, pour chacune des méthodes en changeant seulement le pas d'intégration.

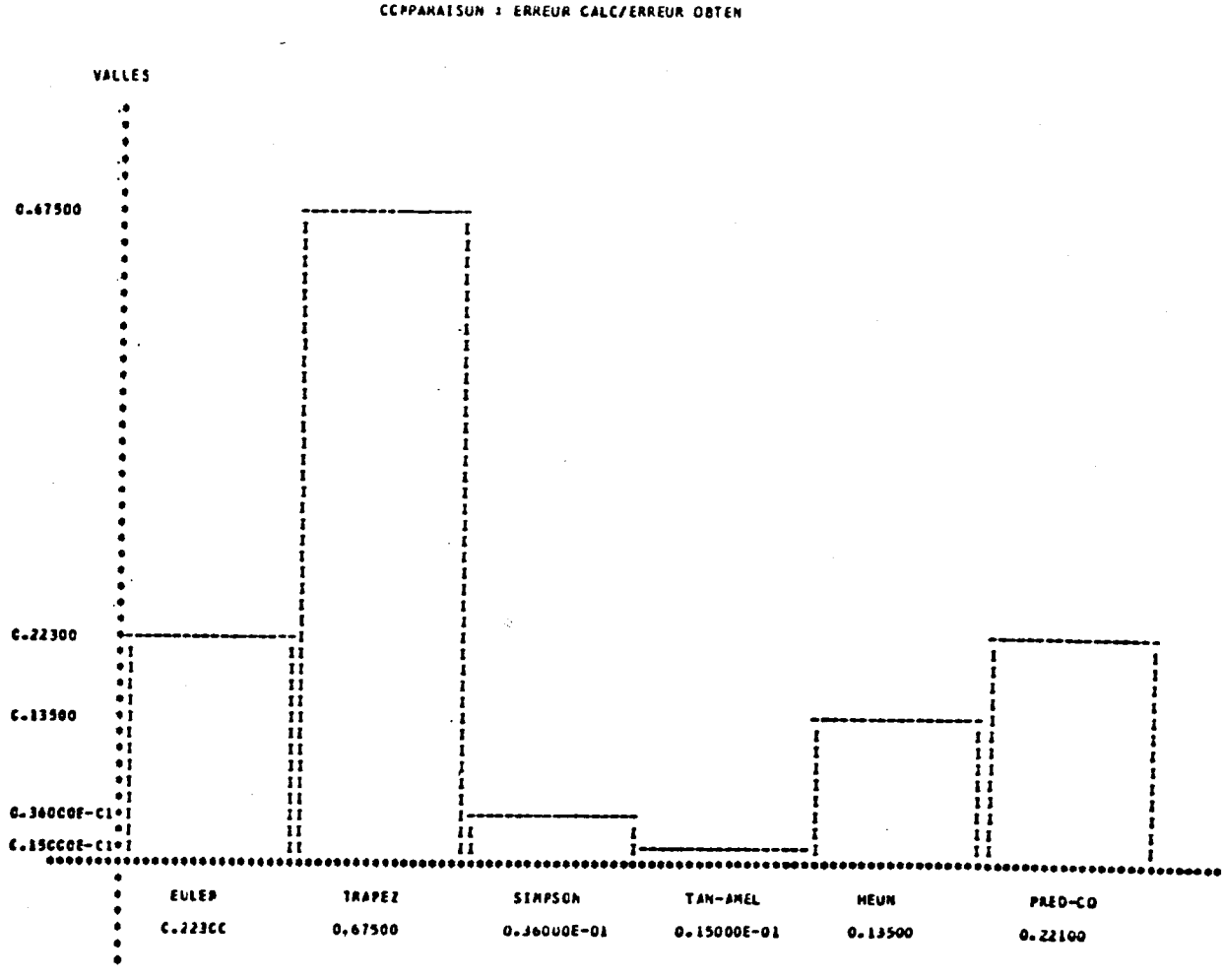


Figure 5

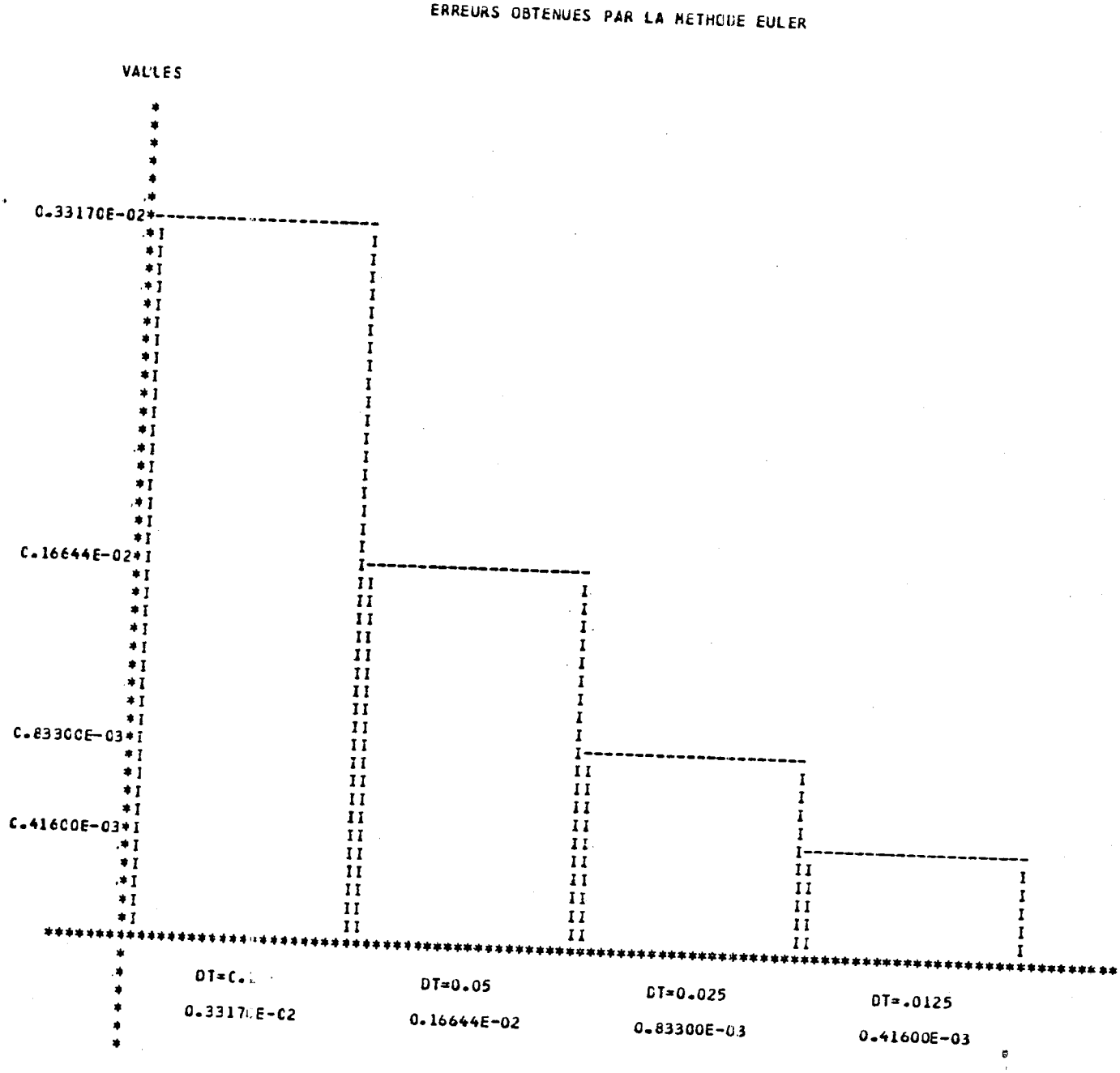

ERREURS OBTENUES PAR LA METHODE EULER
VALLES
0.33170E-02
0.16644E-02
0.83300E-03
0.41600E-03
DT=0.1
0.33170E-02
DT=0.05
0.16644E-02
DT=0.025
0.83300E-03
DT=.0125
0.41600E-03

Figure 6

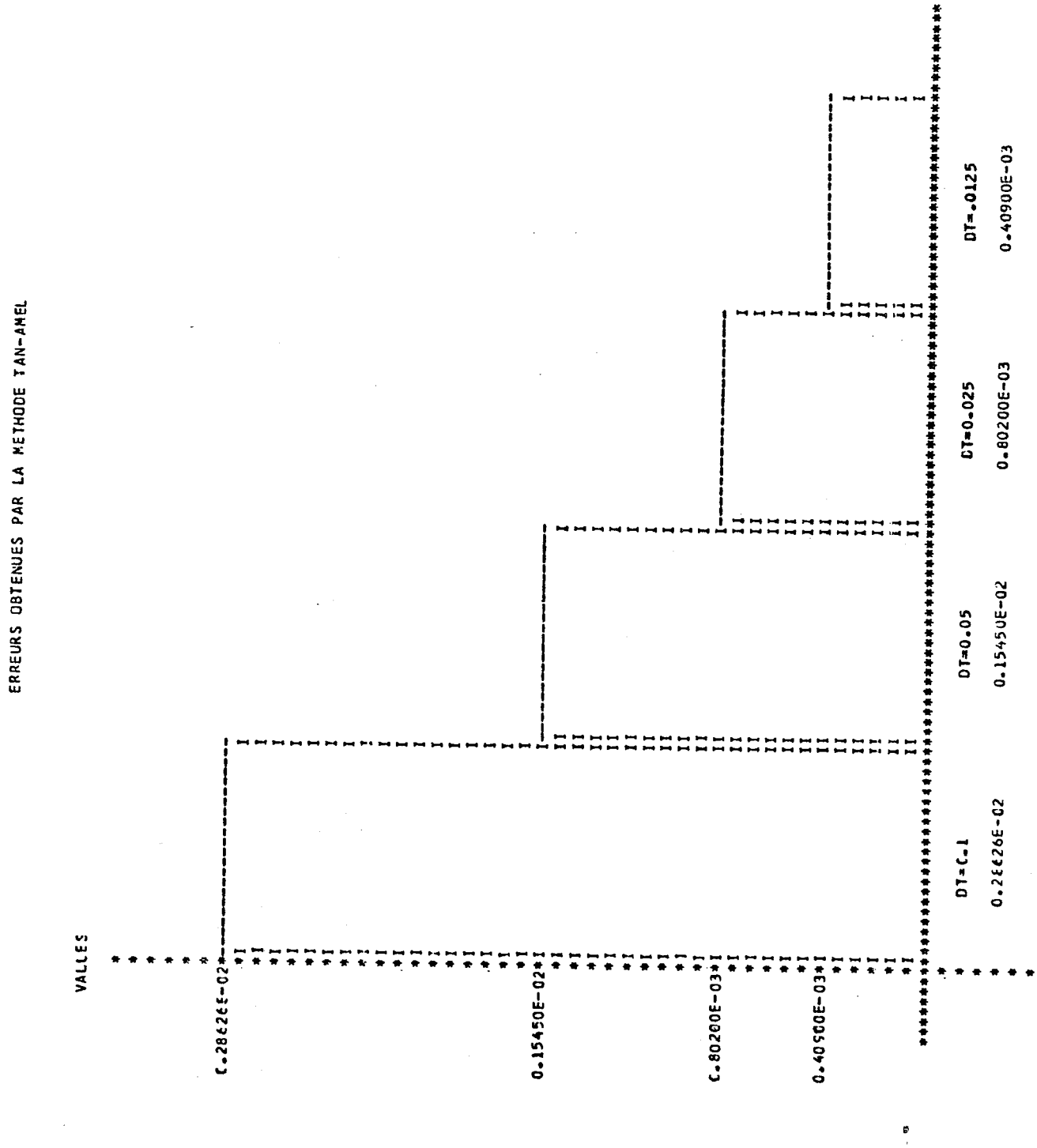

ERREURS OBTENUES PAR LA METHODE TAN-AMEL
VALLES
0.28626E-02
0.15450E-02
0.80200E-03
0.40500E-03
DT=0.1
DT=0.05
DT=0.025
DT=.0125
0.26626E-02
0.15450E-02
0.80200E-03
0.40900E-03

Figure 7

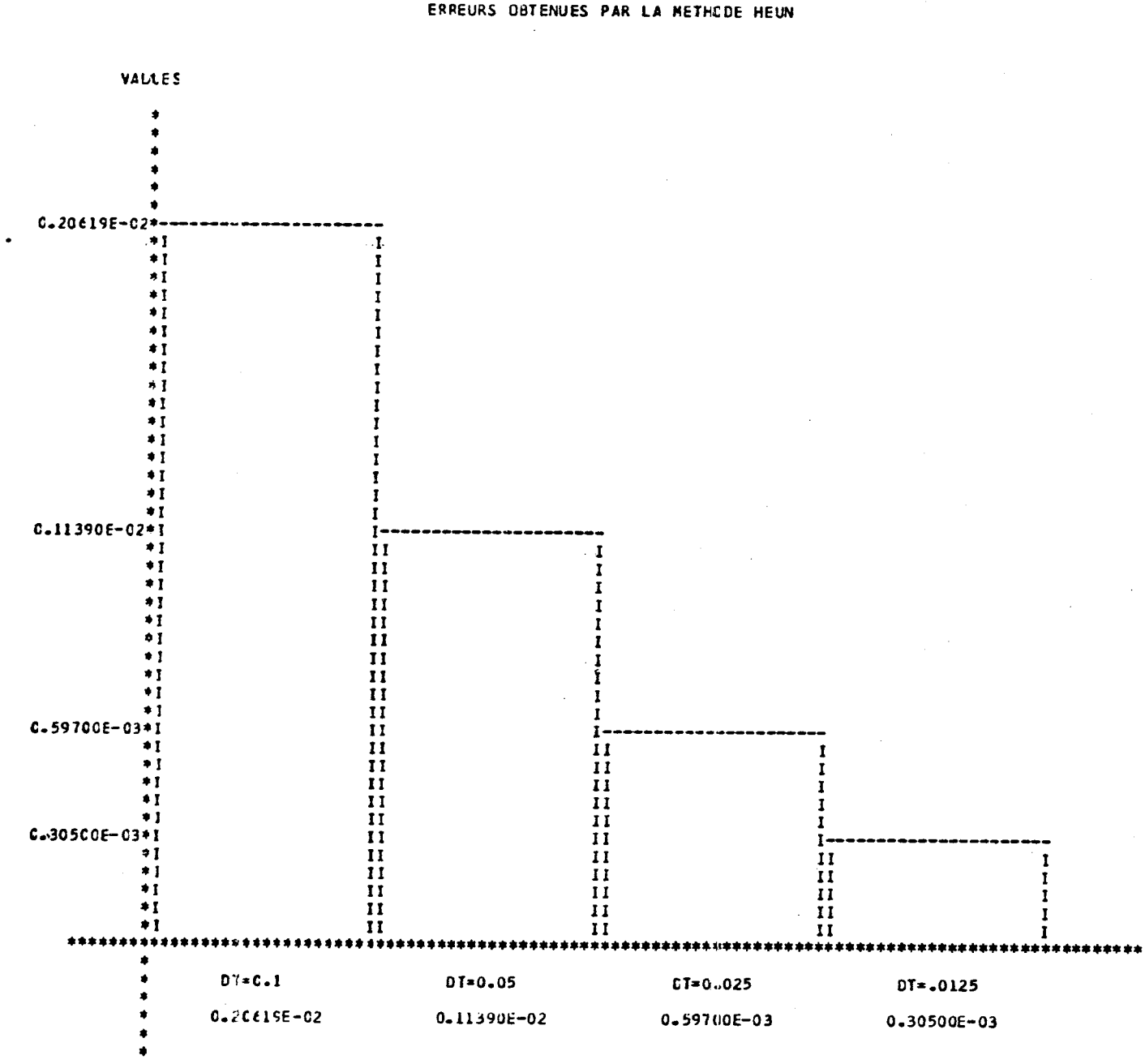

ERREURS OBTENUES PAR LA METHODE HEUN
VALEURS
0.20619E-02
0.11390E-02
0.59700E-03
0.30500E-03
DT=0.1
0.20619E-02
DT=0.05
0.11390E-02
DT=0.025
0.59700E-03
DT=.0125
0.30500E-03

Figure 8

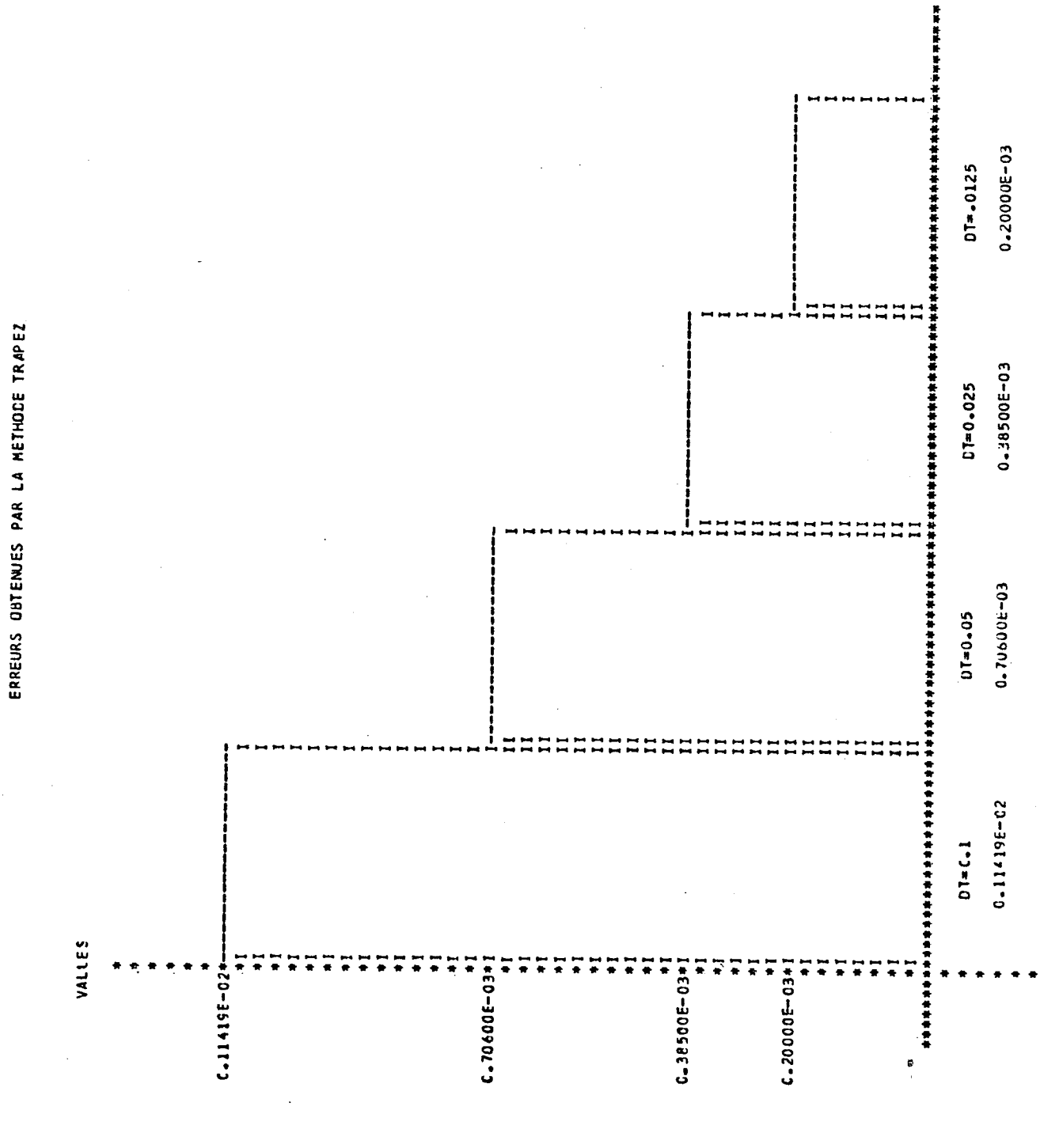
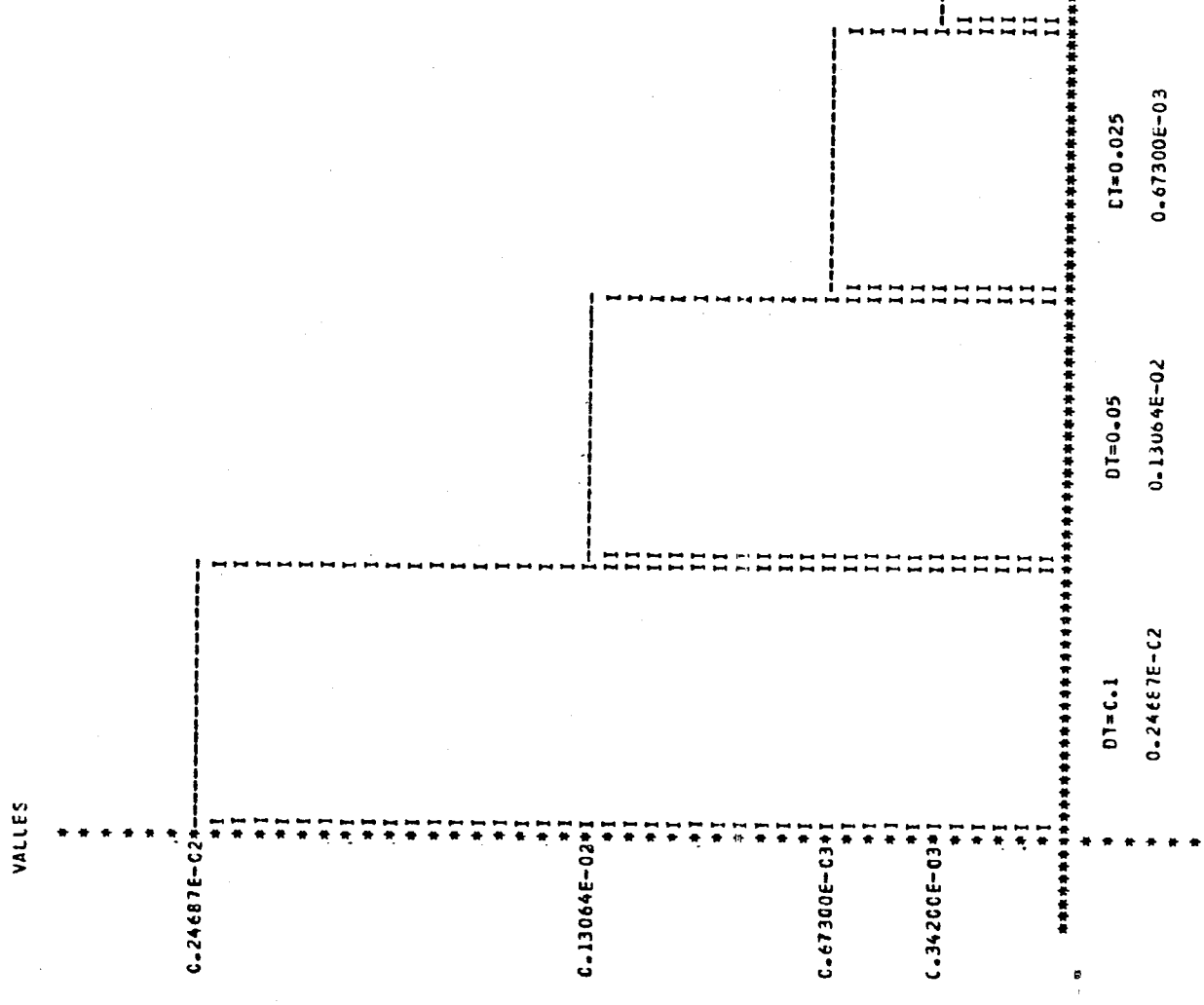

ERREURS OBTENUES PAR LA METHODE TRAPEZ
VALLES
0.11419E-02
0.70600E-03
0.38500E-03
0.20000E-03
DT=0.1
DT=0.05
DT=0.025
DT=.0125
0.11419E-02
0.70600E-03
0.38500E-03
0.20000E-03

Figure 9


ERREURS OBTENUES PAR LA METHODE SIMPSON
VALLES
0.24687E-02
0.13064E-02
0.67300E-03
0.34200E-03
DT=0.1
DT=0.05
DT=0.025
DT=.0125
0.24687E-02
0.13064E-02
0.67300E-03
0.34200E-03

VI . AJUSTEMENT DU PAS D'INTEGRATION PAR RAPPORT A L'ERREUR EVALUEE

Grâce au fait que toutes les méthodes sus-mentionnées sont convergentes, on peut ajuster le pas d'intégration h, jusqu'à ce qu'on obtienne la précision voulue. Il est clair que pour des raisons de sécurité, l'utilisateur doit indiquer le pas d'intégration minimal ; bien que la méthode soit convergente, l'existence d'erreurs dues non pas à la méthode d'intégration, mais par exemple à l'arrondi, peut faire que la précision demandée ne soit jamais obtenue. Il est clair que si cela arrive, l'utilisateur devra utiliser une méthode d'ordre plus élevé. On rappelle en outre que la méthode d'évaluation de l'erreur, indiquée dans la section 4, est une évaluation de l'erreur dûe exclusivement à la méthode d'intégration, mais que cette évaluation est toujours soumise aux erreurs d'arrondi.

La précision demandée par l'utilisateur est en pourcentage, car il peut ne pas connaître a priori l'ordre de grandeur des résultats de sa simulation. Si la précision n'est pas obtenue pour une variable d'état quelconque, on divise le pas d'intégration par deux, et on répète l'opération jusqu'à la précision demandée ou jusqu'à obtenir un pas inférieur au pas d'intégration minimal ; dans ce cas, la simulation s'arrête en indiquant à l'utilisateur la cause de l'arrêt.

VII . LA METHODE D'INTEGRATION EXPONENTIELLE

Cette méthode a déjà fait ses preuves dans le cadre d'une autre application, le programme de simulation de circuits IMAG2 (5). On suppose que $F(y_K, t_K)$ est de la forme $A_e^{\lambda t} + B$. En choisissant A.B et λ de façon à assurer la cohérence entre la fonction F et sa dérivée F' au point t_K. On obtient alors

$$y_{K+1} = y_K + h \cdot (B+A(e^{\lambda h}-1)/\lambda h)$$

L'utilisateur doit spécifier les valeurs minimale et maximale du pas d'intégration h, l'ajustement se faisant automatiquement en fonction de l'erreur $E = h/F(t+h,y_{K+1}) - (B+Ae^{\lambda h})$ / et de la stabilité, selon algorithme décrit en (6).

Cette méthode se montre beaucoup plus stable par rapport à la propagation de l'erreur que les méthodes de Runge-Kutta ; c'est-à-dire pour le même pas d'intégration les résultats de l'intégration sont plus stables. De façon à visualiser cet effet, on prend la fonction sinus à l'entrée et on applique un délai matériel de premier ordre mal conditionné (magnitude du délai très grand par rapport au pas d'intégration). L'équation différentielle du délai matériel de premier ordre est

$$\frac{dOUT(t)}{dt} = \frac{1}{D(t)}(IN(t) - OUT(t) \cdot (1 + \frac{dD(t)}{dt})$$

avec la condition initiale OUT(O) voir à ce propos (7).

Dans la figure 10 on compare les résultats obtenus pour cet exemple avec les méthodes d'Euler et Exponentielle.

FIG. 10.1 DT = 1

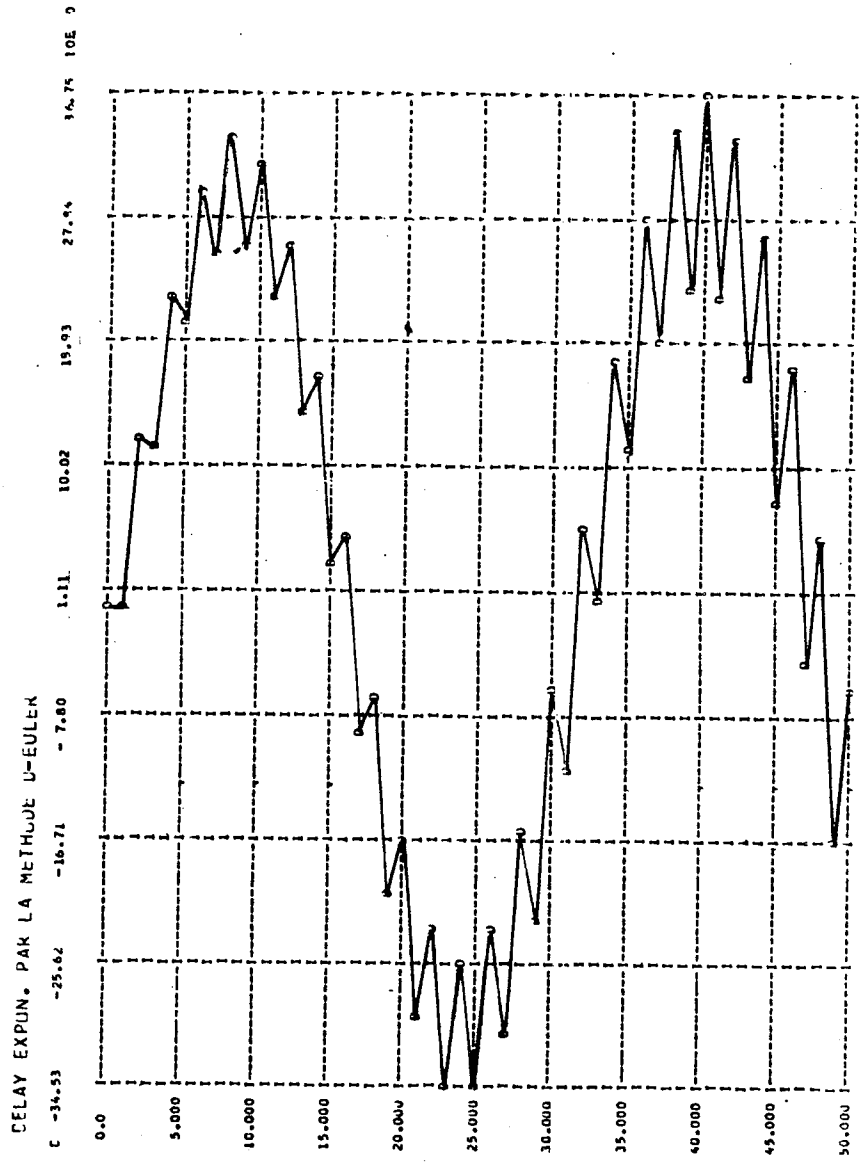
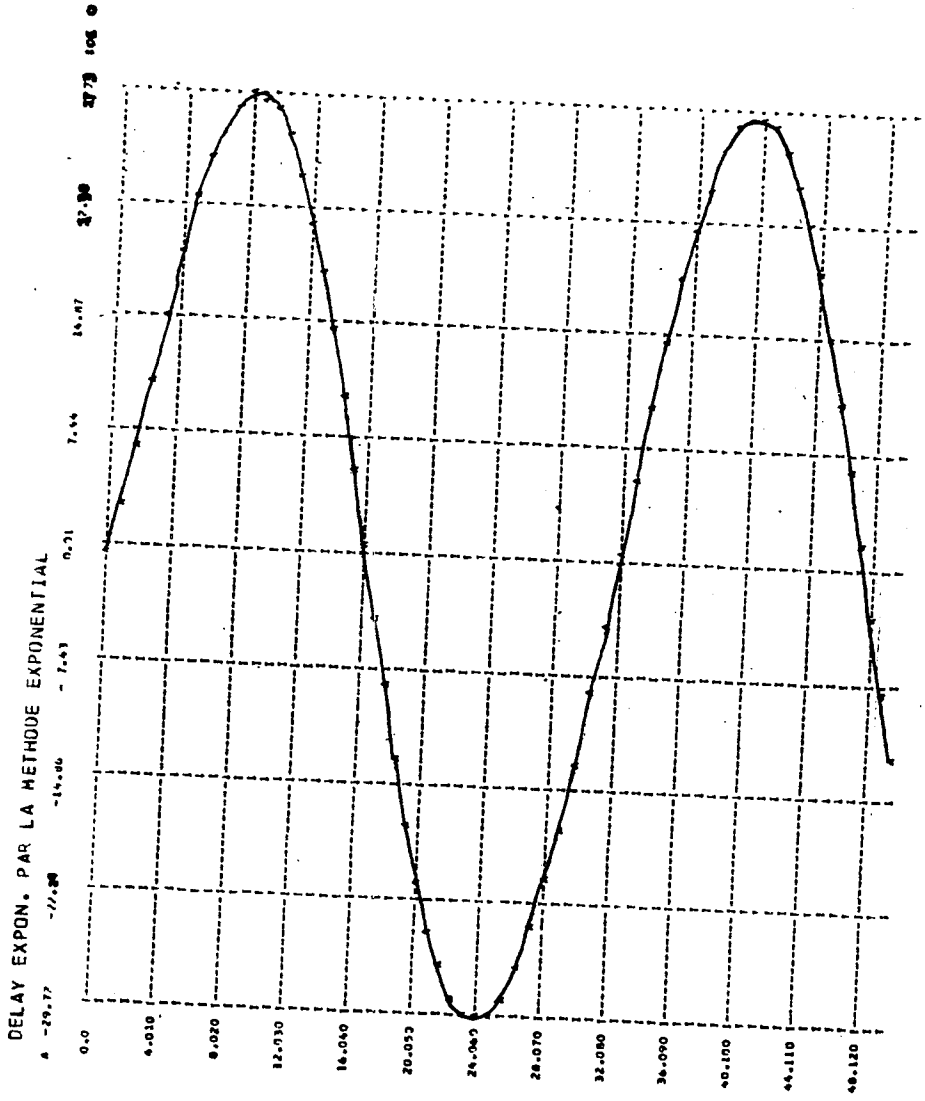


FIG. 10.2 $\overline{DT} = 1$



Il existe néanmoins des méthodes plus stables, même avec un rayon de stabilité infini, comme la méthode trapézoïdale (implicite), mais son coût d'exploitation est beaucoup plus élevé.

La méthode exponentielle étant une méthode d'ordre 2, donne une bonne précision, semblable à celle des méthodes Runge-Kutta d'ordre 2 ou 3 ; pourtant son coût à la simulation est légèrement supérieur à ces méthodes, car il faut donner un pas d'intégration maximal suffisamment petit pour faire une exploitation correcte des résultats, sans perdre pour cela la précision obtenue à cause de l'interpolation. Par contre, la méthode exponentielle étant une méthode à pas variable, l'exploitation des résultats est plus lente que dans les autres méthodes.

Ce n'est pas étonnant que pour des fonctions type exponentiel, les résultats soient plus précis que pour les autres méthodes ; ainsi pour l'exemple du paragraphe 5, on obtient comme valeur de solution au point t=3.0, y=0.011151, ce qui fait une erreur réelle de 0.000042.

Cette méthode paraît très utile pour traiter des systèmes ayant des constantes de temps peu différentes, tout en respectant la précision obtenue par les autres méthodes implémentées.

VIII . CONCLUSION

Nous avons constaté que l'utilisation d'une seule méthode (comme celle d'Euler par DYNAMO) est insuffisante et nous avons essayé d'y remédier.

Toutes les méthodes d'intégration existentes n'ont pas été présentées, ni implémentées. Les modèles dynamiques développés en Sciences Sociales en général n'exigent pas une grande précision, car les valeurs même sont connues de façon très approximative. Ceci nous a amené à ne pas considérer en particulier les méthodes implicites (en général plus précises que les méthodes explicites, mais nettement plus grands consommateurs de temps machine). Enfin c'est à l'utilisateur de dire le dernier mot à ce sujet.

A l'heure actuelle en modélisation socio-économique on traite des modèles à un seul niveau d'agrégation. Nous sommes persuadés que grâce à une conception modulaire descendante (8) on pourrait avoir bientôt des modèles avec plusieurs niveaux d'agrégation et donc avec des constantes de temps d'ordre différents. Mais ceci n'implique pas nécessairement que l'on traite des systèmes "stiff".

Les systèmes "stiff" se caractérisent par l'existence de constantes de temps très différentes ; par exemple, pour des variables de taux qui ont des échelles de temps très différents. On est obligé d'utiliser a priori avec les méthodes décrites, un pas d'intégration plus petit que la plus petite des constantes de temps, ceci peut nous amener non pas seulement à des simulations très longues mais à des erreurs assez grandes pour certaines variables (11). Les constantes de temps d'un système peuvent se déterminer formellement comme la réciproque des parties réelles des valeurs propres non nulles du Jacobien J. ($J = \frac{\partial F}{\partial y}$).

Le problème de "stiffness" commence à se présenter lorsque la différence entre les constantes de temps sont au moins de l'ordre de 10^3. Dans le cadre de la modélisation proposée, cela implique par exemple des relations entre variables sensibles dans un intervalle du mois et des autres sensibles seulement dans un intervalle d'un siècle !

Il existe bien sûr des méthodes de calcul implicites pour résoudre des systèmes d'équations "stiff", ces méthodes sont toujours très lourdes. Les travaux en particulier de Gear (9), (10) ont été déjà implémentés dans des programmes de simulation pour des systèmes mécanico-électriques où le problème de "stiffness" apparaît très souvent (11).

Il paraît donc plus raisonnable actuellement d'envisager l'ajustement indépendamment du pas d'intégration pour les équations différentielles non-couplées. Cela peut s'avérer intéressant pour des simulations très longues.

L'introduction de la modélisation continue dans l'espace (12) mis à part des problèmes méthodologiques, peut amener à la considération des systèmes aux équations à dérivés partiels (13) ; dans ce cas on devra faire appel à des méthodes d'intégration différentes de celles développées dans cette annexe.

REFERENCES

(1) P. POUZET

"Analyse Numérique", tome II
Laboratoire de Calcul, Faculté des Sciences, Université de Lille, 1972

(2) J. KUNTZMANN

"Méthodes Numériques"
Ed. Hermann, Paris 1969

(3) B.C. PATTEN

"A primer for ecological modeling and simulation with analog and digital computers"
(pp 4-12)
in «Systems Analysis and Simulation in Ecology»
B.C. PATTEN, Ed. Vol. I - Academic Press, New-York, 1971

(4) F.B. HILDEBRAND

"Finite-Difference Equations and Simulations"
Prentice Hall, 1968

(5) JACOLIN, LE FAOU, PIMORT, VERAN

"IMAG II, Description du Programme"
Documentation interne - IMAG, 1970

(6) FOWTER M., WARTEN R.

"Numerical Integration technique for ordinary differential equations with widely separated aigenvalues"
Sept 1967, pp 537-543 - IBM Journal

(7) RIVERA E.

"Review of Delays in Dynamic System Simulation Models"
Proocedings Informatica 76
pp 5-104, Bled, Yougoslavie, octobre 1976

(8) RECHENMANN F.

"Analyse et modélisation descendantes des systèmes socio-économiques"
Thèse Docteur-Ingénieur, ENSIMAG, Grenoble, 1976

(9) GEAR C.W.

"The Automatic Integration of Ordinary Differential Equations"
Communications of the ACM, vol. 14, 1971, pp 176-179

(10) GEAR C.W.

"The Numerical Integration of Ordinary Differential Equations"
Mathematics of Computation, Vol. 21, Avril 1967 pp 146-156

(11) WATSON D., GOURLAY A.R.

"Implicit Integration for CSMP III and the problem of stiffness"
Simulation, February 1976, pp 57-61

(12) RECHENMANN F., RIVERA E., UVIETTA P.

"Introduction de la Notion d'Espace dans les Modèles de Simulation"
Colloque sur les méthodes mathématiques appliquées à la géographie. Besançon, 2-3 octobre 1975

(13) NOURELDIN H.A.

"Simulation of the Initial-Value Problem in Partial Differential Equations",
Simulation 75, Zurich, june 1975.

- ANNEXE V -

UN EXEMPLE DE DOCUMENTATION

DOCUMENTATION

VARIAB	TYPE	DEFINITION	UNITS
ACPRIX	RATE	ACCELERATEUR D'EVOLUTION DU PRIX	FRANCS
ASE	AUXILIAR	ATTRACTIVITE LIEE AU SOUS-EMPLOI	SANS DIM
ASL	AUXILIAR	ATTRACTIVITE LIEE AU SOUS-LOGEMENT	SANS DIM
ATTR	AUXILIAR	ATTRACTIVITE	SANS DIM
ATTRN	CONSTANT	ATTRACTIVITE NORMALE	SANS DIM
ATTRP	AUXILIAR	ATTRACTIVITE PERCUE	SANS DIM
DL	AUXILIAR	DEMANDE DE LOGEMENTS	SANS DIM
DPT	AUXILIAR	NOMBRE DE DEPARTS	HABIT/AN
DVMI	CONSTANT	DUREE MOYENNE D'UN EMPLOI DANS L'INDUSTRIE	ANS
DVML	CONSTANT	DUREE DE VIE MOYENNE DES LOGEMENTS	ANS
EMP	AUXILIAR	NOMBRE TOTAL D'EMPLOIS EXISTANTS	EMPLOIS
EMPS	AUXILIAR	NOMBRE D'EMPLOIS DANS LES SERVICES	EMPLOIS
EMPUI	CONSTANT	UNITE D'EMPLOI DANS L'INDUSTRIE	SANS DIM
FACP	AUXILIAR	FACTEUR D'ACCELERATION DU PRIX	SANS DIM
FCLRN	AUXILIAR	FACTEUR DE CONSTRUCTION LIE AU SOUS-LOGEMENT	SANS DIM
FCP	AUXILIAR	FACTEUR DE PRIX LIE AUXILIAR LA SPECULATION	SANS DIM
FPRIX	AUXILIAR	FACTEUR DE PRIX LIMITANT LA CREATION D'EMPLO	SANS DIM
FSE	AUXILIAR	FACTEUR DE SOUS-EMPLOI	SANS DIM
FSL	AUXILIAR	FACTEUR DE SOUS-LOGEMENT	SANS DIM
IC	RATE	FLUX D'EMPLOIS CREES DANS L'INDUSTRIE	EMPLOIS/AN
ID	RATE	FLUX D'EMPLOIS DISPARUS DANS L'INDUSTRIE	EMPLOIS/AN
IMIG	AUXILIAR	NOMBRE D'IMMIGRANTS	HABIT.
INC	LEVEL	EMPLOIS DANS L'INDUSTRIE	EMPLOIS
LC	RATE	FLUX DE LOGEMENTS CONSTRUITS	LOGEM./AN
LCN	CONSTANT	TAUX NORMAL DE CONSTRUCTION DE LOGEMENTS	SANS DIM
LD	RATE	FLUX DE LOGEMENTS DETRUITS	LOGEM./AN
LOG	LEVEL	NOMBRE DE LOGEMENTS EXISTANTS	LOGEM.
MN	RATE	FLUX DU MOUVEMENT NATUREL	HABIT./AN
NMSE	AUXILIAR	FACTEUR LIE AU SOUS-EMPLOI PERCU	SANS DIM
NMSLT	CONSTANT	FACTEUR NORMAL LIE AU SOUS-LOGEMENT	SANS DIM
PAC	AUXILIAR	POPULATION ACTIVE TOTALE	HABITANTS
POP	LEVEL	POPULATION	HABITANTS
PRIX	LEVEL	PRIX DES TERRAINS	FRANCS
PRIXI	INITIAL	PRIX INITIAL DES TERRAINS	FRANCS
RPX	AUXILIAR	RAPPORT DE PRIX	SANS DIM
RSE	AUXILIAR	RATIO DU SOUS-EMPLOI TOTAL	EMPLOIS/HAB
RSEP	AUXILIAR	RATIO PERCU DE SOUS-EMPLOI TOTAL	SANS DIM
RSL	AUXILIAR	RATIO DE SOUS-LOGEMENT	SANS DIM
SDN	AUXILIAR	NOMBRE DESIRE D'EMPLOIS DANS LES SERVICES	EMPLOIS
SDNPP	CONSTANT	TAUX D'EMPLOI DANS LES SERVICES PAR PERSONNE	EMPLOIS/HAB
SM	RATE	FLUX DE SOLDE MIGRATOIRE	HABIT./AN
SUI	CONSTANT	SURFACE UNITAIRE PAR EMPLOI INDUSTRIEL	M**2/EMPLOI
SUL	CONSTANT	SURFACE UNITAIRE PAR LOGEMENT	M**2/LOGEM.
SURF	AUXILIAR	SURFACE OCCUPEE TOTALE	M**2
SURFI	AUXILIAR	SURFACE OCCUPEE PAR LES INDUSTRIES	M**2
SURFL	AUXILIAR	SURFACE OCCUPEE PAR LES LOGEMENTS	M**2
TACP	CONSTANT	TAUX DE POPULATION ACTIVE	SANS DIM
TACPA	CONSTANT	TAUX NORMAL D'ACCELERATION DU PRIX	SANS DIM
TEMN	CONSTANT	TAUX NORMAL D'EMIGRATION	HABIT/AN
TCN	CONSTANT	TAUX NORMAL DE CREATION D'EMPLOIS DANS L'IND	SANS DIM
TIMN	CONSTANT	TAUX NORMAL D'IMMIGRATION	HABIT/AN
TMN	CONSTANT	TAUX MOYEN DE MOUVEMENT NATUREL	HABIT/AN
ULPP	CONSTANT	UNITE DE LOGEMENT PAR PERSONNE	SANS DIM
VEXF	AUXILIAR	VITESSE D'EXPANSION DE LA SURFACE TOTALE	SANS DIM

PAGE 001

```
C
C  MODELE REGIONAL (5 AGGLOMERATIONS), ETUDE MIGRATOIRE
C
C ***PARTIE INITIALE***
C
C **********************************
      SUBROUTINE INIT(Y,DY,TIME,M)
C **********************************
C PASSAGE DES PARAMETRES NUMERIQUES
      COMMON /AUTRES/ FPRIXT,FSLT,NMSLT,ASET,ASLT,FCLRLT,FACPT,FCPT
     1,EMPJI,SDNPP,TMN,ULPP,LCN,DVML,TACPN,SUL,SUI,TCN,DVMI,TACP,TENN
     2,TIMN,ATTRN,ATENV,TENMB,TEMNB,DEL,PRIXI
      REAL FPRIXT(13),FSLT(9),NMSLT(9),ASET(9),ASLT(9),FCLRLT(9),FACPT(9
     *),FCPT(5)
      REAL LCN,VIS(11)
      REAL A(9),DA(9),B(9),DB(9),C(9),DC(9),D(9),DD(9),E(9),DE(9)
      REAL MN1,MN2,MN3,MN4,MN5
      REAL ENOM(11),VIS(11)
      DIMENSION Y(1),DY(1)
C CORRESPONDANCE NOMS VARIABLES A SAUVEGARDER
      REAL*8 NOMVAR(11)
      DATA NOMVAR/'POP1    ','POP2    ','POP3    ','POP4    ','POP5    '
     *,'ATTRP1  ','ATTRP2  ','ATTRP3  ','ATTRP4  ','ATTRP5  ','SM2     '
     */
C SAUVEGARDE  DES NOMS
      CALL NMSTCK(11,NOMVAR)
      CONTINUE
C
C ***PARTIE DYNAMIQUE***
C
C **************************
      ENTRY DERIV(Y,DY,TIME,M)
C **************************
C
C DENOMINATION INTERNE AU NIVEAU GLOBAL DES VARIABLES D'ETAT
C
      DO 1 I=1,9
C VILLE 1
      A(I)=Y(I)
      DA(I)=DY(I)
C VILLE 2
      B(I)=Y(9+I)
      DB(I)=DY(9+I)
C VILLE 3
      C(I)=Y(18+I)
      DC(I)=DY(18+I)
C VILLE 4
      D(I)=Y(27+I)
      DD(I)=DY(27+I)
C VILLE 5
      E(I)=Y(36+I)
      DE(I)=DY(36+I)
 1    CONTINUE
C
C CORRESPONDANCE
```

PAGE 002

```
C
      POP1=Y(9)
      POP2=Y(18)
      POP3=Y(27)
      POP4=Y(36)
      POP5=Y(45)
C
C NIVEAU HIERARCHIQUE  AGGLOMERATION
C
C
C DEBUT DE LA DESCRIPTION
C
C SIMULATION VILLES SECTEURS (EMPLOI-LOGEMENT)
C       SIMULATION VILLE 1
            CALL VILLE(A,DA,ATTRP1,DPT1,MN1,VIS)
C
C       SIMULATION VILLE 2
            CALL VILLE(B,DB,ATTRP2,DPT2,MN2,VIS)
C
C       SIMULATION VILLE 3
            CALL VILLE(C,DC,ATTRP3,DPT3,MN3,VIS)
C
C       SIMULATION VILLE 4
            CALL VILLE(D,DD,ATTRP4,DPT4,MN4,VIS)
C
C       SIMULATION VILLE 5
            CALL VILLE(E,DE,ATTRP5,DPT5,MN5,VIS)
C
C NIVEAU HIERARCHIQUE INTER-URBAIN
C
C EQUATIONS DE RELATIONS INTER-URBAINS
C
C CALCUL DE PONDERATIONS DES ATTRACTIVITES RELATIVES
      ATX=ATTRP1+ATTRP3+ATTRP4
      ATY=ATTRP3+ATTRP2
      ATZ=ATTRP4+ATTRP3+ATTRP2+ATTRP5
      ATA=ATTRP1+ATTRP3+ATTRP5+ATTRP4+ATENV
      ATB=ATTRP3+ATTRP1+ATTRP4+ATTRP5+ATTRP2+ATENV
C
C **********************************
C
C MODIFICATIONS INCORPORES PAR LE SCENARIO :
C  'SATELLISATION DE LA VILLE RELAI'
C
C       *  *  *  *  *  *  *
C  CS=COEFFICIENT DE SATURATION DE L'ATTRACTIVITE
C
      CS4=(1./1.E5)*(5.E5-POP4)
      CS3=(1./1.E5)*(1.E6-POP3)
C  LE COEFFICIENT CS COMMENCE A JOUER 1.E5 HABITANTS AVANT LA SATURATION
      ATTRP4=CLIP(ATTRP4,ATTRP4*CS4,4.E5,POP4)
      ATTRP3=CLIP(ATTRP3,ATTRP3*CS3,9.E5,POP3)
C  L'EXCES D'ATTRACTIVITE DE LA METROPOLE 3 SE
C  RAPPORTE SUR UN FLUX MIGRATOIRE VERS LA VILLE 2
C  QUI DEVIENT SATELLITE DE LA METROPOLE 3
```

```
      FAY=DPT3*(ATTRP2/ATA)
C *********************************
C
C CALCUL DES FLUX MIGRATOIRES RELATIVES
C
      FXB=DPT1*(ATTRP4/ATX)
      FXA=DPT1*(ATTRP3/ATX)
      FAX=DPT3*(ATTRP1/ATA)
      FBZ=DPT4*(ATTRP5/ATB)
      FZB=DPT5*(ATTRP4/ATZ)
      FZY=DPT5*(ATTRP2/ATZ)
      FZA=DPT5*(ATTRP3/ATZ)
      FYA=DPT2*(ATTRP3/ATY)
      FAB=DPT3*(ATTRP4/ATA)
      FBA=DPT4*(ATTRP3/ATB)
      FBE=DPT4*(ATENV/ATB)
      FEB=POP4*TEMNB*ATTRP4
      FEA=POP3*TENMB*ATTRP3
      FAE=DPT3*(ATENV/ATA)
C SOLDES MIGRATOIRES
      SM1=FAX-(FXA+FXB)
C *********************************
C CHANGEMENT DU AU SCENARIO
      SM2=CLIP(FZY+FYA,FZY-FYA+FAY,9.E5,POP3)
C *********************************
      SM3=FBA+FXA+FZA+FYA+FEA-(FAX+FAB+FAE)
      SM4=FAB+FXB+FZB+FEB-(FBA+FBZ+FBE)
      SM5=FBZ-(FZB+FZY+FZA)
      SM2=CLIP(SM2,SM2+FAY,9.E5,POP3)
C
C NIVEAU HIERARCHIQUE URBAIN
C
C SIMULATION VILLES SECTEUR DEMOGRAPHIQUE
C
      CALL POP(SM1,MN1,DPOP1)
C
      CALL POP(SM2,MN2,DPOP2)
C
      CALL POP(SM3,MN3,DPOP3)
C
      CALL POP(SM4,MN4,DPOP4)
C
      CALL POP(SM5,MN5,DPOP5)
C
      DO 7 I=1,9
      DY(I)=DPOP1
      DY(I+9)=DPOP2
      DY(I+18)=DPOP3
      DY(I+27)=DPOP4
  7   DY(I+36)=DPOP5
C
C CORRESPONDANCE VARIABLES A SAUVEGARDER
      CALL CORRESP(FNOM,11,POP1,POP2,POP3,POP4,POP5,ATTRP1,
     *ATTRP2,ATTRP3,ATTRP4,ATTRP5,SM2)
C SAUVEGARDE DES VARIABLES DANS LE MEME ORDRE QUE NMSTCK
```

PAGE 004

```
      CALL AUXST(TIME,M,11,FNOM)
      RETURN
      END
C
C ***************************************
C  MODULES DE LA DESCRIPTION
C ***************************************
      SUBROUTINE VILLE(Y,DY,ATTRP,DPT,MN,VIS)
C
C SOUS-REGION VILLE,DYNAMIQUE EMPLOI-LOGEMENT-PRIX TERRAINS
C       ***    ***                ***      ***   ***
C
      COMMON /AUTRES/ FPRIXT,FSLT,NMSLT,ASET,ASLT,FCLRLT,FACPT,FCPT
     1,EMPUI,SDNPP,TMN,ULPP,LCN,DVML,TACPN,SUL,SUI,TCN,DVMI,TACP,TEMN
     2,TIMN,ATTRN,ATENV,TENMB,TEMNB,DEL,PRIXI
      REAL FPRIXT(13),FSLT(9),NMSLT(9),ASET(9),ASLT(9),FCLRLT(9),FACPT(9
     *),FCPT(5)
      REAL Y(1),DY(1),IND,LOG,MN,ID,LD,LC,NMSE,IC,LCN
      REAL VIS(1)
C VARIABLES D'ETAT INTERNES (DELAIS)
      Y1=Y(1)
      Y2=Y(2)
      Y3=Y(3)
C AUTRES VARIABLES D'ETAT
      ATTRP=Y(4)
      RSEP=Y(5)
      IND=Y(6)
      PRIX=Y(7)
      LOG=Y(8)
      POP=Y(9)
C
C  DELAIS
      EMPS=DELAY3(Y1,Y2,Y3,EN2,EN3,DEL)
C
C CORPS DU MODULE
      PAC=POP*TACP
      SDN=POP*SDNPP
      DL=POP*ULPP
      MN=POP*TMN
      LD=LOG/DVML
      RSL=LOG/DL
      ASL=TABHL(ASLT,RSL,0.,2.,.25)
      FCLRL=TABHL(FCLRLT,RSL,0.,2.,.25,.9)
      ID=IND/DVMI
      EMP=EMPS+IND*EMPUI
      RSE=EMP/PAC
      ASE=TABHL(ASET,RSE,0.,2.,.25,.9)
      LC=LCN*FCLRL*POP
      VSURFL=(LC-LD)*SUL
      SURFL=LOG*SUL
      SURFI=IND*SUI
      SURF=SURFL+SURFI
      FPRIX=TABHL(FPRIXT,PRIX,10.,70.,5.)
      NMSE=TABHL(NMSLT,RSEP,0.,2.,.25)
      FSE=TABHL(FSLT,NMSE,0.,2.,.25)
```

PAGE 005

```
      IC=IND*TCN*FSE*FPRIX
      VSURFI=(IC-ID)*SUI
      VEXP=(VSURFL+VSURFI)/SURF
      FACP=TABHL(FACPT,VEXP,0.,0.1,0.0125)
      ACPRIX=CLIP(PRIX*TACPN,PRIX*TACPN*FACP,0.,VEXP)
      ATTR=ATTRN*ASL*ASE
      DPT=POP*TEMN/ATTR
C EQUATIONS DES TAUX DELAIS
      DY(3)=TDELY3(SDN,EN2,EN3,EMPS,DL1,DL2)
      DY(2)=DL2
      DY(1)=DL1
C AUTRES EQUATIONS DE TAUX
      DY(4)=ATTR-ATTRP
      DY(5)=RSE-RSEP
      DY(6)=IC-ID
      DY(7)=ACPRIX
      DY(8)=LC-LD
      RETURN
      END
C
      SUBROUTINE POP(SM,MN,DPOP)
C
C SOUS-REGION VILLE,DYNAMIQUE POPULATION
C       ***     ***                *****
C
      REAL MN
      DPOP=SM-MN
      RETURN
      END
C
C ***************************************************
C FONCTIONS DU LANGAGE DE DESCRIPTION
C AJOUTES NORMALEMENT LORS DE LA PRECOMPILATION
C ***************************************************
C
      FUNCTION DELAY3(L1,L2,L3,EN2,EN3,DEL)
C FONCTION DELAY MATERIEL DE 3E. ORDRE
C
      REAL L1,L2,L3
      DELTA=DEL/3
      IN2=L1/DELTA
      IN3=L2/DELTA
      DELAY3=L3/DELTA
      RETURN
      END
C
      FUNCTION TDELY3(ENTRE,EN2,EN3,DELAY3,DL1,DL2)
C FONCTION FLUX CORRESPONDANT AU DELAY MATERIEL DE 3E. ORDRE
C
      DL1=ENTRE-EN2
      DL2=EN2-EN3
      TDELY3=EN3-DELAY3
      RETURN
      END
C
```

PAGE 006

```
      FUNCTION CLIP(A1,A2,SWT,TI)
C FONCTION AIGUILLAGE
      IF (TI.LT.SWT) GOTO 1
      CLIP=A2
      RETURN
    1 CLIP=A1
      RETURN
      END
C
      FUNCTION TABHL(T,X,XINF,XSUP,XINCR)
C FONCTION INTERPOLATION DU TABLEAU T SELON LA VARIABLE X
      DIMENSION T(1)
      Y=AMIN1(X,XSUP-1.E-25)
      Y=AMAX1(Y,XINF)
      K=IFIX((Y-XINF)/XINCR)+1
      A=(T(K+1)-T(K))/XINCR
      TABHL=A*(Y-(K-1)*XINCR-XINF)+T(K)
      RETURN
      END
```

J E U D E D O N N E E S

```
C ***************
      BLOCK DATA
C ***************
C DIMENSION ET VALEURS INITIALES DES VARIABLES D'ETAT
      COMMON /STATE/ Y,DY
C  Y = VALEUR DES VARIABLES D'ETAT
C DY =VALEURS DES VARIABLES DE FLUX
      COMMON /DIM/JDIMY
C  JDIMY = NOMBRE DES VARIABLES D'ETAT
      COMMON /AUTRES/ FPRIXT,FSLT,NMSLT,ASET,ASLT,FCLRLT,FACPT,FCPT
     1,EMPUI,SONPP,TMN,ULPP,LCN,DVML,TACPN,SCL,SUI,TCN,DVMI,TACP,TEMN
     2,TIMN,ATTRN,ATENV,TENMB,TEMNB,DEL,PRIXI
      REAL Y(50),DY(50)
      REAL LCN,VIS(11)
      REAL FPRIXT(13),FSLT(9),NMSLT(9),ASET(9),ASLT(9),FCLRLT(9),FACPT(9
     *),FCPT(5)
      INTEGER JDIMY
      DATA JDIMY/45/
C INITIALISATION DES VARIABLES D'ETAT
C              DELAI(EMPS)     ATTRP   RSEP     IND    PRIX     LOG       POP
      DATA Y/3*2750C.0000,  2.0,     1.0,    41000.,10.0, 55000.,  16500.,
     1        3*7500.0,      2.0,     1.0,    11000.,10.0,15000.0,  45000.
     2     3*128333.333,     2.0,     1.0,   182000.,10.0,256000.,770000.,
     3     3*46666.6666,     2.0,     1.0   ,70000.,10.0,  94000.0,280000.,
     4     3*8333.3333,      2.0,    1.0,    12500.,10.0, 16000.,50000./
C VALEURS DES CONSTANTS AU NIVEAU INTER-URBAIN
      DATA ATENV/.5/,TENMB/0.02/,TEMNB/0.02/,DEL/5.0/,PRIXI/10.0/
C VALEURS CONSTANTS MODELE VILLE
      DATA EMPUI/1./, SONPP/.10/, TMN/.007/, ULPP/.3/, LCN/.05/,
     1DVML/120./, TACPN/.05/, SCL/.035/, SUI/.012/, TCN/.03/,
     2DVMI/50./, TACP/0.4/, TEMN/.02/, TIMN/0.05/, ATTRN/1./
C FONCTIONS TABLEAUX
      DATA FPRIXT/2.,2.,1.9,1.8,1.7,1.6,1.5,1.4,1.3,1.2,1.1,1.,1./
      DATA FSLT/2.,1.9,1.8,1.7,1.2,.9,.6,.4,.3/
      DATA NMSLT/.3,.35,.5,.6,1.,1.10,1.20,1.30,2./
      DATA ASET/.2,.2,.25,.4,1.,1.6,1.75,1.8,2./
      DATA ASLT/0.,.1,.25,.6,1.,1.4,1.6,1.7,1.75/
      DATA FCLRLT/2.5,2.4,2.3,2.,1.,.37,.2,.1,.05/
      DATA FACPT/1.,1.2,1.5,1.7,2.,2.4,3.,3.8,5./
      DATA FCPT/3.,2.6,1.8,0.6,.25/
      END
C
```

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

dernière page de la thèse

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 Avril 1974,

VU le rapport de présentation de :

- Monsieur Jean MERMET, Chargé de Recherche au Centre National de la Recherche Scientifique -GRENOBLE-

Monsieur Francisco Eduardo RIVERA PORTO

est autorisé à présenter une thèse en soutenance pour l'obtention du titre de DOCTEUR de TROISIEME CYCLE, spécialité "Génie Informatique".

A Grenoble, le 15 Mars 1977

Ph. TRAYNARD